Unit -1

Introduction to wsn:

Unique Constraints and Challenges

Unlike a centralized system, a sensor network is subject to a unique set of resource constraints such as finite on-board battery power and limited network communication bandwidth. In a typical sensor network, each sensor node operates untethered and has a microprocessor and a small amount of memory for signal processing and task scheduling. Each node is also equipped with one or more sensing devices such as acoustic microphone arrays, video or still cameras, infrared (IR), seismic, or magnetic sensors. Each sensor node communicates wirelessly with a few other local nodes within its radio communication range. Sensor networks extend the existing Internet deep into the physical environment. The resulting new network is orders of magnitude more expansive and dynamic than the current TCP/IP network and is creating entirely new types of traffic that are quite different from what one finds on the Internet now. Information collected by and transmitted on a sensor network describes conditions of physical environments for example, temperature, humidity, or vibration and requires advanced query interfaces and search engines to effectively support user-level functions. Sensor networks may internetwork with an IP core network via a number of gateways, as in Figure 1.1. A gateway routes user queries or commands to appropriate nodes in a sensor network. It also routes sensor data, at times aggregated and summarized, to users who have requested it or are expected to utilize the information. A data repository or storage service may be present at the gateway, in addition to data logging at each sensor.

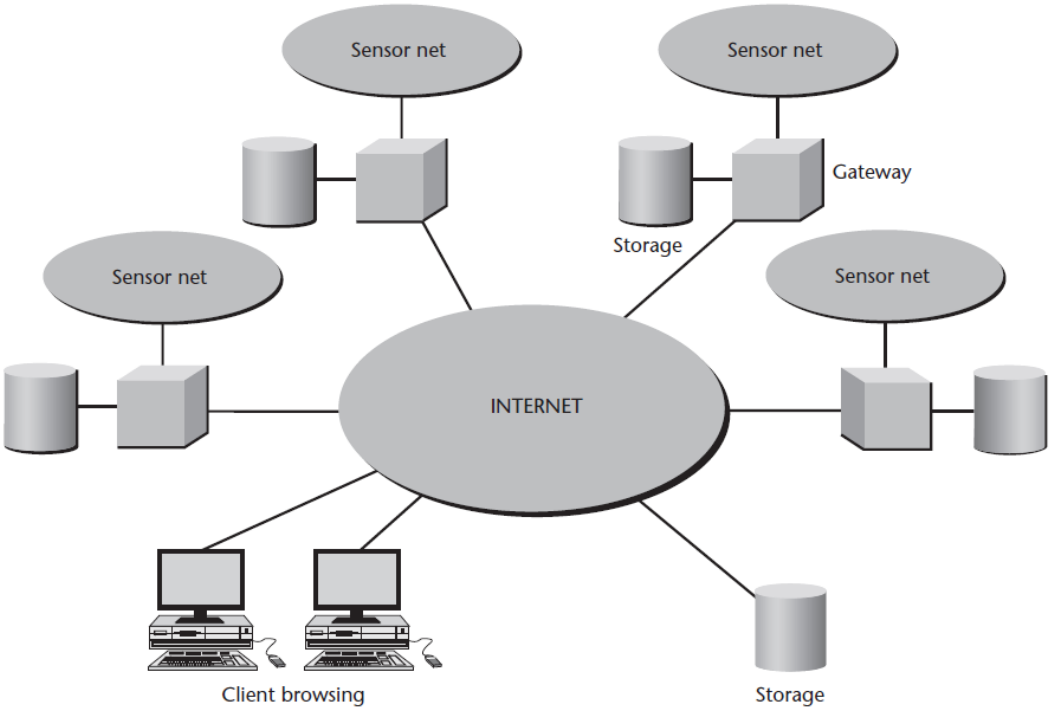
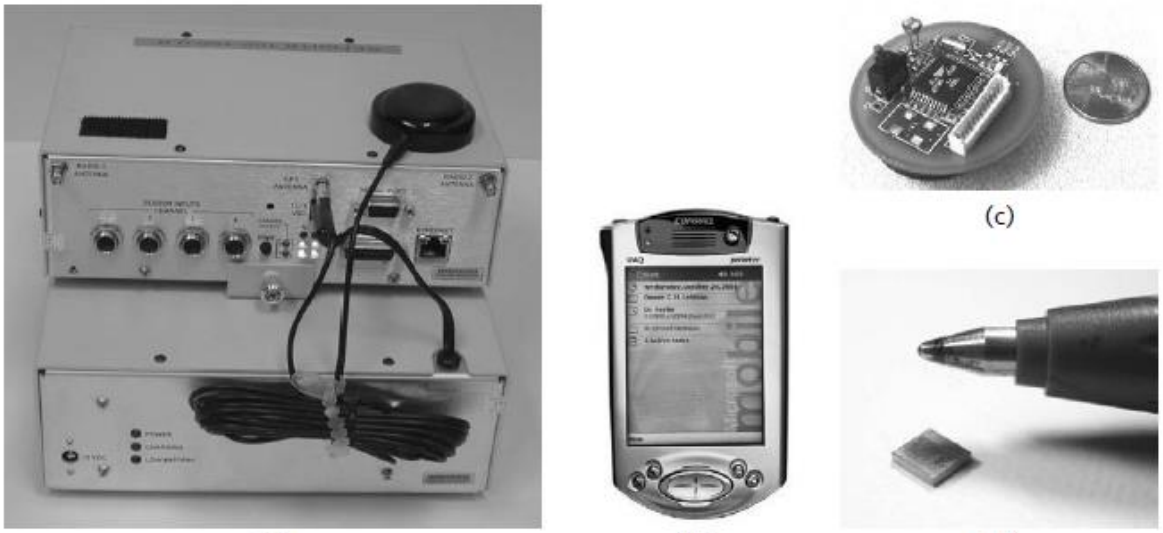


Figure 1.1 Sensor networks significantly expand the existing Internet into physical spaces. The data processing, storage, transport, querying, as well as the internetworking between the TCP/IP and sensor networks present a number of interesting research challenges that must be addressed from a multidisciplinary, cross-layer perspective.

The repository may serve as an intermediary between users and sensors, providing a persistent data storage. Additionally, one or more data storage devices may be attached to the IP network, to archive sensor data from a number of edge sensor networks and to support a variety of user-initiated browsing and search functions. The current generation of wireless sensor hardware ranges from shoe-box-sized Sensoria WINS NG sensors [158] with an SH-4 microprocessor to matchbox-sized Berkeley motes with an 8-bit microcontroller . A few samples of sensor hardware are shown in Figure 1.2; their corresponding capabilities are summarized and compared in Table 1.1. It is well known that communicating 1 bit over the wireless medium at short ranges consumes far more energy thanprocessing that bit. For the Sensoria sensors and Berkeley motes, the ratio of energy consumption for communication and computation is in the range of 1000 to 10,000. Despite the advances in silicon fabrication technologies, wireless communication will continue to dominate the energy consumption of networked embedded systems for the foreseeable future . Thus, minimizing the amount and range of communication as much as possible—for example, through local collaboration among sensors, duplicate data

suppression, or invoking only the nodes that are relevant to a given task—can significantly prolong the life of a sensor network and leave nodes free to support multiuser operations.

In addition, the shorter RF transmission range improves spectrum usage and increases throughput for a sensor network. The information management and networking for this new network will require more than just building faster routers, switchers, and browsers. A sensor network is designed to collect information from a physical environment. Networking will be intimately coupled with the needs of sensing and control, and hence the application



gure 1.2 Samples of wireless sensor hardware: (a) Sensoria WINS NG 2.0 sensor node; (b) HP iPAQ with 802.11b and microphone; (c) Berkeley/Crossbow sensor mote, alongside a U.S. penny; (d) An early prototype of Smart Dust MEMS integrated sensor, being developed at UC Berkeley. (*Picture courtesy of Kris Pister and Jason Hill*).

Table 1.1 Comparison of the four sensor platforms shown in Figure 1.2.

	WINS NG 2.0 Node	iPAQ with 802.11 and A/D Cards in Sleeve	Berkeley MICA Mote*	Smart Dust**
Parts cost*** (quantity 1000+)	$100s	$100s	$10s	<$1
Size (cm^3)	5300	600	40	.002
Weight (g) (including battery)	5400	350	70	.002
Battery capacity (kJ)	300	35	15	(Less)
Sensors	Off-board	Microphone & light sensors integrated, others off-board	Integrated on PCB: Acceleration, temperature, light, sound	MEMS sensors to be integrated
Memory	32 MB RAM, 32 MB flash	64 MB RAM, 32 MB flash	4 KB RAM, 128 KB flash	(Less)
CPU	Hitachi SH4	StrongARM or XScale	ATmega 103L	(Less powerful)
Operating system	Linux	WinCE or Linux	TinyOS	(smaller)
Processing capability	400 MIPS/ 1.4 GFLOPS	240 MIPS	4 MIPS	(Less)
Radio range	100 m	100 m	30 m	(Shorter)

semantics. To optimize for performance and resources such as energy, one has to rethink the existing TCP/IP stack and design an appropriate sensor network abstraction to support application development. For example, in many applications, it is more appropriate to address nodes in a sensor network by physical properties, such as node locations or proximity, than by IP addresses. How and where data is generated by sensors and consumed by users will affect the way data is compressed, routed, and aggregated. Because of the peer-to-peer connectivity and the lack of a global infrastructure support, the sensors have to rely on discovery protocols to construct local models about the network and environment. Mobility and instability in wireless links preclude the use of many existing edge-network gateway protocols for internetworking IP and sensor networks.

To summarize, the challenges we face in designing sensor network systems and applications include:

• *Limited hardware:* Each node has limited processing, storage, and communication capabilities, and limited energy supply and bandwidth.

• *Limited support for networking:* The network is peer-to-peer, with a mesh topology and dynamic, mobile, and unreliable connectivity. There are no universal routing protocols or central registry services. Each node acts both as a router and as an application host.

• *Limited support for software development:* The tasks are typically real-time and massively distributed, involve dynamic collaboration among nodes, and must handle multiple competing events.

Global properties can be specified only via local instructions. Because of the coupling between applications and system layers, the software architecture must be codesigned with the information processing architecture.

1.2 Advantages of Sensor Networks

Networked sensing offers unique advantages over traditional centralized approaches. Dense networks of distributed communicating sensors can improve signal-to-noise ratio (SNR) by reducing average distances from sensor to source of signal, or target. Increased energy efficiency in communications is enabled by the multihop topology of the network [184]. Moreover, additional relevant information from other sensors can be aggregated during this multihop transmission through in-network processing [104]. But perhaps the greatest advantages of networked sensing are in improved robustness and scalability. A decentralized sensing system is inherently more robust against individual sensor node or link failures, because of redundancy in the network. Decentralized algorithms are also far more scalable in practical deployment and may be the only way to achieve the large scales needed for some applications.

Energy Advantage

Because of the unique attenuation characteristics of radio-frequency (RF) signals, a multihop RF network provides a significant energy saving over a single-hop network for the same distance. Consider the following simple example of an N-hop network. Assume the overall distance for transmission is Nr, where r is the one-hop distance. The minimum receiving power at a node for a given transmission error rate is $Preceive$, and the power at a transmission node is $Psend$. Then, the RF attenuation model near the ground is given by

$$Preceive \propto Psend r^{\alpha},$$

where r is the transmission distance and α is the RF attenuation exponent. Due to multipath and other interference effects, α is typically in the range of 2 to 5. Equivalently,

$Psend \propto r\alpha Preceive$.
Therefore, the power advantage of an N-hop transmission versus a single-hop transmission over the same distance Nr is

$$\begin{aligned}\eta rf &= Psend(Nr)N \cdot Psend(r)\\ &= (Nr)\alpha PreceiveN \cdot r\alpha Preceive\\ &= N\alpha - 1.\end{aligned}$$

Figure 1.3 illustrates the power attenuation for the multihop and single-hop networks. A larger N gives a larger power saving due to the consideration of RF energy alone. However, this analysis ignores the power usage by other components of an RF circuitry. Using nodes increases not only the cost, but also the power consumption of these other RF components. In practice, an optimal design seeks to balance the two conflicting factors for an overall cost and energy efficiency. Latency and robustness considerations may also argue against an unduly large number of relay nodes.

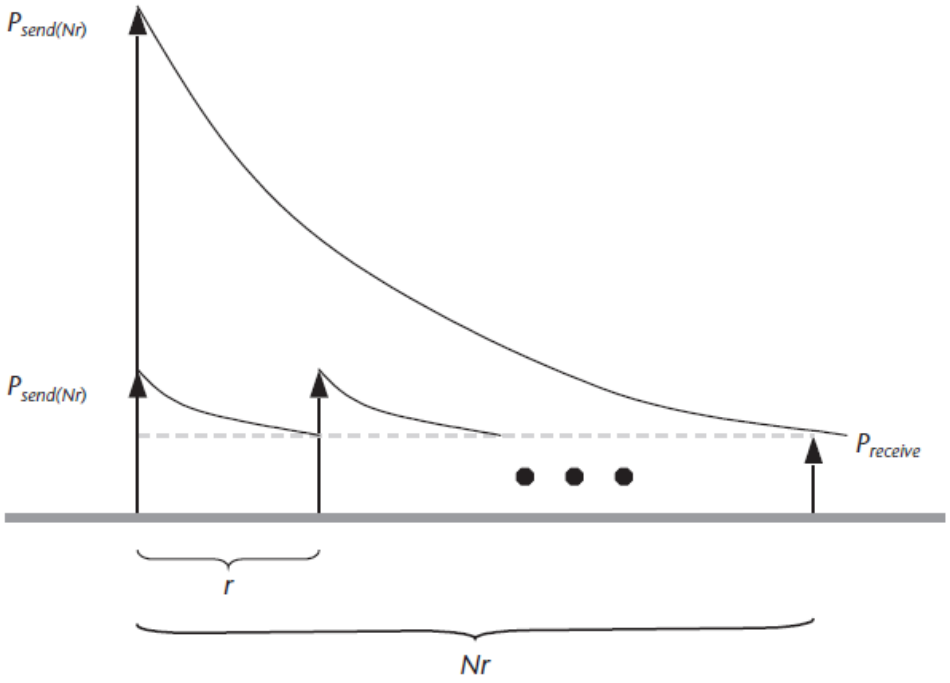


Figure 1.3 The power advantage of using a multihop RF communication over a distance of Nr.

1.2.2 Detection Advantage

Each sensor has a finite sensing range, determined by the noise floor at the sensor.A denser sensor field improves the odds of detecting a signal source within the range. Once a signal source is inside the sensing range of a sensor, further increasing the sensor density decreases the average distance from a sensor to the signal source, hence improving the signal-to-noise ratio (SNR). Let us consider the acoustic sensing case in a two-dimensional plane, where the acoustic power received at a distance r is

power received at a distance r is

$$P_{receive} \propto \frac{P_{source}}{r^2},$$

which assumes an inverse distance squared attenuation. The SNR is given by

$$\mathrm{SNR}_r = 10 \log \frac{P_{receive}}{P_{noise}} = 10 \log P_{source} - 10 \log P_{noise} - 20 \log r.$$

Increasing the sensor density by a factor of k reduces the average distance to a target by a factor of $\frac{1}{\sqrt{k}}$. Thus, the SNR advantage of the denser sensor network is

$$\eta_{snr} = \mathrm{SNR}_{\frac{r}{\sqrt{k}}} - \mathrm{SNR}_r = 20 \log \frac{r}{\frac{r}{\sqrt{k}}} = 10 \log k. \qquad (1.2)$$

Therefore, an increase in sensor density by a factor of k improves the SNR at a sensor by $10 \log k$ db.

1.3 Sensor Network Applications

A sensor network is designed to perform a set of high-level information processing tasks such as detection, tracking, or classification. Measures of performance for these tasks are well defined, including detection of false alarms or misses, classification errors, and track quality. Applications of sensor networks are wide ranging and can vary significantly in application requirements, modes of deployment (e.g., ad hoc versus instrumented environment), sensing modality, or means of power supply (e.g., battery versus wall-socket). Sample commercial and military applications include:

- Environmental monitoring (e.g., traffic, habitat, security)
- Industrial sensing and diagnostics (e.g., appliances, factory, supply chains)
- Infrastructure protection (e.g., power grids, water distribution)
- Battlefield awareness (e.g., multitarget tracking)
- Context-aware computing (e.g., intelligent home, responsive environment)

1.4 Collaborative Processing

As the above applications have illustrated, many sensing tasks require a sensor network system to process data cooperatively and to combine information from multiple sources. In traditional centralized sensing and signal processing systems, raw data collected by sensors are relayed to the edges of a network where the data is processed.

From the scalability point of view, the nonlocal processing at the edges depletes precious bandwidth. If every sensor has some data that it needs to send to another node in a network, then a wellknown wireless capacity result by Gupta and Kumar states that the per node throughput scales as $\sqrt{1}N$ —in other words, it goes to zero as the number of nodes N in a wireless ad hoc network increases .

This result holds regardless of optimality in routing, power control, or transmission. Intuitively, this states that, as the number of nodes increases, every node spends almost all of its time forwarding packets of other nodes. From the energy point of view, transmitting raw data to distant nodes is wasteful of scarce resources. The diminishing wireless capacity result can be somewhat mitigated by introducing mobility to nodes, if an application is delay-tolerant . In a sensor network context, one can clearly do better. Since data from multiple sensors with overlapping sensing regions is almost always correlated, one can remove the redundant information in the data, through in-network aggregation and compression local to the nodes that generate the data, before shipping it to a remote node. In fact, it can be shown that the amount of non redundant data that a network generates grows as $O(\log N)$, assuming that the network is sampling a physical phenomenon with a prescribed accuracy requirement [206]. This is encouraging since the amount of data generated per node scales as $O\_\log N.N\_$, which is within the per-node throughput constraint derived by Gupta and Kumar. On the other hand, instead of applying data compression techniques to samples after they are collected, nodes can be more selective in what data to generate or communicate. For energy-constrained and multiuser decentralized systems, it becomes critical to carefully select the sensor nodes that participate in a sensor collaboration, balancing the information contribution of each against its resource consumption or potential utility for other users. We use the term *collaborative signal and information processing* (CSIP) to refer to signal and information processing problems dominated by this issue of selecting embedded sensors to participate in an information processing task .

1.5 Key Definitions of Sensor Networks

Sensor networks is an interdisciplinary research area that draws on contributions from signal processing, networking and protocols, databases and information management, distributed algorithms, and embedded systems and architecture. In the following, we define a number of key terms and concepts that will be used throughout the book as we develop techniques and examples for sensor networks.

• *Sensor:* A transducer that converts a physical phenomenon such as heat, light, sound, or motion into electrical or other signals that may be further manipulated by other apparatus.

• *Sensor node:* A basic unit in a sensor network, with on-board sensors, processor, memory, wireless modem, and power supply. It is often abbreviated as *node*. When a node has only a single sensor on board, the node is sometimes also referred to as a *sensor*, creating some confusion.

• *Network topology:* A connectivity graph where nodes are sensor nodes and edges are communication links. In a wireless network, the link represents a one-hop connection, and the neighbors of a node are those within the radio range of the node.

• *Routing:* The process of determining a network path from a packet source node to its destination.

• *Date-centric:* Approaches that name, route, or access a piece of data via properties, such as physical location, that are external to a communication network. This is to be contrasted with addresscentric approaches which use logical properties of nodes related to the network structure.

• *Geographic routing:* Routing of data based on geographical attributes such as locations or regions. This is an example of datecentric networking.

• *In-network:* A style of processing in which the data is processed and combined near where the data is generated.

• *Collaborative processing:* Sensors cooperatively processing data from multiple sources in order to serve a high-level task. This typically requires communication among a set of nodes.

• *State:* A snapshot about a physical environment (e.g., the number of signal sources, their locations or spatial extent, speed of

movement), or a snapshot of the system itself (e.g.,the network state).

• *Uncertainty:* A condition of the information caused by noise in sensor measurements, or lack of knowledge in models. The uncertainty affects the system's ability to estimate the state accurately

and must be carefully modeled. Because of the ubiquity of uncertainty in the data, many sensor network estimation problems are cast in a statistical framework. For example, one may use a covariance matrix to characterize the uncertainty in a Gaussian-like process or more general probability distributions for non-Gaussian processes.

• *Task:* Either high-level system tasks which may include sensing, communication, processing, and resource allocation, or application tasks which may include detection, classification, localization, or tracking.

• *Detection:* The process of discovering the existence of a physical phenomenon. A threshold-based detector may flag a detection whenever the signature of a physical phenomenon is determined to be significant enough compared with the threshold.

• *Classification:* The assignment of class labels to a set of physical phenomena being observed.

• *Localization and tracking:* The estimation of the state of a physical entity such as a physical phenomenon or a sensor node from a set of measurements. Tracking produces a series of estimates over time.

• *Value of information or information utility:* A mapping of data to a scalar number, in the context of the overall system task and knowledge. For example, information utility of a piece of sensor data may be characterized by its relevance to an estimation task at hand and computed by a mutual information function.

• *Resource:* Resources include sensors, communication links, processors, on-board memory, and node energy reserves. Resource allocation assigns resources to tasks, typically optimizing some performance objective.

• *Sensor tasking:* The assignment of sensors to a particular task and the control of sensor state (e.g., on/off, pan/tilt) for accomplishing the task.

• *Node services:* Services such as time synchronization and node localization that enable applications to discover properties of a node and the nodes to organize themselves into a useful network.

• *Data storage:* Sensor information is stored, indexed, and accessed by applications. Storage may be local to the node where the data is generated, load-balanced across a network, or anchored at a few points (warehouses).

• *Embedded operating system (OS):* The run-time system support for sensor network applications. An embedded OS typically provides an abstraction of system resources and a set of utilities

• *System performance goal:* The abstract characterization of system properties. Examples include scalability, robustness, and network longevity, each of which may be measured by a set of evaluation metrics.

• *Evaluation metric:* A measurable quantity that describes how well the system is performing on some absolute scale. Examples include packet loss (system), network dwell time (system), track loss (application), false alarm rate (application), probability of correct association (application), location error (application), or processing latency (application/system). An evaluation method is a process for comparing the value of applying the metrics on an experimental system with that of some other benchmark system.

Enabling technologies for wireless sensor networks

Building such wireless sensor networks has only become possible with some fundamental advances in enabling technologies. First and foremost among these technologies is the miniaturization of hardware. Smaller feature sizes in chips have driven down the power consumption of the basic components of a sensor node to a level that the constructions of WSNs can be contemplated. This is particularly relevant to microcontrollers and memory chips as such, but also, the radio modems, responsible for wireless communication, have become much more energy efficient. Reduced chip size and improved energy efficiency is accompanied by reduced cost, which is necessary to make redundant deployment of nodes affordable.

These three basic parts of a sensor node have to accompanied by power supply. This requires, depending on application, high capacity batteries that last for long times, that is, have only a negligible self-discharge rate, and that can efficiently provide small amounts of current. Ideally, a sensor node also has a device for **energy scavenging**, recharging the battery with energy gathered from the environment – solar cells or vibration-based power generation are conceivable options. Such a concept requires the battery to be efficiently chargeable with small amounts of

current, which is not a standard ability. Both batteries and energy scavenging are still objects of ongoing research. The counterpart to the basic hardware technologies is software. The first question to answer here is the principal division of tasks and functionalities in a single node – the architecture of the operating system or runtime environment. This environment has to support simple retasking, cross-layer information exchange, and modularity to allow for simple maintenance. This software architecture on a single node has to be extended to a network architecture, where the division of tasks between nodes, not only on a single node, becomes the relevant question – for example, how to structure interfaces for application programmers. The third part to solve then is the question of how to design appropriate communication protocols.

Single-node architecture

2.1 Hardware components

2.1.1 Sensor node hardware overview

When choosing the hardware components for a wireless sensor node, evidently the application's requirements play a decisive factor with regard mostly to size, costs, and energy consumption of the nodes – communication and computation facilities as such are often considered to be of acceptable quality, but the trade-offs between features and costs is crucial. In some extreme cases, an entire sensor node should be smaller than 1 cc, weigh (considerably) less than 100 g, be substantially cheaper than US$1, and dissipate less than 100 μW . In even more extreme visions, the nodes are sometimes claimed to have to be reduced to the size of grains of dust. In more realistic applications, the mere size of a node is not so important; rather, convenience, simple power supply, and cost are more important .

These diversities notwithstanding, a certain common trend is observable in the literature when looking at typical hardware platforms for wireless sensor nodes. While there is certainly not a single standard available, nor would such a standard necessarily be able to support all application types, this section will survey these typical sensor node architectures. In addition, there are a number of research projects that focus on shrinking any of the components in size, energy consumption, or costs, based on the fact that custom off-the-shelf components do currently not

live up to some of the more stringent application requirements. But as this book focuses on the networking aspects of WSNs, these efforts are not discussed here.

A basic sensor node comprises five main components (Figure):

Controller :A controller to process all the relevant data, capable of executing arbitrary code.

Memory : Some memory to store programs and intermediate data; usually, different types of memory are used for programs and data.

Sensors and actuators: The actual interface to the physical world: devices that can observe or control physical parameters of the environment.

Communication : Turning nodes into a network requires a device for sending and receiving information over a wireless channel.

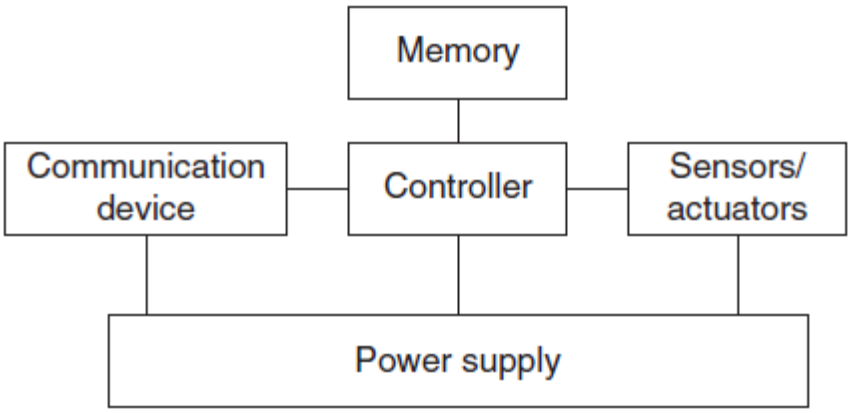


Figure 2.1 Overview of main sensor node hardware components

Power supply As usually no tethered power supply is available, some form of batteries are necessary to provide energy. Sometimes, some form of recharging by obtaining energy from the environment is available as well (e.g. solar cells).

Each of these components has to operate balancing the trade-off between as small an energy consumption as possible on the one hand and the need to fulfill their tasks on the other hand.

For example, both the communication device and the controller should be turned off as long as possible. To wake up again, the controller could, for example, use a preprogrammed timer to be reactivated after some time. Alternatively, the sensors could be programmed to raise an interrupt if a given event occurs – say, a temperature value exceeds a given threshold or the communication device detects an incoming transmission.

2.1.2 Controller

Microcontrollers versus microprocessors, FPGAs, and ASICs

The controller is the core of a wireless sensor node. It collects data from the sensors, processes this data, decides when and where to send it, receives data from other sensor nodes, and decides on the actuator's behavior. It has to execute various programs, ranging from time-critical signal processing and communication protocols to application programs; it is the Central Processing Unit (CPU) of the node. Such a variety of processing tasks can be performed on various controller architectures, representing trade-offs between flexibility, performance, energy efficiency, and costs. One solution is to use general-purpose processors, like those known from desktop computers. These processors are highly overpowered, and their energy consumption is excessive. But simpler processors do exist, specifically geared toward usage in embedded systems. These processors are commonly referred as **microcontrollers**. Some of the key characteristics why these microcontrollers are particularly suited to embedded systems are their flexibility in connecting with other devices (like sensors), their instruction set amenable to time-critical signal processing, and their typically low power consumption; they are also convenient in that they often have memory built in. In addition, they are freely programmable and hence very flexible. Microcontrollers are also suitable for WSNs since they commonly have the possibility to reduce their power consumption by going into **sleep states** where only parts of the controller are active; details vary considerably between different controllers. Details regarding power consumption and energy efficiency are discussed in Section 2.2. One of the main differences to general-purpose systems is that microcontroller-based systems usually do not feature a memory management unit, somewhat limiting the functionality of memory – for example, protected or virtual memory is difficult, if not impossible, to achieve.

A specialized case of programmable processors are Digital Signal Processors (DSPs). They are specifically geared, with respect to their architecture and their instruction set, for processing large amounts of vectorial data, as is typically the case in signal processing applications. In a wireless sensor node, such a DSP could be used to process data coming from a simple analog, wireless communication device to extract a digital data stream. In broadband wireless communication, DSPs are an appropriate and successfully used platform. But in wireless sensor networks, the **20** Single-node architecture requirements on wireless communication are usually much more modest (e.g. simpler, easier to process modulations are used that can be efficiently handled in hardware by the communication device itself) and the signal processing tasks related

to the actual sensing of data is also not overly complicated. Hence, these advantages of a DSP are typically not required in a WSN node and they are usually not used.

Another option for the controller is to depart from the high flexibility offered by a (fairly generalpurpose) microcontroller and to use Field-Programmable Gate Arrays (FPGAs) or Application- Specific Integrated Circuits (ASICs) instead. An FPGA can be reprogrammed (or rather reconfigured) "in the field" to adapt to a changing set of requirements; however, this can take time and energy – it is not practical to reprogram an FPGA at the same frequency as a microcontroller could change between different programs. An ASIC is a specialized processor, custom designed for a given application such as, for example, high-speed routers and switches. The typical trade-off here is loss of flexibility in return for a considerably better energy efficiency and performance. On the other hand, where a microcontroller requires software development, ASICs provide the same functionality in hardware, resulting in potentially more costly hardware development. For a dedicated WSN application, where the duties of a the sensor nodes do not change over lifetime and where the number of nodes is big enough to warrant the investment in ASIC development, they can be a superior solution. At the current stage of WSN technology, however, the bigger flexibility and simpler usage of microcontrollers makes them the generally preferred solution. However, this is not necessarily the final solution as "convenient programmability over several orders of energy consumption and data processing requirements is a worthy research goal".

In addition, splitting processing tasks between some low-level, fixed functionality put into a very energy-efficient ASIC and high-level, flexible, relatively rarely invoked processing on a microcontroller is an attractive design and research option .

For the remainder of this book, a microcontroller-based architecture is assumed.

Some examples for microcontrollers

Microcontrollers that are used in several wireless sensor node prototypes include the Atmel processor or Texas Instrument's MSP 430. In older prototypes, the Intel StrongArm processors have also been used, but this is no longer considered as a practical option; it is included here for the sake of completeness. Nonetheless, as the principal properties of these processors and controllers are quite similar, conclusions from these earlier research results still hold to a large degree.

Intel StrongARM

The Intel StrongARM [379] is, in WSN terms, a fairly high-end processor as it is mostly geared toward handheld devices like PDAs. The SA-1100 model has a 32-bit Reduced Instruction Set Computer (RISC) core, running at up to 206 MHz.

Texas Instruments MSP 430

Texas Instrument provides an entire family of microcontrollers under the family designation MSP 430 . Unlike the StrongARM, it is explicitly intended for embedded applications. Accordingly, it runs a 16-bit RISC core at considerably lower clock frequencies (up to 4 MHz) but comes with a wide range of interconnection possibilities and an instruction set amenable to easy handling of peripherals of different kinds. It features a varying amount of on-chip RAM (sizes are 2–10 kB), several 12-bit analog/digital converters, and a real-time clock. It is certainly powerful enough to handle the typical computational tasks of a typical wireless sensor node (possibly with the exception of driving the radio front end, depending on how it is connected – bit or byte interface – to the controller).

Atmel ATmega

The Atmel ATmega 128L [28] is an 8-bit microcontroller, also intended for usage in embedded applications and equipped with relevant external interfaces for common peripherals.

2.1.3 Memory

The memory component is fairly straightforward. Evidently, there is a need for Random Access Memory (RAM) to store intermediate sensor readings, packets from other nodes, and so on. While RAM is fast, its main disadvantage is that it loses its content if power supply is interrupted. Program code can be stored in Read-Only Memory (ROM) or, more typically, in Electrically Erasable Programmable Read-Only Memory (EEPROM) or flash memory (the later being similar to EEPROM but allowing data to be erased or written in blocks instead of only a byte at a time). Flash memory can also serve as intermediate storage of data in case RAM is insufficient or when the power supply of RAM should be shut down for some time. The long read and write access delays of flash memory should be taken into account, as well as the high required energy.

Correctly dimensioning memory sizes, especially RAM, can be crucial with respect to manufacturing costs and power consumption. However, even general rules of thumbs are difficult to give as the memory requirements are very much application dependent.

2.1.4 Communication device

Choice of transmission medium

The communication device is used to exchange data between individual nodes. In some cases, wired communication can actually be the method of choice and is frequently applied in many sensor networklike settings (using field buses like Profibus, LON, CAN, or others). The communication devices for these networks are custom off-the-shelf components.

The case of wireless communication is considerably more interesting. The first choice to make is that of the transmission medium – the usual choices include radio frequencies, optical communication, and ultrasound; other media like magnetic inductance are only used in very specific cases.

Of these choices, Radio Frequency (RF)-based communication is by far the most relevant one as it best fits the requirements of most WSN applications: It provides relatively long range and high data rates, acceptable error rates at reasonable energy expenditure, and does not require line of sight between sender and receiver. Thus, RF-based communication and transceiver will receive the lion share of attention here; other media are only treated briefly at the end of this section.

For a practical wireless, RF-based system, the carrier frequency has to be carefully chosen. Chapter 4 contains a detailed discussion; for the moment, suffice it to say that wireless sensor networks typically use communication frequencies between about 433 MHz and 2.4 GHz.

The reader is expected to be familiar with the basics of wireless communication; a survey is included in Chapter 4.

Transceivers

For actual communication, both a transmitter and a receiver are required in a sensor node. The essential task is to convert a bit stream coming from a microcontroller (or a sequence of bytes or frames) and convert them to and from radio waves. For practical purposes, it is usually convenient to use a device that combines these two tasks in a single entity. Such combined devices are called **transceivers**. Usually, half-duplex operation is realized since transmitting and receiving at the same time on a wireless medium is impractical in most cases (the receiver would only hear the own transmitter anyway).

A range of low-cost transceivers is commercially available that incorporate all the circuitry required for transmitting and receiving – modulation, demodulation, amplifiers, filters, mixers,

and so on. For a judicious choice, the transceiver's tasks and its main characteristics have to be understood.

Transceiver tasks and characteristics

To select appropriate transceivers, a number of characteristics should be taken into account. The most important ones are: **Service to upper layer** A receiver has to offer certain services to the upper layers, most notably to the Medium Access Control (MAC) layer. Sometimes, this service is **packet oriented**; sometimes, a transceiver only provides a **byte interface** or even only a **bit interface** to the microcontroller. In any case, the transceiver must provide an interface that somehow allows the MAC layer to initiate frame transmissions and to hand over the packet from, say, the main memory of the sensor node into the transceiver (or a byte or a bit stream, with additional processing required on the microcontroller). In the other direction, incoming packets must be streamed into buffers accessible by the MAC protocol.

Power consumption and energy efficiency The simplest interpretation of energy efficiency is the energy required to transmit and receive a single bit. Also, to be suitable for use in WSNs, transceivers should be switchable between different states, for example, active and sleeping. The idle power consumption in each of these states and during switching between them is very important – details are discussed in Section 2.2.

Carrier frequency and multiple channels Transceivers are available for different carrier frequencies; evidently, it must match application requirements and regulatory restrictions. It is often useful if the transceiver provides several carrier frequencies ("channels") to choose from, helping to alleviate some congestion problems in dense networks. Such channels or "subbands" are relevant, for example, for certain MAC protocols (FDMA or multichannel CSMA/ ALOHA techniques,).

State change times and energy A transceiver can operate in different modes: sending or receiving, use different channels, or be in different power-safe states. In any case, the time and the energy required to change between two such states are important figures of merit. The turnaround time between sending and receiving, for example, is important for various medium access protocols.

Data rates Carrier frequency and used bandwidth together with modulation and coding determine the gross data rate. Typical values are a few tens of kilobits per second – considerably

less than in broadband wireless communication, but usually sufficient for WSNs. Different data rates can be achieved, for example, by using different modulations or changing the symbol rate.

Modulations The transceivers typically support one or several of on/off-keying, ASK, FSK, or similar modulations. If several modulations are available, it is convenient for experiments if they are selectable at runtime even though, for real deployment, dynamic switching between modulations is not one of the most discussed options.

Coding Some transceivers allow various coding schemes to be selected.

Transmission power control Some transceivers can directly provide control over the transmission power to be used; some require some external circuitry for that purpose. Usually, only a discrete number of power levels are available from which the actual transmission power can be chosen. Maximum output power is usually determined by regulations.

Noise figure The **noise figure** NF of an element is defined as the ratio of the Signal-to-Noise Ratio (SNR) ratio SNR*I* at the input of the element to the SNR ratio SNR*O* at the element's output:

NF = SNR*I*SNR*O*

Gain The **gain** is the ratio of the output signal power to the input signal power and is typically given in dB. Amplifiers with high gain are desirable to achieve good energy efficiency.

Power efficiency The **efficiency** of the radio front end is given as the ratio of the radiated power to the overall power consumed by the front end; for a power amplifier, the efficiency describes the ratio of the output signal's power to the power consumed by the overall power amplifier.

Receiver sensitivity The **receiver sensitivity** (given in dBm) specifies the minimum signal power at the receiver needed to achieve a prescribed *Eb/N*0 or a prescribed bit/packet error rate. Better sensitivity levels extend the possible range of a system.

Range While intuitively the range of a transmitter is clear, a formal definition requires some care. The range is considered in absence of interference; it evidently depends on the maximum transmission power, on the antenna characteristics, on the attenuation caused by the environment, which in turn depends on the used carrier frequency, on the modulation/coding scheme that is used, and on the bit error rate that one is willing to accept at the receiver. It also depends on the quality of the receiver, essentially captured by its sensitivity. Typical values are difficult to give here, but prototypes or products with ranges between a few meters and several hundreds of meters are available.

Blocking performance The blocking performance of a receiver is its achieved bit error rate in

the presence of an interferer. More precisely, at what power level can an interferer (at a fixed distance) send at a given offset from the carrier frequency such that target BER can still be met? An interferer at higher frequency offsets can be tolerated at large power levels. Evidently, blocking performance can be improved by interposing a filter between antenna and transceiver.

An important special case is an adjacent channel interferer that transmits on neighboring frequencies. The adjacent channel suppression describes a transceiver's capability to filter out signals from adjacent frequency bands (and thus to reduce adjacent channel interference) has a direct impact on the observed Signal to Interference and Noise Ratio (SINR).

Out of band emission The inverse to adjacent channel suppression is the out of band emission of a transmitter. To limit disturbance of other systems, or of the WSN itself in a multichannel setup, the transmitter should produce as little as possible of transmission power outside of its prescribed bandwidth, centered around the carrier frequency.

Carrier sense and RSSI In many medium access control protocols, sensing whether the wireless channel, the carrier, is busy (another node is transmitting) is a critical information. The receiver has to be able to provide that information. The precise semantics of this carriersense signal depends on the implementation. For example, the IEEE 802.15.4 standard distinguishes the following modes:

• The received energy is above threshold; however, the underlying signal does not need to comply with the modulation and spectral characteristics.

• A carrier has been detected, that is, some signal which complies with the modulation.

• Carrier detected and energy is present.

Also, the signal strength at which an incoming data packet has been received can provide useful information (e.g. a rough estimate about the distance from the transmitter assuming the transmission power is known); a receiver has to provide this information in the Received Signal Strength Indicator (RSSI).

Frequency stability : The **frequency stability** denotes the degree of variation from nominal center frequencies when environmental conditions of oscillators like temperature or pressure change. In extreme cases, poor frequency stability can break down communication links, for example, when one node is placed in sunlight whereas its neighbor is currently in the shade.

Voltage range Transceivers should operate reliably over a range of supply voltages. Otherwise,

inefficient voltage stabilization circuitry is required. Transceivers appropriate for WSNs are available from many manufacturers. Usually, there is an entire family of devices to choose from, for example, customized to different regulatory restrictions on carrier frequency in Europe and North America. Currently popular product series include the RFM TR 1001, the Chipcon CC 1000 and CC 2420 (as one of the first IEEE 802.15.4 compliant models), and the Infineon TDA525x family, to name but a few. They are described in a bit more detail at the end of this section. An important peculiarity and a key difference compared to other communication devices is the fact that these simple transceivers often lack a unique identifier: each Ethernet device, for example, has a MAC-level address that uniquely identifies this individual device. For simple transceivers, the additional cost of providing such an identifier is relatively high with respect to the device's total costs, and thus, unique identifiers cannot be relied upon to be present in all devices. The availability of such device identifiers is very useful in many communication protocols and their absence will have considerable consequences for protocol design. Improving these commercial designs to provide better performance at lower energy consumption and reduced cost is an ongoing effort by a large research community, facing challenges such as low transistor transconductance or limitations of integrated passive RF components. As these hardwarerelated questions are not the main focus of this book, the reader is referred to other material .

Transceiver structure

A fairly common structure of transceivers is into the Radio Frequency (RF) front end and the baseband part:

• the **radio frequency front end** performs analog signal processing in the actual radio frequency band, whereas

• the **baseband processor** performs all signal processing in the digital domain and communicates with a sensor node's processor or other digital circuitry.

Between these two parts, a frequency conversion takes place, either directly or via one or several Intermediate Frequencys (IFs). The boundary between the analog and the digital domain is constituted by Digital/Analog Converters (DACs) and Analog/Digital Converters (ADCs).

A detailed discussion of the low-power design of RF front end and baseband circuitry is well beyond the scope of this book; one place to start with is reference [3].

The **RF front end** performs analog signal processing in the actual radio frequency band, for

example in the 2.4 GHz Industrial, Scientific, and Medical (ISM) band; it is the first stage of the interface between the electromagnetic waves and the digital signal processing of the further transceiver stages [46, 470]. Some important elements of an RF front ends architecture are sketched figure :

• The Power Amplifier (PA) accepts upconverted signals from the IF or baseband part and amplifies

them for transmission over the antenna.

• The Low Noise Amplifier (LNA) amplifies incoming signals up to levels suitable for further processing without significantly reducing the SNR [470]. The range of powers of the incoming signals varies from very weak signals from nodes close to the reception boundary to strong signals from nearby nodes; this range can be up to 100 dB. Without management actions, the LNA is active all the time and can consume a significant fraction of the transceiver's energy.

• Elements like local oscillators or voltage-controlled oscillators and mixers are used for frequency conversion from the RF spectrum to intermediate frequencies or to the baseband. The incoming signal at RF frequencies fRF is multiplied in a mixer with a fixed-frequency signal from the local oscillator (frequency fLO). The resulting intermediate-frequency signal has frequency fLO − fRF.

Depending on the RF front end architecture, other elements like filters are also present.

The efficiency of RF front ends in wireless sensor networks is discussed in Section 4.3

.

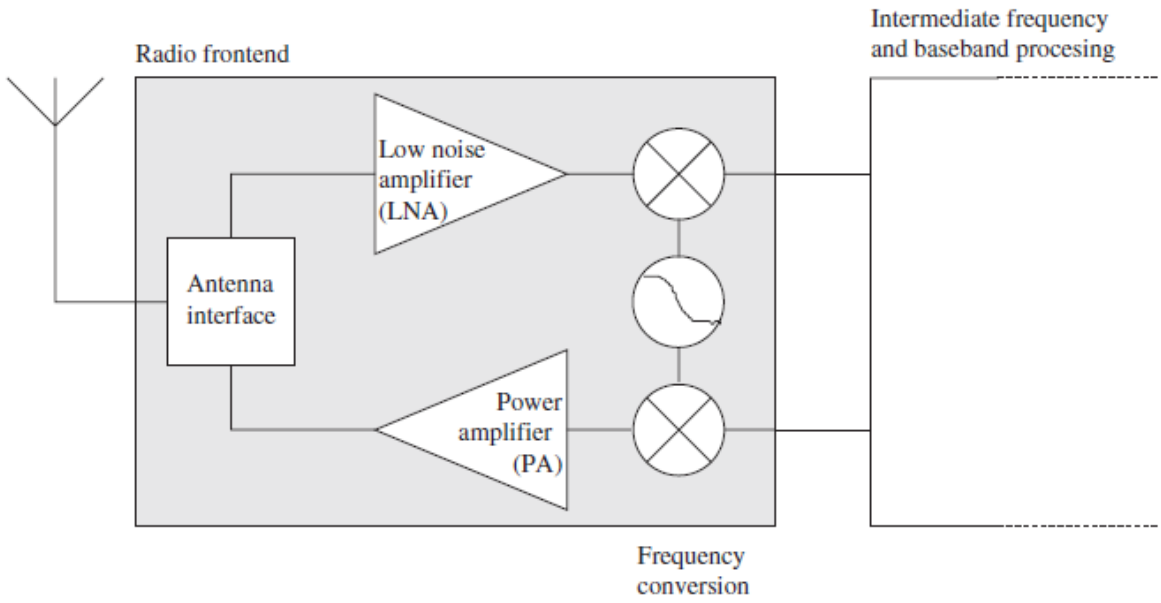


Figure 2.2 RF front end [46]

Transmit : In the **transmit state**, the transmit part of the transceiver is active and the antenna radiates energy.

Receive : In the **receive state** the receive part is active.

Idle :A transceiver that is ready to receive but is not currently receiving anything is said to be in an **idle state**. In this idle state, many parts of the receive circuitry are active, and others can be switched off. For example, in the synchronization circuitry, some elements concerned with acquisition are active, while those concerned with tracking can be switched off and activated only when the acquisition has found something. Myers et al. [580] also discuss techniques for switching off parts of the acquisition circuitry for IEEE 802.11 transceivers. A major source of power dissipation is **leakage**.

Sleep: In the **sleep state**, significant parts of the transceiver are switched off. There are transceivers offering several different sleep states, see reference [580] for a discussion of sleep states for IEEE 802.11 transceivers. These sleep states differ in the amount of circuitry switched off and in the associated **recovery times** and **startup energy** [855]. For example, in a complete power down of the transceiver, the startup costs include a complete initialization as well as configuration of the radio, whereas in “lighter” sleep modes, the clock driving certain transceiver parts is throttled down while configuration and operational state is remembered.

2.1.5 Sensors and actuators

Without the actual sensors and actuators, a wireless sensor network would be beside the point entirely. But as the discussion of possible application areas has already indicated, the possible range of sensors is vast. It is only possible to give a rough idea on which sensors and actuators can be used in a WSN.

Sensors

Sensors can be roughly categorized into three categories:

Passive, omnidirectional sensors : These sensors can measure a physical quantity at the point of the sensor node without actually manipulating the environment by active probing – in this sense, they are passive. Moreover, some of these sensors actually are self-powered in the sense that they obtain the energy they need from the environment – energy is only needed to amplify their analog signal. There is no notion of "direction" involved in these measurements. Typical examples for such sensors include thermometer, light sensors, vibration, microphones, humidity, mechanical stress or tension in materials, chemical sensors sensitive for given substances, smoke detectors, air pressure, and so on.

Passive, narrow-beam sensors These sensors are passive as well, but have a well-defined notion of direction of measurement. A typical example is a camera, which can "take measurements" in a given direction, but has to be rotated if need be.

Active sensors This last group of sensors actively probes the environment, for example, a sonar or radar sensor or some types of seismic sensors, which generate shock waves by small 5 See explosions. These are quite specific – triggering an explosion is certainly not a lightly undertaken action – and require quite special attention.

In practice, sensors from all of these types are available in many different forms with many individual peculiarities. Obvious trade-offs include accuracy, dependability, energy consumption, cost, size, and so on – all this would make a detailed discussion of individual sensors quite ineffective. Overall, most of the theoretical work on WSNs considers passive, omnidirectional sensors. Narrow-beam-type sensors like cameras are used in some practical testbeds, but there is no real systematic investigation on how to control and schedule the movement of such sensors. Active sensors are not treated in the literature to any noticeable extent.

An assumption occasionally made in the literature [128, 129] is that each sensor node has a certain **area of coverage** for which it can reliably and accurately report the particular quantity that it is observing. More elaborately, a sensor detection model is used, relating the distance between a sensor and the to-be-detected event or object to a detection probability; an example for such a detection model is contained in references [599, 944].

Strictly speaking, this assumption of a coverage area is difficult to justify in its simplest form. Nonetheless, it can be practically useful: It is often possible to postulate, on the basis of application specific knowledge, some properties of the physical quantity under consideration, in particular, how quickly it can change with respect to distance. For example, temperature or air pressure are unlikely to vary very strongly within a few meters. Hence, allowing for some inevitable inaccuracies in the measurement, the maximum rate of changeover distance can be used to derive such a "coverage radius" within which the values of a single sensor node are considered "good enough". The precise mathematical tools for such a derivation are spatial versions of the sampling theorems.

Actuators

Actuators are just about as diverse as sensors, yet for the purposes of designing a WSN, they are a bit simpler to take account of: In principle, all that a sensor node can do is to open or close a switch or a relay or to set a value in some way. Whether this controls a motor, a light bulb, or some other physical object is not really of concern to the way communication protocols are designed. Hence, in this book, we shall treat actuators fairly summarily without distinguishing between different types.

In a real network, however, care has to be taken to properly account for the idiosyncrasies of different actuators. Also, it is good design practice in most embedded system applications to pair any actuator with a controlling sensor – following the principle to "never trust an actuator" [429].

2.1.6 Power supply of sensor nodes

For untethered wireless sensor nodes, the power supply is a crucial system component. There are essentially two aspects: First, storing energy and providing power in the required form; second, attempting to replenish consumed energy by "scavenging" it from some node-external power source over time.

Storing power is conventionally done using batteries. As a rough orientation, a normal AA battery stores about 2.2–2.5 Ah at 1.5 V. Battery design is a science and industry in itself, and

energy scavenging has attracted a lot of attention in research. This section can only provide some small glimpses of this vast field; some papers that deal with these questions (and serve as the basis for this section) are references [134, 392, 667, 670] and, in particular, reference [703].

Storing energy: Batteries

Traditional batteries

The power source of a sensor node is a battery, either nonrechargeable ("primary batteries") or, if an energy scavenging device is present on the node, also rechargeable ("secondary batteries").

Capacity They should have high capacity at a small weight, small volume, and low price. The main metric is energy per volume, J/cm3. Table 2.2 shows some typical values of energy densities, using traditional, macroscale battery technologies. In addition, research on "microscale" batteries, for example, deposited directly onto a chip, is currently ongoing.

Capacity under load They should withstand various usage patterns as a sensor node can consume quite different levels of power over time and actually draw high current in certain operation modes.

Current numbers on power consumption of WSN nodes vary and are treated in detail in Section 2.2, so it is difficult to provide precise guidelines. But for most technologies, the larger the battery, the more power can be delivered instantaneously. In addition, the rated battery capacity specified by a manufacturer is only valid as long as maximum discharge currents are not exceeded, lest capacity drops or even premature battery failure occurs [670].

Self-discharge Their self-discharge should be low; they might also have to last for a long time (using certain technologies, batteries are operational only for a few months, irrespective of whether power is drawn from them or not). Zinc-air batteries, for example, have only a very short lifetime (on the order of weeks), which offsets their attractively high energy density.

Efficient recharging Recharging should be efficient even at low and intermittently available recharge power; consequently, the battery should also not exhibit any "memory effect". Some of the energy-scavenging techniques described below are only able to produce current in the μA region (but possibly sustained) at only a few volts at best. Current battery technology would basically not recharge at such values.

Relaxation Their relaxation effect – the seeming self-recharging of an empty or almost empty battery when no current is drawn from it, based on chemical diffusion processes within the cell – should be clearly understood. Battery lifetime and usable capacity is considerably extended if

this effect is leveraged. As but one example, it is possible to use multiple batteries in parallel and "schedule" the discharge from one battery to another, depending on relaxation properties and power requirements of the operations to be supported [153].

This effect is due to the need for active material in a battery to be transported to the electrodes. If too much power is drawn, this transport is not fast enough and the battery fails even though energy is still stored in it.

Unconventional energy stores

Apart from traditional batteries, there are also other forms of energy reservoirs that can be contemplated. In a wider sense, fuel cells also qualify as an electro-chemical storage of energy, directly producing electrical energy by oxidizing hydrogen or hydrocarbon fuels. Fuel cells actually have excellent energy densities (e.g. methanol as a fuel stores 17.6 kJ/cm3), but currently available systems still require a nonnegligible minimum size for pumps, valves, and so on. A slightly more traditional approach to using energy stored in hydrocarbons is to use miniature versions of heat engines, for example, a turbine [243]. Shrinking such heat engines to the desired sizes still requires a considerable research effort in MicroElectroMechanical Systems (MEMSs); predictions regarding power vary between 0.1–10 W at sizes of about 1 cc [703]. And lastly, even radioactive substances have been proposed as an energy store [463]. Another option are so-called "gold caps", high-quality and high-capacity capacitors, which can store relatively large amounts of energy, can be easily and quickly recharged, and do not wear out over time.

DC–DC Conversion

Unfortunately, batteries (or other forms of energy storage) alone are not sufficient as a direct power source for a sensor node. One typical problem is the reduction of a battery's voltage as its capacity drops. Consequently, less power is delivered to the sensor node's circuits, with immediate consequences for oscillator frequencies and transmission power – a node on a weak battery will have a smaller transmission range than one with a full battery, possibly throwing off any calibrations done for the range at full battery ranges.

A DC – DC converter can be used to overcome this problem by regulating the voltage delivered to the node's circuitry. To ensure a constant voltage even though the battery's supply voltage drops, the DC – DC converter has to draw increasingly higher current from the battery when the battery is already becoming weak, speeding up battery death (see Figure 3 in reference [670]). Also, the DC – DC converter does consume energy for its own operation, reducing overall

efficiency. But the advantages of predictable operation during the entire life cycle can outweigh these disadvantages.

Energy scavenging

Some of the unconventional energy stores described above – fuel cells, micro heat engines, radioactivity – convert energy from some stored, secondary form into electricity in a less direct and easy to use way than a normal battery would do. The entire energy supply is stored on the node itself – once the fuel supply is exhausted, the node fails.

To ensure truly long-lasting nodes and wireless sensor networks, such a limited energy store is unacceptable. Rather, energy from a node's environment must be tapped into and made available to the node – **energy scavenging** should take place. Several approaches exist [667, 701, 703]:

Photovoltaics The well-known solar cells can be used to power sensor nodes. The available power depends on whether nodes are used outdoors or indoors, and on time of day and whether for outdoor usage. Different technologies are best suited for either outdoor or indoor usage. The resulting power is somewhere between 10 μW/cm2 indoors and 15 mW/cm2 outdoors. Single cells achieve a fairly stable output voltage of about 0.6 V (and have therefore to be used in series) as long as the drawn current does not exceed a critical threshold, which depends, among other factors, on the light intensity. Hence, solar cells are usually used to recharge secondary batteries. Best trade-offs between complexity of recharging circuitry, solar cell efficiency, and battery lifetime are still open questions.

Temperature gradients Differences in temperature can be directly converted to electrical energy. Theoretically, even small difference of, for example, 5 K can produce considerable power, but practical devices fall very short of theoretical upper limits (given by the Carnot efficiency).

Seebeck effect-based thermoelectric generators are commonly considered; one example is a generator, which will be commercially available soon, that achieves about 80 μW/cm2 at about 1 V from a 5 Kelvin temperature difference.

Vibrations One almost pervasive form of mechanical energy is vibrations: walls or windows in buildings are resonating with cars or trucks passing in the streets, machinery often has lowfrequency vibrations, ventilations also cause it, and so on. The available energy depends on both amplitude and frequency of the vibration and ranges from about 0.1 μW/cm3 up to 10, 000 μW/cm3 for some extreme cases (typical upper limits are lower).

Converting vibrations to electrical energy can be undertaken by various means, based on electromagnetic, electrostatic, or piezoelectric principles. Figure 2.4 shows, as an example, a generator based on a variable capacitor [549]. Practical devices of 1 cm3 can produce about 200 μW/cm3 from 2.25 m/s2, 120 Hz vibration sources, actually sufficient to power simple wireless transmitters [702].

Pressure variations Somewhat akin to vibrations, a variation of pressure can also be used as a power source. Such piezoelectric generators are in fact used already. One well-known example is the inclusion of a piezoelectric generator in the heel of a shoe, to generate power as a human walks about [759]. This device can produce, on average, 330 μW/cm2. It is, however, not clear how such technologies can be applied to WSNs.

Energy consumption of sensor nodes

2.2.1 Operation states with different power consumption

As the previous section has shown, energy supply for a sensor node is at a premium: batteries have small capacity, and recharging by energy scavenging is complicated and volatile. Hence, the energy consumption of a sensor node must be tightly controlled. The main consumers of energy are the controller, the radio front ends, to some degree the memory, and, depending on the type, the sensors.

To give an example, consider the energy consumed by a microcontroller per instruction. A typical ball park number is about 1 nJ per instruction [391]. To put this into perspective with the battery capacity numbers from Section 2.1.6, assume a battery volume of one cubic millimeter, which is about the maximum possible for the most ambitious visions of “smart dust”. Such a battery could store about 1 J. To use such a battery to power a node even only a single day, the node must not consume continuously more than $1/(24 \cdot 60 \cdot 60)$ Ws/s ≈ 11.5 μW. No current controller, let alone an entire node, is able to work at such low-power levels.

One important contribution to reduce power consumption of these components comes from chip-level and lower technologies: Designing low-power chips is the best starting point for an energy-efficient sensor node. But this is only one half of the picture, as any advantages gained by such designs can easily be squandered when the components are improperly operated.

Microcontroller energy consumption

Basic power consumption in discrete operation states

Embedded controllers commonly implement the concept of multiple operational states as outlined above; it is also fairly easy to control. Some examples probably best explain the idea.

Intel StrongARM

The Intel StrongARM [379] provides three sleep modes:

• In *normal mode*, all parts of the processor are fully powered. Power consumption is up to 400 mW.

• In *idle mode*, clocks to the CPU are stopped; clocks that pertain to peripherals are active. Any interrupt will cause return to normal mode. Power consumption is up to 100 mW.

• In *sleep mode*, only the real-time clock remains active. Wakeup occurs after a timer interrupt and takes up to 160 ms. Power consumption is up to 50 µW.

Texas Instruments MSP 430

The MSP430 family [814] features a wider range of operation modes: One fully operational mode, which consumes about 1.2 mW (all power values given at 1 MHz and 3 V). There are four sleep modes in total. The deepest sleep mode, LPM4, only consumes 0.3 µW, but the controller is only woken up by external interrupts in this mode. In the next higher mode, LPM3, a clock is also still running, which can be used for scheduled wake ups, and still consumes only about 6 µW.

Atmel ATmega

The Atmel ATmega 128L [28] has six different modes of power consumption, which are in principle similar to the MSP 430 but differ in some details. Its power consumption varies between 6 mW and 15 mW in idle and active modes and is about 75 µW in power-down modes.

Case Study: TinyOS and nesC

Section 2.3.2 has advocated the use of an event-based programming model as the only feasible way to support the concurrency required for sensor node software while staying within the confined resources and running on top of the simple hardware provided by these nodes. The open question is how to harness the power of this programming model without getting lost in the complexity of many individual state machines sending each other events. In addition, modularity should be supported to easily exchange one state machine against another. The operating system

TinyOS [353], along with the programming language nesC [285], addresses these challenges (the exposition here follows mainly these references).

TinyOS supports modularity and event-based programming by the concept of components. A component contains semantically related functionality, for example, for handling a radio interface or for computing routes. Such a component comprises the required state information in a *frame*, the program code for normal *tasks*, and handlers for *events* and *commands*. Both events and commands are exchanged between different components. Components are arranged hierarchically, from low-level components close to the hardware to high-level components making up the actual application. Events originate in the hardware and pass upward from low-level to high-level components; commands, on the other hand, are passed from high-level to low-level components. Figure 2.9 shows a timer component that provides a more abstract version of a simple hardware time. It understands three commands ("init", "start", and "stop") and can handle one event ("fire") from another component, for example, a wrapper component around a hardware timer. It issues "setRate" commands to this component and can emit a "fired" event itself. The important thing to note is that, in staying with the event-based paradigm, both command and event handlers must run to conclusion; they are only supposed to perform very simple triggering duties. In particular, commands must not block or wait for an indeterminate amount of time; they are simply a request upon which some task of the hierarchically lower component has to act. Similarly, an event handler only leaves information in its component's frame and arranges for a task to be executed later; it can also send commands to other components or directly report an event further up.

The actual computational work is done in the tasks. In TinyOS, they also have to run to completion, but can be interrupted by handlers. The advantage is twofold: there is no need for stack management and tasks are atomic with respect to each other. Still, by virtue of being triggered by handlers, tasks are seemingly concurrent to each other.

The arbitration between tasks – multiple can be triggered by several events and are ready to execute – is done by a simple, power-aware First In First Out (FIFO) scheduler, which shuts the node down when there is no task executing or waiting.

Network architecture:

Sensor network scenarios

3.1.1 Types of sources and sinks

Section 1.3 has introduced several typical interaction patterns found in WSNs – event detection, periodic measurements, function approximation and edge detection, or tracking – it has also already briefly touched upon the definition of "sources" and "sinks". A source is any entity in the network that can provide information, that is, typically a sensor node; it could also be an actuator node that provides feedback about an operation.

A sink, on the other hand, is the entity where information is required. There are essentially three options for a sink: it could belong to the sensor network as such and be just another sensor/actuator node or it could be an entity outside this network. For this second case, the sink could be an actual device, for example, a handheld or PDA used to interact with the sensor network; it could also be merely a gateway to another larger network such as the Internet, where the actual request for the information comes from some node "far away" and only indirectly connected to such a sensor network. These main types of sinks are illustrated by Figure 3.1, showing sources and sinks in direct communication.

For much of the remaining discussion, this distinction between various types of sinks is actually fairly irrelevant. It is important, as discussed in Section 3.1.4, whether sources or sinks move, but what they do with the information is not a primary concern of the networking architecture. There are some consequences of a sink being a gateway node; they will be discussed in Section 3.5.

3.1.2 Single-hop versus multihop networks

From the basics of radio communication and the inherent power limitation of radio communication follows a limitation on the feasible distance between a sender and a receiver. Because of this

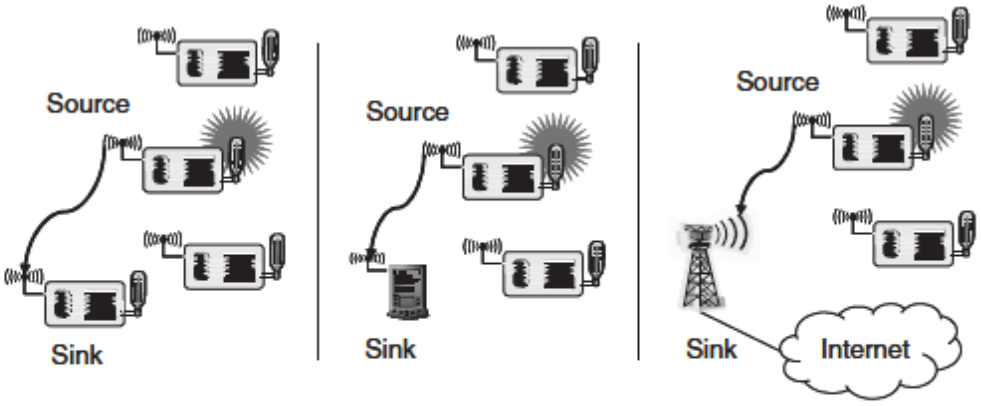


Figure 3.1 Three types of sinks in a very simple, single-hop sensor network

limited distance, the simple, direct communication between source and sink is not always possible, specifically in WSNs, which are intended to cover a lot of ground (e.g. in environmental or agriculture applications) or that operate in difficult radio environments with strong attenuation (e.g. in buildings).

To overcome such limited distances, an obvious way out is to use relay stations, with the data packets taking multi hops from the source to the sink. This concept of multihop networks (illustrated in Figure 3.2) is particularly attractive for WSNs as the sensor nodes themselves can act as such relay nodes, foregoing the need for additional equipment. Depending on the particular application, the likelihood of having an intermediate sensor node at the right place can actually be quite high – for example, when a given area has to be uniformly equipped with sensor nodes anyway – but nevertheless, there is not always a guarantee that such multihop routes from source to sink exist, nor that such a route is particularly short. While multihopping is an evident and working solution to overcome problems with large distances or obstacles, it has also been claimed to improve the energy efficiency of communication.

The intuition behind this claim is that, as attenuation of radio signals is at least quadratic in most environments (and usually larger), it consumes less energy to use relays instead of direct communication: When targeting for a constant SNR at all receivers (assuming for simplicity negligible error rates at this SNR), the *radiated* energy required for direct communication over a distance d is $cd\alpha$ (c some constant, $\alpha \geq 2$ the path loss coefficient); using a relay at distance $d/2$ reduces this energy to $2c(d/2)\alpha$.

But this calculation considers only the radiated energy, not the actually *consumed* energy – in

particular, the energy consumed in the intermediate relay node. Even assuming that this relay belongs to the WSN and is willing to cooperate, when computing the total required energy it is necessary to take into account the complete power consumption of Section 2.2.4. It is an easy exercise to show that energy is actually wasted if intermediate relays are used for short distances *d*. Only for large *d* does the radiated energy dominate the fixed energy costs consumed in transmitter and receiver electronics – the concrete distance where direct and multihop communication are in balance depends on a lot of device-specific and environment-specific parameters. Nonetheless, this relationship is often not considered. In fact, Min and Chandrakasan [560] classify the misconception that multihopping saves energy as the number one myth about energy consumption in wireless communication. Great care should be taken when applying multihopping with the end of improved energy efficiency.

It should be pointed out that only multihop networks operating in a **store and forward** fashion are considered here. In such a network, a node has to correctly receive a packet before it can forward it somewhere. Alternative, innovative approaches attempt to exploit even erroneous reception of packets, for example, when multiple nodes send the same packet and each individual transmission could not be received, but collectively, a node can reconstruct the full packet. Such **cooperative relaying** techniques are not considered here.

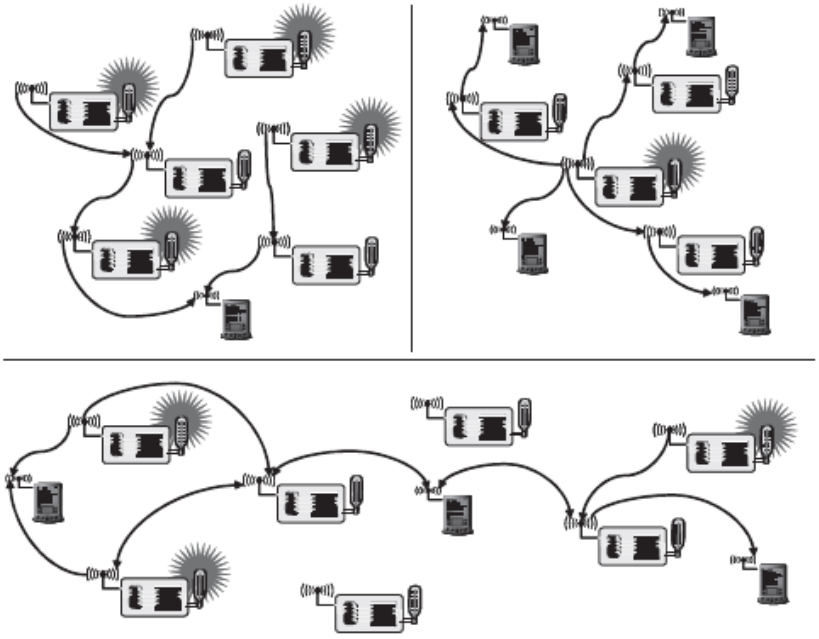


Figure 3.3 Multiple sources and/or multiple sinks. Note how in the scenario in the lower half, both sinks and active sources are used to forward data to the sinks at the left and right end of the network

Multiple sinks and sources

So far, only networks with a single source and a single sink have been illustrated. In many cases, there are multiple sources and/or multiple sinks present. In the most challenging case, multiple sources should send information to multiple sinks, where either all or some of the information has to reach all or some of the sinks. Figure 3.3 illustrates these combinations.

3.1.4 Three types of mobility

In the scenarios discussed above, all participants were stationary. But one of the main virtues of wireless communication is its ability to support mobile participants. In wireless sensor networks, mobility can appear in three main forms:

Node mobility The wireless sensor nodes themselves can be mobile. The meaning of such mobility is highly application dependent. In examples like environmental control, node mobility should not happen; in livestock surveillance (sensor nodes attached to cattle, for example), it is the common rule.

In the face of node mobility, the network has to reorganize itself frequently enough to be able to function correctly. It is clear that there are trade-offs between the frequency and speed of node movement on the one hand and the energy required to maintain a desired level of functionality in the network on the other hand.

Sink mobility The information sinks can be mobile (Figure 3.4). While this can be a special case of node mobility, the important aspect is the mobility of an information sink that is not part of the sensor network, for example, a human user requested information via a PDA while walking in an intelligent building.

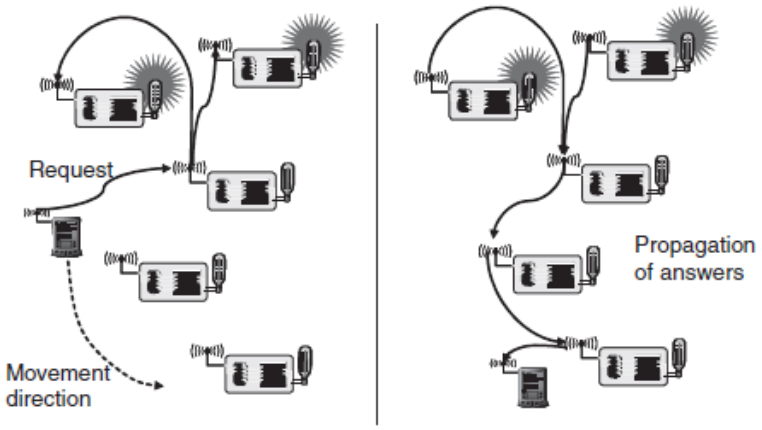


Figure 3.4 A mobile sink moves through a sensor network as information is being retrieved on its behalf

Event mobility In applications like event detection and in particular in tracking applications, the cause of the events or the objects to be tracked can be mobile.

Optimization goals and figures of merit

Quality of service

WSNs differ from other conventional communication networks mainly in the type of service they offer. These networks essentially only move bits from one place to another. Possibly, additional requirements about the offered Quality of Service (QoS) are made, especially in the context of multimedia applications. Such QoS can be regarded as a low-level, networking-device-observable attribute – bandwidth, delay, jitter, packet loss rate – or as a high-level, user-observable, so-called subjective attribute like the perceived quality of a voice communication or a video transmission. While the first kind of attributes is applicable to a certain degree to WSNs as well (bandwidth, for example, is quite unimportant), the second one clearly is not, but is really the more important one to consider! Hence, high-level QoS attributes corresponding to the subjective QoS attributes in conventional networks are required. But just like in traditional networks, high-level QoS attributes in WSN highly depend on the application. Some generic possibilities are:

Event detection/reporting probability What is the probability that an event that actually occurred is not detected or, more precisely, not reported to an information sink that is interested in such an event? For example, not reporting a fire alarm to a surveillance station would be a

severe shortcoming. Clearly, this probability can depend on/be traded off against the overhead spent in setting up structures in the network that support the reporting of such an event (e.g. routing tables) or against the run-time overhead (e.g. sampling frequencies).

Event classification error If events are not only to be detected but also to be classified, the error in classification must be small.

Event detection delay What is the delay between detecting an event and reporting it to any/all interested sinks?

Missing reports In applications that require periodic reporting, the probability of undelivered reports should be small.

Unit -2

Physical layer

Introduction

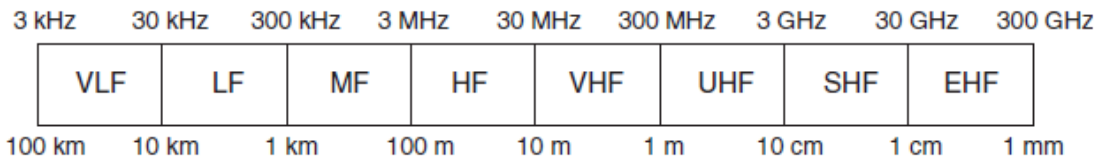
The physical layer is mostly concerned with modulation and demodulation of digital data; this task is carried out by so-called **transceivers**. In sensor networks, the challenge is to find modulation schemes and transceiver architectures that are simple, low cost, but still robust enough to provide the desired service.

The first part of this chapter explains the most important concepts regarding wireless channels and digital communications (over wireless channels); its main purpose is to provide appropriate notions and to give an insight into the tasks involved in transmission and reception over wireless channels. We discuss some simple modulation schemes as well.

Wireless channel and communication fundamentals

Frequency allocation

For a practical wireless, RF-based system, the carrier frequency has to be carefully chosen. This carrier frequency determines the propagation characteristics – for example, how well are obstacles like walls penetrated – and the available capacity. Since a single frequency does not provide any capacity, for communication purposes always a finite portion of the electromagnetic spectrum, called a **frequency band**, is used. In radio-frequency (RF) communications, the range of usable radio frequencies in general starts at the Very Low Frequency (VLF) range and ends with the Extremely High Frequency (EHF) range (Figure 4.1). There is also the option of **infrared** or **optical** communications, used, for example, in the "Smart Dust" system [392]. The infrared spectrum is between wavelengths of 1 mm (corresponding to 300 GHz1) and 2.5 μm (120 THz), whereas the optical range ends at 780 nm (≈385 THz).



VLF = Very low frequency
LF = Low frequency
MF = Medium frequency
HF = High frequency
VHF = Very high frequency
UHF = Ultrahigh frequency
SHF = Super high frequency
EHF = Extremely high frequency

Figure 4.1 Electromagnetic spectrum – radio frequencies

The choice of a frequency band is an important factor in system design. Except for ultrawideband technologies (see Section 2.1.4), most of today's RF-based systems work at frequencies below 6 GHz. The range of radio frequencies is subject to **regulation** to avoid unwanted interference between different users and systems. Some systems have special licenses for reserved bands; for example, in Europe, the GSM system can exclusively use the GSM 900 (880–915 MHz) and GSM 1800 (1710–1785 MHz) bands.2 There are also licensefree bands, most notably the Industrial, Scientific, and Medical (ISM) bands, which are granted by the ITU for private and unlicensed use subject to certain restrictions regarding transmit power, power spectral density, or duty cycle Table 4.1 lists some of the ISM frequency bands. Working in an unlicensed band means that one can just go to a shop, buy equipment, and start to transmit data without requiring any permission from the government/frequency allocation body. It is not surprising that these bands are rather popular, not only for sensor networks but also for/in other wireless technologies. For example, the 2.4-GHz ISM band is used for IEEE 802.11, Bluetooth, and IEEE 802.15.4.

Some considerations in the choice of frequency are the following:

• In the public ISM bands, any system has to live with interference created by other systems (using the same or different technologies) in the same frequency band, simply because there is no usage restriction. For example, many systems share the 2.4-GHz ISM band, including IEEE 802.11b [466, 467], Bluetooth [318, 319], and the IEEE 802.15.4 WPAN [468] – they **coexist** with each other in the same band. Therefore, all systems in these bands have to be robust against interference from other systems with which they cannot explicitly coordinate their operation.

Coexistence needs to be approached both on the physical and the MAC layer [154, 359, 360, 469]. On the other hand, requesting allocation of some exclusive spectrum for a specific sensor network application from the competent regulatory organizations is a time consuming and likely futile endeavor.

• An important parameter in a transmission system is the **antenna efficiency**, which is defined as the ratio of the **radiated power** to the total input power to the antenna; the remaining power is dissipated as heat. The small form factor of wireless sensor nodes allows only small antennas. For example, radio waves at 2.4 GHz have a wave length of 12.5 cm, much longer than the intended dimensions of many sensor nodes. In general, it becomes more difficult to construct efficient antennas as the ratio of antenna dimension to wavelength decreases. As the efficiency decreases, more energy must be spent to achieve a fixed radiated power. These problems are discussed in some detail in reference [115, Chap. 8].

4.2.2 Modulation and demodulation

When digital computers communicate, they exchange **digital data**, which are essentially sequences of **symbols**, each symbol coming from a finite alphabet, the **channel alphabet**. In the process of **modulation**, (groups of) symbols from the channel alphabet are mapped to one of a finite number of **waveforms** of the same finite length; this length is called the **symbol duration**. With two different waveforms, a **binary modulation** results; if the size is $m \in \mathrm{N}, m > 2$, we talk about m-ary modulation. Some common cases for the symbol alphabet are binary data (the alphabet being $\{0, 1\}$) or bipolar data ($\{-1, 1\}$) in spread-spectrum systems.

When referring to the "speed" of data transmission/modulation, we have to distinguish between the following parameters:

Symbol rate The **symbol rate** is the inverse of the symbol duration; for binary modulation, it is also called **bit rate**.

Data rate The **data rate** is the rate in bit per second that the modulator can accept for transmission; it is thus the rate by which a user can transmit binary data. For binary modulation, bit rate and data rate are the same and often the term bit rate is (sloppily) used to denote the data rate.

Physical layer and transceiver design considerations in WSNs

So far, we have discussed the basics of the PHY without specific reference to wireless sensor networks. Some of the most crucial points influencing PHY design in wireless sensor networks are:

• Low power consumption.

• As one consequence: small transmit power and thus a small transmission range.

• As a further consequence: low duty cycle. Most hardware should be switched off or operated in a low-power standby mode most of the time.

• Comparably low data rates, on the order of tens to hundreds kilobits per second, required.

• Low implementation complexity and costs.

• Low degree of mobility.

• A small form factor for the overall node.

In this section, we discuss some of the implications of these requirements.

In general, in sensor networks, the challenge is to find modulation schemes and transceiver architectures that are simple, low-cost but still robust enough to provide the desired service.

4.3.1 Energy usage profile

The choice of a small transmit power leads to an energy consumption profile different from other wireless devices like cell phones. These pivotal differences have been discussed in various places already but deserve a brief summary here. First, the radiated energy is small, typically on the order of 0 dBm (corresponding to 1mW). On the other hand, the overall transceiver (RF front end and baseband part) consumes much more energy than is actually radiated; Wang et al. [855] estimate that a transceiver working at frequencies beyond 1 GHz takes 10 to 100mW of power to radiate 1 mW. In reference [115, Chap. 3], similar numbers are given for 2.4-GHz CMOS transceivers: For a radiated power of 0 dBm, the transmitter uses actually 32 mW, whereas the receiver uses even more, 38 mW. For the Mica motes, 21 mW are consumed in transmit mode and 15 mW in receive mode [351]. These numbers coincide well with the observation that many practical transmitter designs have efficiencies below 10% [46] at low radiated power.

A second key observation is that for small transmit powers the transmit and receive modes consume more or less the same power; it is even possible that reception requires more power than transmission [670, 762]; depending on the transceiver architecture, the idle mode's power

consumption can be less or in the same range as the receive power [670]. To reduce average power consumption in a low-traffic wireless sensor network, keeping the transceiver in idle mode all the time would consume significant amounts of energy. Therefore, it is important to put the transceiver into sleep state instead of just idling. It is also important to explicitly include the received power into energy dissipation models, since the traditional assumption that receive energy is negligible is no longer true.

However, there is the problem of the **startup energy**/**startup time**, which a transceiver has to spend upon waking up from sleep mode, for example, to ramp up phase-locked loops or voltagecontrolled oscillators. During this startup time, no transmission or reception of data is possible [762]. For example, the μAMPS-1 transceiver needs a startup time of 466 μs and a power dissipation of 58 mW [561, 563]. Therefore, going into sleep mode is unfavorable when the next wakeup comes fast. It depends on the traffic patterns and the behavior of the MAC protocol to schedule the transceiver operational state properly. If possible, not only a single but multiple packets should be sent during a wakeup period, to distribute the startup costs over more packets. Clearly, one can attack this problem also by devising transmitter architectures with faster startup times. One such architecture is presented in reference [855].

A third key observation is the relative costs of communications versus computation in a sensor node. Clearly, a comparison of these costs depends for the communication part on the BER requirements, range, transceiver type, and so forth, and for the computation part on the processor type, the instruction mix, and so on. However, in [670], a range of energy consumptions is given for Rockwell's WIN nodes, UCLA's WINS NG 2.0 nodes, and the MEDUSA II nodes. For the WIN nodes, 1500 to 2700 instructions can be executed per transmitted bit, for the MEDUSA II nodes this ratio ranges from 220:1 up to 2900:1, and for the WINS NG nodes, it is around 1400:1. The bottom line is that computation is cheaper than communication!

4.3.2 Choice of modulation scheme

A crucial point is the choice of modulation scheme. Several factors have to be balanced here: the required and desirable data rate and symbol rate, the implementation complexity, the relationship between radiated power and target BER, and the expected channel characteristics. To maximize the time a transceiver can spend in sleep mode, the transmit times should be minimized. The higher the data rate offered by a transceiver/modulation, the smaller the time needed to transmit a given amount of data and, consequently, the smaller the energy consumption. A second

important observation is that the power consumption of a modulation scheme depends much more on the symbol rate than on the data rate [115, Chap. 3]. For example, power consumption measurements of an IEEE 802.11b Wireless Local Area Network (WLAN) card showed that the power consumption depends on the modulation scheme, with the faster Complementary Code Keying (CCK) modes consuming more energy than DBPSK and DQPSK; however, the relative differences are below 10% and all these schemes have the same symbol rate. It has also been found that for the μAMPS-1 nodes the power consumption is insensitive to the data rate [762]. Obviously, the desire for "high" data rates at "low" symbol rates calls for *m*-ary modulation schemes. However, there are trade-offs:

• *m*-ary modulation requires more complex digital and analog circuitry than 2-ary modulation [762], for example, to parallelize user bits into *m*-ary symbols.

Dynamic modulation scaling

Even if it is possible to determine the optimal scheme for a given combination of BER target, range, packet sizes and so forth, such an optimum is only valid for short time; as soon as one of the constraints changes, the optimum can change, too. In addition, other constraints like delay or the desire to achieve high throughput can dictate to choose higher modulation schemes. Therefore, it is interesting to consider methods to *adapt* the modulation scheme to the current situation. Such an approach, called **dynamic modulation scaling**, is discussed in reference .
In particular, for the case of *m*-ary QAM and a target BER of 10^{-5}, a model has been developed that uses the symbol rate *B* and the number of levels per symbol *m* as parameters. This model expresses the energy required per bit and also the achieved delay per bit (the inverse of the data rate), taking into account that higher modulation levels need higher radiated energy. Extra startup costs are not considered. Clearly, the bit delay decreases for increasing *B* and *m*. The energy per bit depends much more on *m* than on *B*. In fact, for the particular parameters chosen, it is shown that both energy per bit and delay per bit are minimized for the maximum symbol rate. With modulation scaling, a packet is equipped with a delay constraint, from which directly a minimal required data rate can be derived. Since the symbol rate is kept fixed, the approach is to choose the smallest *m* that satisfies the required data rate and which thus minimizes the required energy per bit. Such delay constraints can be assigned either explicitly or implicitly. One approach explored in the paper is to make the delay constraint depend on the packet backlog (number of

queued packets) in a sensor node: When there are no packets present, a small value for *m* can be used, having low energy consumption. As backlog increases, *m* is increased as well to reduce the backlog quickly and switch back to lower values of *m*. This modulation scaling approach has some similarities to the concept of **dynamic voltage scaling** discussed in Section 2.2.2.

4.3.4 Antenna considerations

The desired small form factor of the overall sensor nodes restricts the size and the number of antennas. As explained above, if the antenna is much smaller than the carrier's wavelength, it is hard to achieve good antenna efficiency, that is, with ill-sized antennas one must spend more transmit energy to obtain the same radiated energy. Secondly, with small sensor node cases, it will be hard to place two antennas with suitable distance to achieve receive diversity. As discussed in Section 4.2.7, the antennas should be spaced apart at least 40–50% of the wavelength used to achieve good effects from diversity. For 2.4 GHz, this corresponds to a spacing of between 5 and 6 cm between the antennas, which is hard to achieve with smaller cases. In addition, radio waves emitted from an antenna close to the ground – typical in some applications – are faced with higher path-loss coefficients than the common value $\alpha = 2$ for free-space communication. Typical attenuation values in such environments, which are also normally characterized by obstacles (buildings, walls, and so forth), are about $\alpha = 4$ [245, 648].

Moreover, depending on the application, antennas must not protrude from the casing of a node, to avoid possible damage to it. These restrictions, in general, limit the achievable quality and characteristics of an antenna for wireless sensor nodes. Nodes randomly scattered on the ground, for example, deployed from an aircraft, will land in random orientations, with the antennas facing the ground or being otherwise obstructed. This can lead to nonisotropic propagation of the radio wave, with considerable differences in the strength of the emitted signal in different directions.

WIRELESS LANS AND PANS

Network Architecture WLAN

This section lists the types of WLANs, the components of a typical WLAN, and the services offered by a WLAN. Infrastructure Based Versus Ad Hoc LANs WLANs can be broadly classified into two types, infrastructure networks and ad hoc LANs, based on the underlying architecture. Infrastructure networks contain special nodes called *access points* (APs), which are connected via existing networks. APs are special in the sense that they can interact with wireless nodes as well as with the existing wired network. The other wireless nodes, also known as mobile stations (STAs), communicate via APs. The APs also act as bridges with other networks. Ad hoc LANs do not need any fixed infrastructure. These networks can be set up on the fly at any place. Nodes communicate directly with each other or forward messages through other nodes that are directly accessible. Components in a Typical IEEE 802.11 Network IEEE 802.11 is the most popular WLAN standard that defines the specification for the physical andMAC layers. The success of this standard can be understood from the fact that the revenue from the products based on this standard touched $730 million in the second quarter of the year 2003. The principles and mechanisms followed in this standard are explained later. In what follows, the basic components in a typical IEEE 802.11 WLAN [2] are listed. The set of stations that can remain in contact (*i.e.*, are associated) with a given AP is called a basic service set (BSS). The coverage area of an AP within which member stations (STAs or MTs) may remain in communication is called the basic service area (BSA). The stations that are a part of a BSSneed to be located within the BSA of the corresponding AP. A BSS is the basic building block of the network. BSSs are connected by means of a distribution system (DS) to form an extended network.

DS refers to an existing network infrastructure. The implementation of the DS is not specified by theIEEE 802.11 standard. The services of the DS, however, are specified rigidly. This gives a lot of flexibility in the design of the DS. The APs are connected by means of the DS.

Portals are logical points through which non-IEEE 802.11 packets (wired LAN packets) enter the system. They are necessary for integrating wireless networks with the existing wired networks. Just as an AP interacts with the DS as well as the wireless nodes, the portal interacts with the wired network as well as with the DS. The BSSs, DS, and the portals together with the stations

they connect constitute the extended service set (ESS). An ad hoc LAN has only one BSS. Therefore, ad hoc LANs are also known as independent basic service sets (IBSSs). It may be noted that the ESS and IBSS appear identical to the logical link control (LLC). Figure 2.1 gives a schematic picture of what a typical ESSlooks like.

MANET

Simply stating, a Mobile Ad hoc NETwork (MANET) is one that comes together as needed, not necessarily with any support from the existing Internet infrastructure or any other kind of fixed stations. We can formalize this statement by defining an ad hoc network as an autonomous system of mobile hosts (also serving as routers) connected by wireless links, the union of which forms a communication network modeled in the form of an arbitrary graph. This is in contrast to the well-known single hop cellular network model that supports the needs of wireless communication by installing base stations as access points. In these cellular networks, communications between two mobile nodes completely rely on the wired backbone and the fixed base stations. In a MANET, no such infrastructure exists and the network topology may dynamically change in an unpredictable manner since nodes are free to move.

As for the mode of operation, ad hoc networks are basically peer-to-peer multi-hop mobile wireless networks where information packets are transmitted in a store-andforward manner from a source to an arbitrary destination, via intermediate nodes as shown in Figure 1. As the nodes move, the resulting change in network topology must be made known to the other nodes so that outdated topology information can be updated or removed. For example, as MH2 in Figure 1 changes its point of attachment from MH3 to MH4 other nodes part of the network should use this new route to forward packets to MH2.

Note that in Figure 1, and throughout this text, we assume that it is not possible to have all nodes within range of each other. In case all nodes are close-by within radio range, there are no routing issues to be addressed. In real situations, the power needed to obtain complete connectivity may be, at least, infeasible, not to mention issues such as battery life. Therefore, we are interested in scenarios where only few nodes are within radio range of each other.

Figure 1 raises another issue of symmetric (bi-directional) and asymmetric (unidirectional) links. As we shall see later on, some of the protocols we discuss consider symmetric links with associative radio range, i.e., if (in Figure 1) MH1 is within radio range of MH3, then MH3 is also

within radio range of MH1. This is to say that the communication links are symmetric. Although this assumption is not always valid, it is usually made because routing in asymmetric networks is a relatively hard task. In certain cases, it is possible to find routes that could avoid asymmetric links, since it is quite likely that these links imminently fail. Unless stated otherwise, throughout this text we consider symmetric links, with all nodes having identical capabilities and responsibilities.

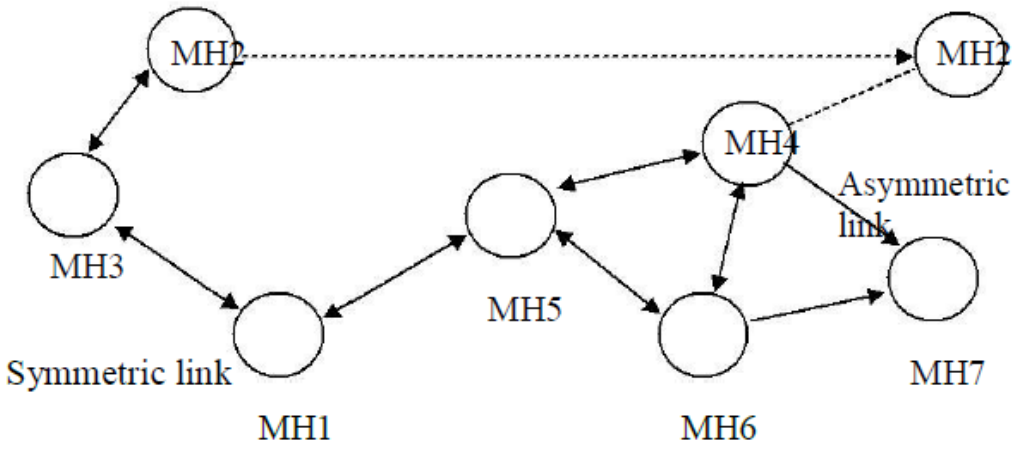


Figure 1 – A Mobile Ad hoc network

The issue of symmetric and asymmetric links is one among the several challenges encountered in a MANET. Another important issue is that different nodes often have different mobility patterns. Some nodes are highly mobile, while others are primarily stationary. It is difficult to predict a node's movement and pattern of movement. Table 1 summarizes some of the main characteristics [Duggirala 2000] and challenges faced in a MANET.

Wireless Sensor Networks [Estrin 1999, Kahn 1999] is an emerging application area for ad hoc networks which has been receiving a large attention. The idea is that a collection of cheap to manufacture, stationary, tiny sensors would be able to sense, coordinate activities and transmit some physical characteristics about the surrounding environment to an associated base station. Once placed in a given environment, these sensors remain stationary. Furthermore, it is expected that power will be a major driving issue behind protocols tailored to these networks, since the lifetime of the battery usually defines the sensor's lifetime. One of the most cited examples is the battlefield surveillance of enemy's territory wherein a large number of sensors are dropped from

an airplane so that activities on the ground could be detected and communicated. Other potential commercial fields include machinery prognosis, bio sensing and environmental monitoring.

This rest of this text is organized as follows. We initially provide necessary background on ad hoc networking by illustrating its diverse applications. Next, we cover the routing aspect in a MANET, considering both unicast and multicast communication. MAC issues related to a MANET are then illustrated. Following, sensor networks, its diverse applications, and associated routing protocols are discussed. Finally, we conclude this text by discussing the current standard activities at both IETF and the Bluetooth SIG, and also bringing up some open problems that have not received much attention so far and still need to be addressed.

Applications of MANETs

There are many applications to ad hoc networks. As a matter of fact, any day-today application such as electronic email and file transfer can be considered to be easily deployable within an ad hoc network environment. Web services are also possible in case any node in the network can serve as a gateway to the outside world. In this discussion, we need not emphasize the wide range of military applications possible with ad hoc networks. Not to mention, the technology was initially developed keeping in mind the military applications, such as battlefield in an unknown territory where an infrastructured network is almost impossible to have or maintain. In such situations, the ad hoc networks having self-organizing capability can be effectively used where other technologies either fail or cannot be deployed effectively. Advanced features of wireless mobile systems, including data rates compatible with multimedia applications, global roaming capability, and coordination with other network structures, are enabling new applications. Some well-known ad hoc network applications are:

• Collaborative Work – For some business environments, the need for collaborative computing might be more important outside office environments than inside. After all, it is often the case where people do need to have outside meetings to cooperate and exchange information on a given project.

• Crisis-management Applications – These arise, for example, as a result of natural disasters where the entire communications infrastructure is in disarray. Restoring communications quickly is essential. By using ad hoc networks, an infrastructure could be set up in hours instead of days/weeks required for wire-line communications.

• Personal Area Networking and Bluetooth – A personal area network (PAN) is a short-range, localized network where nodes are usually associated with a given person. These nodes could be attached to someone's pulse watch, belt, and so on.

In these scenarios, mobility is only a major consideration when interaction among several PANs is necessary, illustrating the case where, for instance, people meet in real life. Bluetooth [Haarsten 1998], is a technology aimed at, among other things, supporting PANs by eliminating the need of wires between devices such as printers, PDAs, notebook computers, digital cameras, and so on, and is discussed later.

UNIT -3

REF TEXTBOOK-AD HOC WIRELESS NETWORK

3.1.ISSUES IN DESIGNING A MAC PROTOCOL FOR AD HOC WIRELESS NETWORKS

The following are the main issues that need to be addressed while designing a MAC protocol for ad hoc wireless networks.

1 Bandwidth Efficiency

As mentioned earlier, since the radio spectrum is limited, the bandwidth available for communication is also very limited. The MAC protocol must be designed in such a way that the scarce bandwidth is utilized in an efficient manner. The control overhead involved must be kept as minimal as possible. Bandwidth efficiency can be defined as the ratio of the bandwidth used for actual data transmission to the total available bandwidth. The MACprotocol must try to maximize this bandwidth efficiency.

2 Quality of Service Support

Due to the inherent nature of the ad hoc wireless network, where nodes are usually mobile most of the time, providing quality of service (QoS) support to data sessions in such networks is very difficult. Bandwidth reservation made at one point of time may become invalid once the node moves out of the region where the reservation was made. QoS support is essential for supporting time-critical traffic sessions such as in military communications. The MACprotocol for ad hoc wireless networks that are to be used in such real-time applications must have some kind of a resource reservation mechanism that takes into consideration the nature of the wireless channel and the mobility of nodes

3 Synchronization

The MAC protocol must take into consideration the synchronization between nodes in the network. Synchronization is very important for bandwidth (time slot) reservations by nodes. Exchange of control packets may be required for achieving time synchronization among nodes.

The control packets must not consume too much of network bandwidth

4 Hidden and Exposed Terminal Problems

The hidden and exposed terminal problems are unique to wireless networks. The hidden terminal problem refers to the collision of packets at a receiving node due to the simultaneous transmission of those nodes that are not within the direct transmission range of the sender, but are within the transmission range of the receiver. Collision occurs when both nodes transmitpackets at the same time without knowing about the transmission of each other. For example,consider Figure 6.1. Here, if both node S1 and node S2 transmit to node R1 at the same time,their packets collide at node R1. This is because both nodes S1 and S2 are hidden from eachother as they are not within the direct transmission range of each other and hence do not know about the presence of each other.

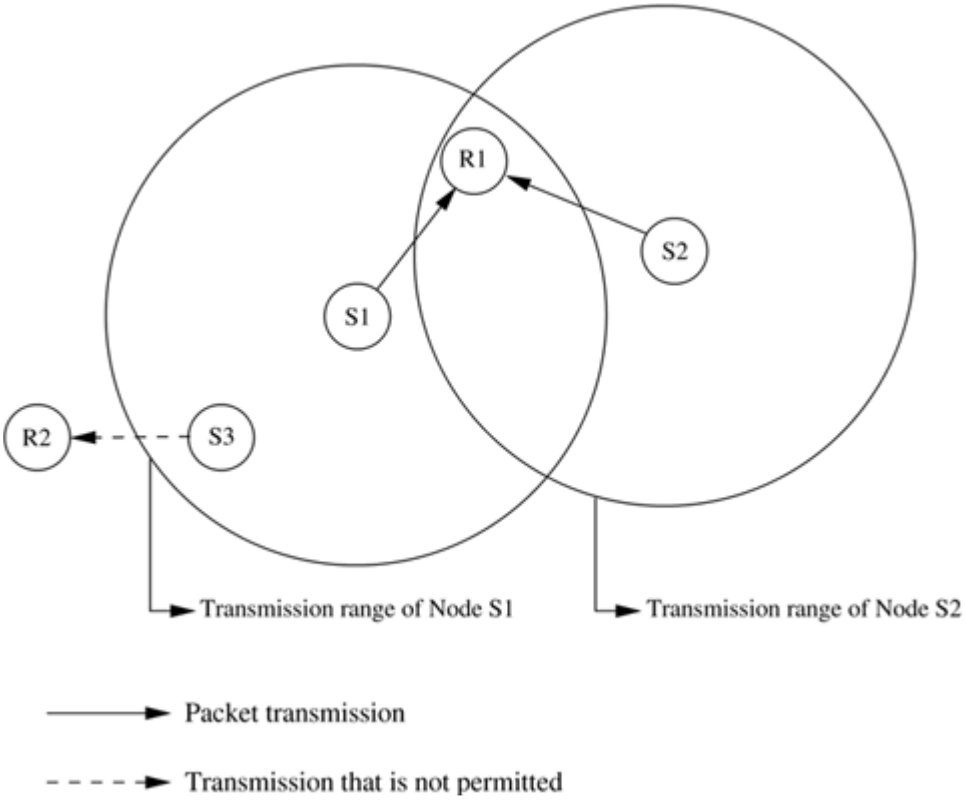


FIGURE:HIDDEN AND EXPOSED PROBLEM

The exposed terminal problem refers to the inability of a node, which is blocked due to transmission by a nearby transmitting node, to transmit to another node. Consider the example in Figure 6.1. Here, if a transmission from node S1 to another node R1 is already in progress,

node S3 cannot transmit to node R2, as it concludes that its neighbor node S1 is in transmitting mode and hence it should not interfere with the on-going transmission.
The hidden and exposed terminal problems significantly reduce the throughput of a network when the traffic load is high. It is therefore desirable that the MAC protocol be free from the hidden and exposed terminal problems

5 Error-Prone Shared Broadcast Channel

Another important factor in the design of a MAC protocol is the broadcast nature of the radio channel, that is, transmissions made by a node are received by all nodes within its direct transmission range. When a node is receiving data, no other node in its neighborhood, apart from the sender, should transmit. A node should get access to the shared medium only when its transmissions do not affect any ongoing session. Since multiple nodes may contend for the channel simultaneously, the possibility of packet collisions is quite high in wireless networks. A MAC protocol should grant channel access to nodes in such a manner that collisions are minimized. Also, the protocol should ensure that all nodes are treated fairly with respect to bandwidth allocation.

6 Distributed Nature/Lack of Central Coordination

Ad hoc wireless networks do not have centralized coordinators. In cellular networks, for example, the base stations act as central coordinating nodes and allocate bandwidth to the mobile terminals. But this is not possible in an ad hoc network, where nodes keep moving continuously. Therefore, nodes must be scheduled in a distributed fashion for gaining access to the channel. This may require exchange of control information. The MAC protocol must make sure that the additional overhead, in terms of bandwidth consumption, incurred due to this control information exchange is not very high.

7 Mobility of Nodes

This is a very important factor affecting the performance (throughput) of the protocol. Nodes in an ad hoc wireless network are mobile most of the time. The bandwidth reservations made or the control information exchanged may end up being of no use if the node mobility is very high. The MAC protocol obviously has no role to play in influencing the mobility of the nodes. The protocol design must take this mobility factor into consideration so that the performance of the system is not significantly affected due to node mobility.

3.2 DESIGN GOALS OF A MAC PROTOCOL FOR AD HOC WIRELESS NETWORKS

The following are the important goals to be met while designing a medium access control (MAC) protocol for ad hoc wireless networks:

• The operation of the protocol should be distributed.

• The protocol should provide QoS support for real-time traffic.

• The access delay, which refers to the average delay experienced by any packet to get transmitted, must be kept low.

• The available bandwidth must be utilized efficiently.

• The protocol should ensure fair allocation (either equal allocation or weighted allocation) of bandwidth to nodes.

• Control overhead must be kept as low as possible.

• The protocol should minimize the effects of hidden and exposed terminal problems.

• The protocol must be scalable to large networks.

• It should have power control mechanisms in order to efficiently manage energy consumption of the nodes.

• The protocol should have mechanisms for adaptive data rate control (adaptive rate control refers to the ability to control the rate of outgoing traffic from a node after taking into consideration such factors as load in the network and the status of neighbor nodes).

• It should try to use directional antennas which can provide advantages such as reduced interference, increased spectrum reuse, and reduced power consumption.

• Since synchronization among nodes is very important for bandwidth reservations, the protocol should provide time synchronization among nodes.

3.3 CLASSIFICATIONS OF MAC PROTOCOLS

Ad hoc network MAC protocols can be classified into three basic types:

• Contention-based protocols

• Contention-based protocols with reservation mechanisms

• Contention-based protocols with scheduling mechanisms

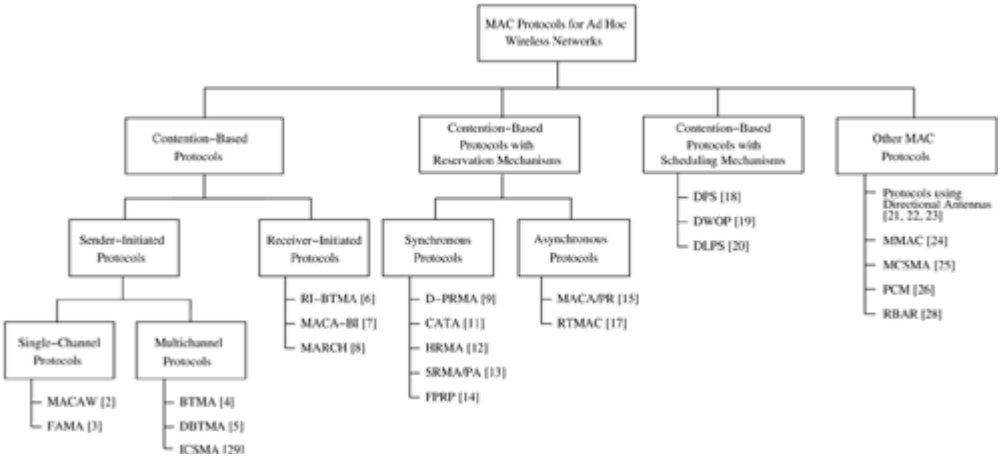


FIG:Classifications of MAC protocols.

Contention-Based Protocols

These protocols follow a contention-based channel access policy. A node does not make any resource reservation *a priori*. Whenever it receives a packet to be transmitted, it contends with its neighbor nodes for access to the shared channel. Contention-based protocols cannot provide QoS guarantees to sessions since nodes are not guaranteed regular access to the channel. Random access protocols can be further divided into two types:

• Sender-initiated protocols: Packet transmissions are initiated by the sender node.

• Receiver-initiated protocols: The receiver node initiates the contention resolution protocol.

Sender-initiated protocols can be further divided into two types:

• Single-channel sender-initiated protocols: In these protocols, the total available bandwidth is used as it is, without being divided. A node that wins the contention to the channel can make use of the entire bandwidth.

• Multichannel sender-initiated protocols: In multichannel protocols, the available bandwidth is divided into multiple channels. This enables several nodes to simultaneously transmit data, each using a separate channel. Some protocols dedicate a frequency channel exclusively for transmitting control information.

Contention-Based Protocols with Reservation Mechanisms

Ad hoc wireless networks sometimes may need to support real-time traffic, which requires QoS guarantees to be provided. In contention-based protocols, nodes are not guaranteed periodic access to the channel. Hence they cannot support real-time traffic. In order to support such traffic, certain protocols have mechanisms for reserving bandwidth *a priori*.Such protocols can provide QoS support to time-sensitive traffic sessions. These protocols can be further classified into two types:

• Synchronous protocols: Synchronous protocols require time synchronization among all nodes in the network, so that reservations made by a node are known to other nodes in its neighborhood. Global time synchronization is generally difficult to achieve.

• Asynchronous protocols: They do not require any global synchronization among nodes in the network. These protocols usually use relative time information for effecting reservations

Contention-Based Protocols with Scheduling Mechanisms

As mentioned earlier, these protocols focus on packet scheduling at nodes, and also scheduling nodes for access to the channel. Node scheduling is done in a manner so that all nodes are treated fairly and no node is starved of bandwidth. Scheduling-based schemes are also used for enforcing priorities among flows whose packets are queued at nodes. Some scheduling schemes also take into consideration battery characteristics, such as remaining battery power, while scheduling nodes for access to the channel.

CONTENTION-BASED PROTOCOLS

1 MACAW: A Media Access Protocol for Wireless LANs

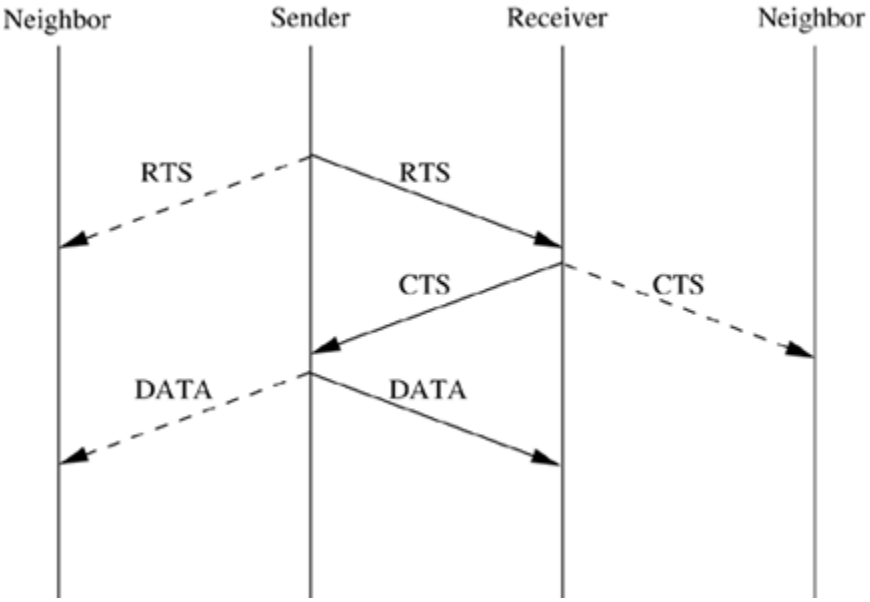
This protocol is based on the multiple access collision avoidance protocol (MACA) proposed by Karn [1]. MACA was proposed due to the shortcomings of CSMA protocols when used for wireless networks. In what follows, a brief description on why CSMA protocols fail in wireless networks is given. This is followed by detailed descriptions of the MACA protocol and the MACAW protocol.

MACA Protocol

The MACA protocol was proposed as an alternative to the traditional carrier sense multiple access (CSMA) protocols used in wired networks. In CSMA protocols, the sender first senses the

channel for the carrier signal. If the carrier is present, it retries after a random period of time. Otherwise, it transmits the packet. CSMA senses the state of the channel only at the transmitter. This protocol does not overcome the hidden terminal problem. In a typical ad hoc wireless network, the transmitter and receiver may not be near each other at all times. In such situations, the packets transmitted by a node are prone to collisions at the receiver due to simultaneous transmissions by the hidden terminals. Also, the bandwidth utilization inCSMA protocols is less because of the exposed terminal problem.

MACA does not make use of carrier-sensing for channel access. It uses two additional signaling packets: the request-to-send (RTS) packet and the clear-to-send (CTS) packet. When a node wants to transmit a data packet, it first transmits an RTS packet. The receiver node, on receiving the RTS packet, if it is ready to receive the data packet, transmits a CTSpacket. Once the sender receives the CTS packet without any error, it starts transmitting the data packet. This data transmission mechanism is depicted in Figure 6.3. If a packet transmitted by a node is lost, the node uses the binary exponential back-off (BEB) algorithm to back off for a random interval of time before retrying. In the binary exponential backoff mechanism, each time a collision is detected, the node doubles its maximum back-off window. Neighbor nodes near the sender that hear the RTS packet do not transmit for a long enough period of time so that the sender could receive the CTS packet. Both the RTS and theCTS packets carry the expected duration of the data packet transmission. A node near the receiver, upon hearing the CTS packet, defers its transmission till the receiver receives the data packet. Thus, MACA overcomes the hidden node problem. Similarly, a node receiving anRTS defers only for a short period of time till the sender could receive the CTS. If no CTS is heard by the node during its waiting period, it is free to transmit packets once the waiting interval is over. Thus, a node that hears only the RTS packet is free to transmit simultaneously when the sender of the RTS is transmitting data packets. Hence, the exposed terminal problem is also overcome in MACA. But MACA still has certain problems, which was why MACAW, described below, was proposed.

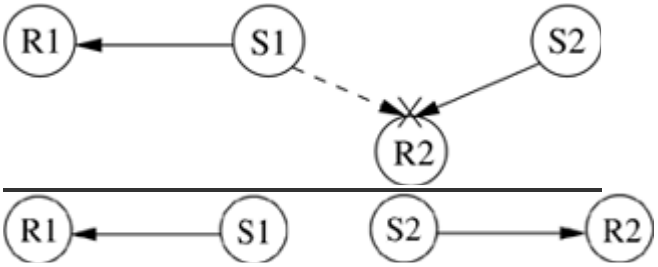


Packet transmission in MACA.

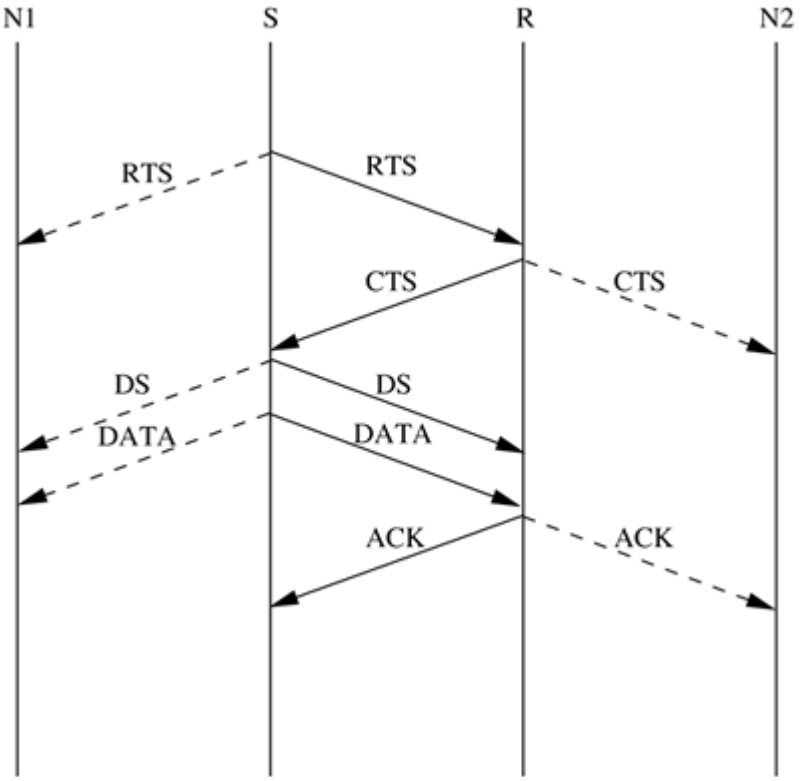
MACAW Protocol

The binary exponential back-off mechanism used in MACA at times starves flows. For example, consider Figure 6.4. Here both nodes S1 and S2 keep generating a high volume of traffic. The node that first captures the channel (say, node S1) starts transmitting packets. The packets transmitted by the other node S2 get collided, and the node keeps incrementing its back-off window according to the BEB algorithm. As a result, the probability of node S2 acquiring the channel keeps decreasing, and over a period of time it gets completely blocked. To overcome this problem, the back-off algorithm has been modified in MACAW [2]. The packet header now has an additional field carrying the current back-off counter value of the transmitting node. A node receiving the packet copies this value into its own back-off counter. This mechanism allocates bandwidth in a fair manner. Another problem with BEBalgorithm is that it adjusts the back-off counter value very rapidly, both when a node successfully transmits a packet and when a collision is detected by the node. The back-off counter is reset to the minimum value after every successful transmission. In the modified back-off process, this would require a period of contention to be repeated after each successful transmission in order to build up the back-off timer values. To prevent such large variations in the back-off values, a multiplicative increase and linear decrease (MILD) back-off mechanism is used in MACAW. Here, upon a collision, the back-off is increased by a multiplicative factor (1.5), and upon a successful transmission, it is

decremented by one. This eliminates contention and hence long contention periods after every successful transmission, at the same time providing a reasonably quick escalation in the back-off values when the contention is high.



Example topology.



Packet exchange in MACAW.

2 Floor Acquisition Multiple Access Protocols

The floor acquisition multiple access (FAMA) protocols [3] are based on a channel access discipline which consists of a carrier-sensing operation and a collision-avoidance dialog between the sender and the intended receiver of a packet. Floor acquisition refers to the process of gaining

control of the channel. At any given point of time, the control of the channel is assigned to only one node, and this node is guaranteed to transmit one or more data packets to different destinations without suffering from packet collisions. Carrier-sensing by the sender, followed by the RTS-CTS control packet exchange, enables the protocol to perform as efficiently as MACA [1] in the presence of hidden terminals, and as efficiently as CSMA otherwise. FAMA requires a node that wishes to transmit packets to first acquire the floor (channel) before starting to transmit the packets. The floor is acquired by means of exchanging control packets. Though the control packets themselves may collide with other control packets, it is ensured that data packets sent by the node that has acquired the channel are always transmitted without any collisions. Any single-channel MAC protocol that does not require a transmitting node to sense the channel can be adapted for performing floor acquisition tasks. Floor acquisition using the RTS-CTS exchange is advantageous as the mechanism also tries to provide a solution for the hidden terminal problem.

Two FAMA protocol variants are discussed in this section: RTS-CTS exchange with no carriersensing, and RTS-CTS exchange with non-persistent carrier-sensing. The first variant uses the ALOHA protocol for transmitting RTS packets, while the second variant uses nonpersistent CSMA for the same purpose.

Multiple Access Collision Avoidance

Multiple access collision avoidance (MACA) [1], which was discussed earlier in this chapter, belongs to the category of FAMA protocols. In MACA, a ready node transmits an RTS packet. A neighbor node receiving the RTS defers its transmissions for the period specified in theRTS. On receiving the RTS, the receiver node responds by sending back a CTS packet, and waits for a long enough period of time in order to receive a data packet. Neighbor nodes of the receiver which hear this CTS packet defer their transmissions for the time duration of the impending data transfer. In MACA, nodes do not sense the channel. A node defers its transmissions only if it receives an RTS or CTS packet. In MACA, data packets are prone to collisions with RTS packets.

According to the FAMA principle, in order for data transmissions to be collision-free, the duration of an RTS must be at least twice the maximum channel propagation delay. Transmission of bursts of packets is not possible in MACA. In FAMA-NTR (discussed below) the MACA protocol is modified to permit transmission of packet bursts by enforcing waiting periods on nodes, which are proportional to the channel propagation time.

FAMA – Non-Persistent Transmit Request

This variant of FAMA, called FAMA – non-persistent transmit request (FAMA-NTR), combines non-persistent carrier-sensing along with the RTS-CTS control packet exchange mechanism. Before sending a packet, the sender node senses the channel. If the channel is found to be busy, then the node backs off for a random time period and retries later. If the channel is found to be free, it transmits the RTS packet. After transmitting the RTS, the sender listens to the channel for one round-trip time in addition to the time required by the receiver node to transmit a CTS. If it does not receive the CTS within this time period or if theCTS received is found to be corrupted, then the node takes a random back-off and retries later. Once the sender node receives the CTS packet without any error, it can start transmitting its data packet burst. The burst is limited to a maximum number of data packets, after which the node releases the channel, and contends with other nodes to again acquire the channel.

In order to allow the sender node to send a burst of packets once it acquires the floor, the receiver node is made to wait for a time duration of τ seconds after processing each data packet received. Here, τ denotes the maximum channel propagation time. A waiting period of 2τ seconds is enforced on a transmitting node after transmitting any control packet. This is done to allow the RTS-CTS exchange to take place without any error. A node transmitting anRTS is required to wait for 2τ seconds after transmitting the RTS in order to enable the receiver node to receive the RTS and transmit the corresponding CTS packet. After sending the final data packet, a sender node is made to wait for τ seconds in order to allow the destination to receive the data packet and to account for the enforced waiting time at the destination node.

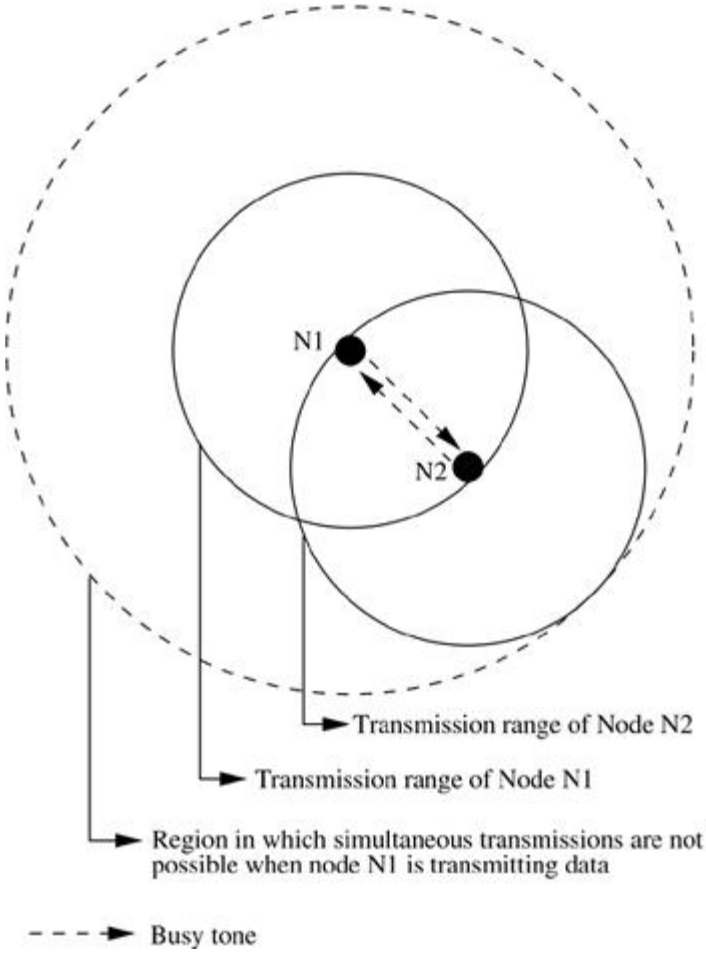
3 Busy Tone Multiple Access Protocols

Busy Tone Multiple Access

The busy tone multiple access (BTMA) protocol [4] is one of the earliest protocols proposed for overcoming the hidden terminal problem faced in wireless environments. The transmission channel is split into two: a data channel and a control channel. The data channel is used for data packet transmissions, while the control channel is used to transmit the busy tone signal. When a node is ready for transmission, it senses the channel to check whether the busy tone is active. If not, it turns on the busy tone signal and starts data transmission; otherwise, it reschedules the packet for transmission after some random rescheduling delay. Any other node which senses the carrier on the incoming data channel also transmits the busy tone signal on the

control channel. Thus, when a node is transmitting, no other node in the two-hop neighborhood of the transmitting node is permitted to simultaneously transmit. Though the probability of collisions is very low in BTMA, the bandwidth utilization is very poor.Figure shows the worst-case scenario where the node density is very high; the dotted circle shows the region in which nodes are blocked from simultaneously transmitting when node N1 is transmitting packets.

Transmission in BTMA.



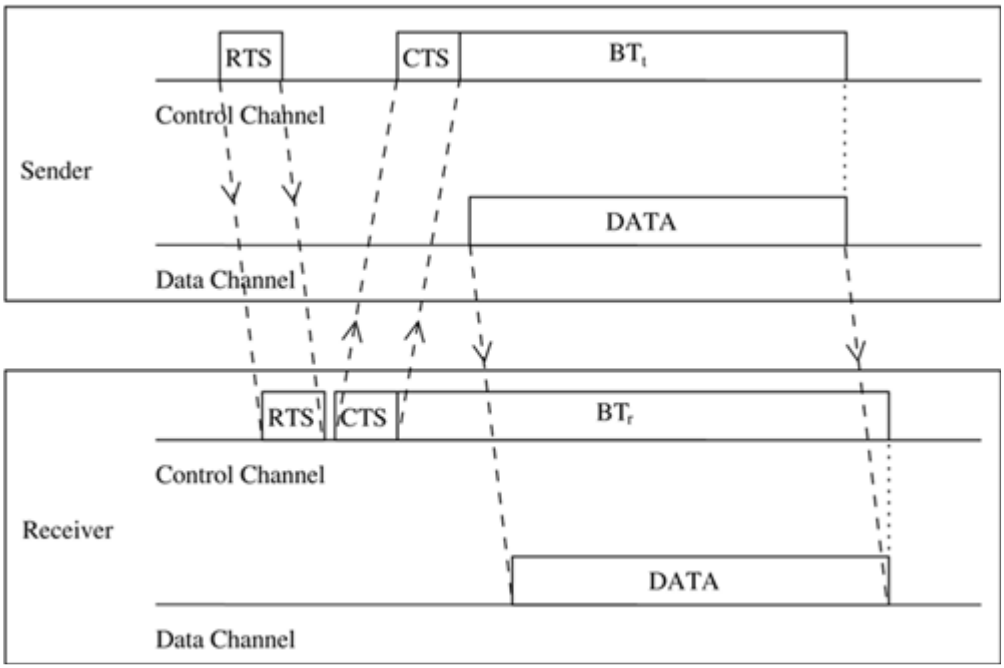
Dual Busy Tone Multiple Access Protocol

The dual busy tone multiple access protocol (DBTMA) [5] is an extension of the BTMAscheme. Here again, the transmission channel is divided into two: the data channel and the control channel. As in BTMA, the data channel is used for data packet transmissions. The control channel is used for control packet transmissions (RTS and CTS packets) and also for

transmitting the busy tones. DBTMA uses two busy tones on the control channel, *BTt* and *BTr* . The *BTt* tone is used by the node to indicate that it is transmitting on the data channel. The *BTr* tone is turned on by a node when it is receiving data on the data channel. The two busy tone signals are two sine waves at different well-separated frequencies.

When a node is ready to transmit a data packet, it first senses the channel to determine whether the *BTr* signal is active. An active *BTr* signal indicates that a node in the neighborhood of the ready node is currently receiving packets. If the ready node finds that there is no *BTr* signal, it transmits the RTS packet on the control channel. On receiving theRTS packets, the node to which the RTS was destined checks whether the *BTt* tone is active in its neighborhood. An active *BTt* implies that some other node in its neighborhood is transmitting packets and so it cannot receive packets for the moment. If the node finds no*BTt* signal, it responds by sending a CTS packet and then turns on the *BTr* signal (which informs other nodes in its neighborhood that it is receiving). The sender node, on receiving this CTS packet, turns on the *BTt* signal (to inform nodes in its neighborhood that it is transmitting) and starts transmitting data packets.

After completing transmission, the sender node turns off the *BTt* signal. The receiver node, after receiving all data packets, turns off the*BTr* signal. The above process is depicted in Figure

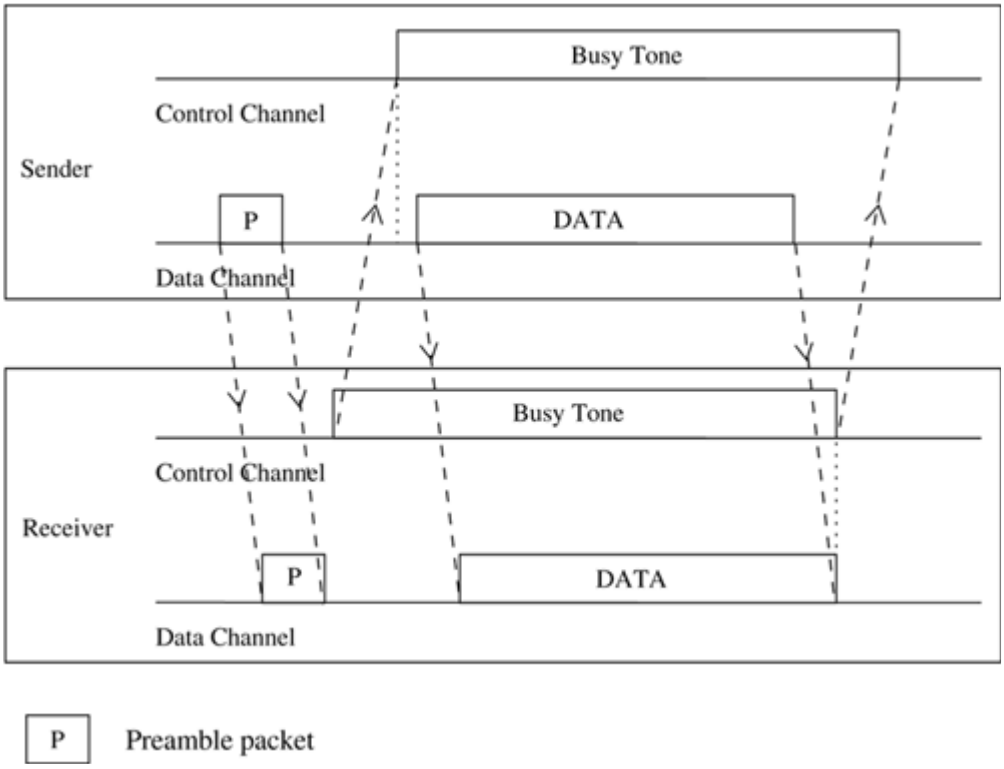


Packet transmission in DBTMA.

Receiver-Initiated Busy Tone Multiple Access Protocol

In the receiver-initiated busy tone multiple access protocol (RI-BTMA) [6], similar toBTMA, the available bandwidth is divided into two channels: a data channel for transmitting data packets and a control channel. The control channel is used by a node to transmit the busy tone signal. A node can transmit on the data channel only if it finds the busy tone to be absent on the control channel. The data packet is divided into two portions: a preamble and the actual data packet. The preamble carries the identification of the intended destination node. Both the data channel and the control channel are slotted, with each slot equal to the length of the preamble. Data transmission consists of two steps. First, the preamble needs to be transmitted by the sender. Once the receiver node acknowledges the reception of this preamble by transmitting the busy tone signal on the control channel, the actual data packet is transmitted. A sender node that needs to transmit a data packet first waits for a free slot, that is, a slot in which the busy tone signal is absent on the control channel. Once it finds such a slot, it transmits the preamble packet on the data channel. If the destination node receives this preamble packet correctly without any error, it transmits the busy tone on the control channel. It continues transmitting the busy tone signal as long as it is receiving data from the sender. If preamble transmission fails, the receiver does not acknowledge with the busy tone, and the sender node waits for the next free slot and tries again. The operation of the RI-BTMA protocol is shown in Figure 6.10. The busy tone serves two purposes. First, it acknowledges the sender about the successful reception of the preamble. Second, it informs the nearby hidden nodes about the impending transmission so that they do not transmit at the same time.

Packet transmission in RI-BTMA.

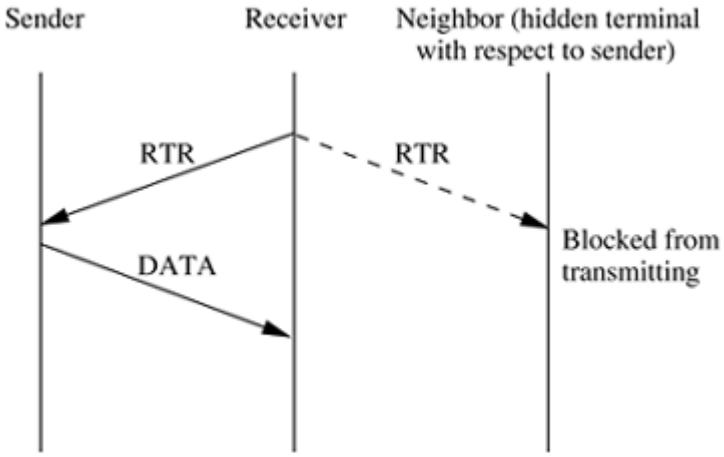


There are two types of RI-BTMA protocols: the basic protocol and the controlled protocol. The basic packet transmission mechanism is the same in both protocols. In the basic protocol, nodes do not have backlog buffers to store data packets. Hence packets that suffer collisions cannot be retransmitted. Also, when the network load increases, packets cannot be queued at the nodes. This protocol would work only when the network load is not high; when network load starts increasing, the protocol becomes unstable. The controlled protocol overcomes this problem. This protocol is the same as the basic protocol, the only difference being the availability of backlog buffers at nodes. Therefore, packets that suffer collisions, and those that are generated during busy slots, can be queued at nodes. A node is said to be in the backlogged mode if its backlog buffer is non-empty. When a node in the backlogged mode receives a packet from its higher layers, the packet is put into the buffer and transmitted later. Suppose the packet arrives at a node when it is not in the backlogged mode, then if the current slot is free, the preamble for the packet is transmitted with probability p in the current slot itself (not transmitted in the same slot with probability $(1 - p)$). If the packet was received during a busy slot, the packet is just put into the backlog buffer, where it waits until the next free slot. A

backlogged node transmits a backlogged packet in the next idle slot with a probability *q*. All other packets in the backlog buffer just keep waiting until this transmission succeeds.
This protocol can work for multi-hop radio networks as well as for single-hop fully connected networks.

4 MACA-By Invitation

MACA-by invitation (MACA-BI) [7] is a receiver-initiated MAC protocol. It reduces the number of control packets used in the MACA [1] protocol. MACA, which is a sender-initiated protocol, uses the three-way handshake mechanism (which was shown in Figure 6.3), where first the RTS and CTS control packets are exchanged, followed by the actual DATA packet transmission. MACA-BI eliminates the need for the RTS packet. In MACA-BI the receiver node initiates data transmission by transmitting a ready to receive (RTR) control packet to the sender (Figure 6.11). If it is ready to transmit, the sender node responds by sending a DATA packet. Thus data transmission in MACA-BI occurs through a two-way handshake mechanism.
Packet transmission in MACA-BI.



The receiver node may not have an exact knowledge about the traffic arrival rates at its neighboring sender nodes. It needs to estimate the average arrival rate of packets. For providing necessary information to the receiver node for this estimation, the DATA packets are modified to carry control information regarding the backlogged flows at the transmitter node, number of packets queued, and packet lengths. Once this information is available at the receiver node, the average rate of the flows can be easily estimated. Suppose the estimation is incorrect or is not possible (when the first data packet of the session is to be transmitted), theMACA-BI protocol

can be extended by allowing the sender node to declare its backlog through an RTS control packet, if an RTR packet is not received within a given timeout period. In MACA, the CTS packet was used to inform the hidden terminals (nodes) about the impending DATA packet transmission, so that they do not transmit at the same time and disrupt the session. This role is played in MACA-BI by the RTR packets. An RTR packet carries information about the time interval during which the DATA packet would be transmitted. When a node hears RTR packets transmitted by its neighbors, it can obtain information about the duration of DATA packet transmissions by nodes that may be either its direct one-hop neighbors or its two hop neighbors, that is, hidden terminals. Since it has information about transmissions by the hidden terminals, it refrains from transmitting during those periods (Figure 6.11). Hence the hidden terminal problem is overcome inMACA-BI. Collision among DATA packets is impossible.

However, the hidden terminal problem still affects the control packet transmissions. This leads to protocol failure, as in certain cases the RTR packets can collide with DATA packets. One such scenario is depicted in Figure 6.12. Here, RTR packets transmitted by receiver nodes R1 and R2 collide at node A. So node A is not aware of the transmissions from nodes S1 and S2. When node A transmits RTR packets, they collide with DATA packets at receiver nodes R1 and R2.

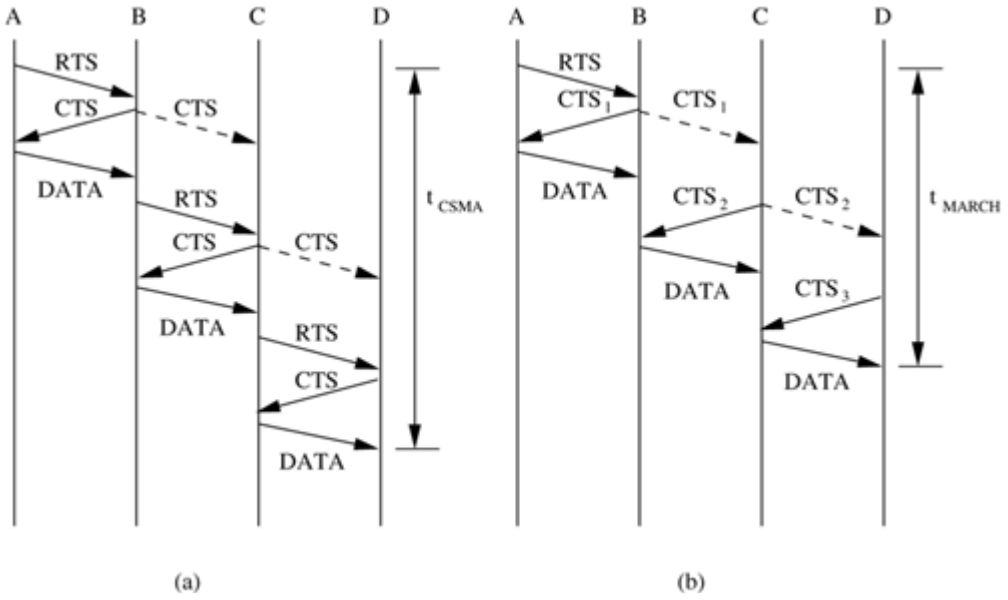
S1, S2 – Sender nodes
R1, R2 – Receiver nodes
A – Neighbor node

5 Media Access with Reduced Handshake

The media access with reduced handshake protocol (MARCH) [8] is a receiver-initiated protocol. MARCH, unlike MACA-BI [7], does not require any traffic prediction mechanism. The protocol exploits the broadcast nature of traffic from omnidirectional antennas to reduce the number of handshakes involved in data transmission. In MACA [1], the RTS-CTS control packets exchange takes place before the transmission of every data packet. But in MARCH, the RTS packet is used only for the first packet of the stream. From the second packet onward, only the CTS packet is used.

A node obtains information about data packet arrivals at its neighboring nodes by overhearing the CTS packets transmitted by them. It then sends a CTS packet to the concerned neighbor node for relaying data from that node. This mechanism is illustrated inFigure 6.13. Figure 6.13 (a) depicts the packet exchange mechanism of MACA. Here two control packets RTS and CTS need to be exchanged before each data packet is transmitted. It can be seen from this figure that node C, for example, can hear both the CTS and the RTSpackets transmitted by node B. MARCH uses this property of the broadcast channel to reduce the two way handshake into a single CTS-only handshake. Figure 6.13 (b) shows the handshake mechanism of MARCH. Here, when node B transmits the *CTS*1 packet, this packet is also heard by node C. A CTS packet carries information regarding the duration of the next data packet. Node C therefore determines the time at which the next data packet would be available at node B. It sends the *CTS*2 packet at that point of time. On receiving the *CTS*2packet, node B sends the data packet directly to node C. It can be observed from the figure that the time taken for a packet transmitted by node A to reach node D in MARCH, that is,*tMARCH*, is less compared to the time taken in MACA, *tMACA*.

Handshake mechanism in (a) MACA and (b) MARCH.

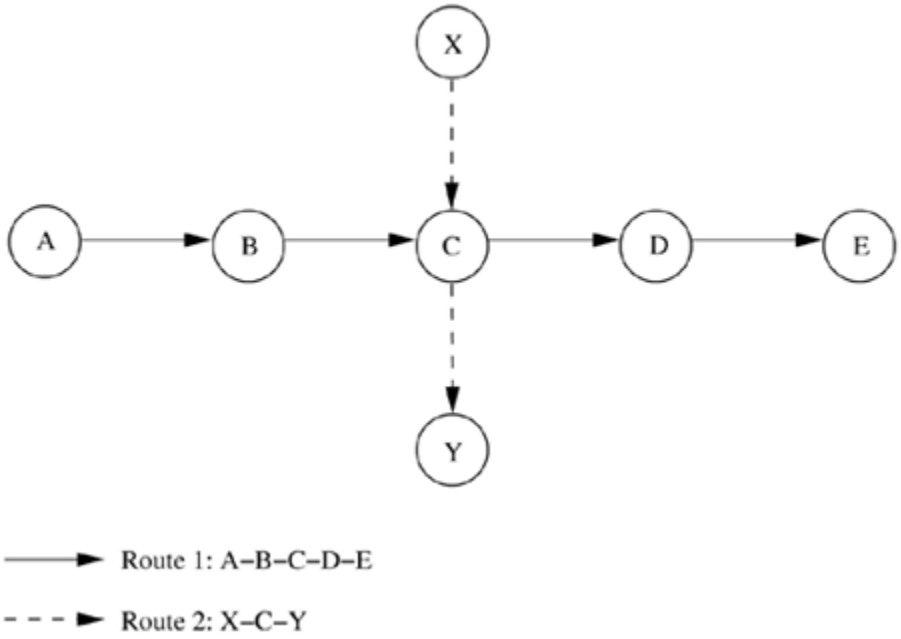


(a) (b)

The CTS packet carries the MAC addresses of the sender and the receiver node, and the route identification number (*RTid*) for that flow. The *RTid* is used by nodes in order to avoid misinterpretation of CTS packets and initiation of false CTS-only handshakes. Consider Figure. Here there are two routes – Route 1: A-B-C-D-E and Route 2: X-C-Y. When node C hears a CTS

packet transmitted by node B, by means of the *RTid* field on the packet, it understands that the CTS was transmitted by its upstream node (upstream node refers to the next hop neighbor node on the path from the current node to the source node of the data session) on Route 1. It invokes a timer T which is set to expire after a certain period of time, long enough for node B to receive a packet from node A. A CTS packet is transmitted by node C once the timer expires. This CTS is overheard by node Y also, but since the *RTid* carried on the CTS is different from the *RTid* corresponding to Route 2, node Y does not respond. In MARCH, the MAC layer has access to tables that maintain routing information (such as*RTid),* but the protocol as such does not get involved in routing.

Figure 6.14. Example topology.



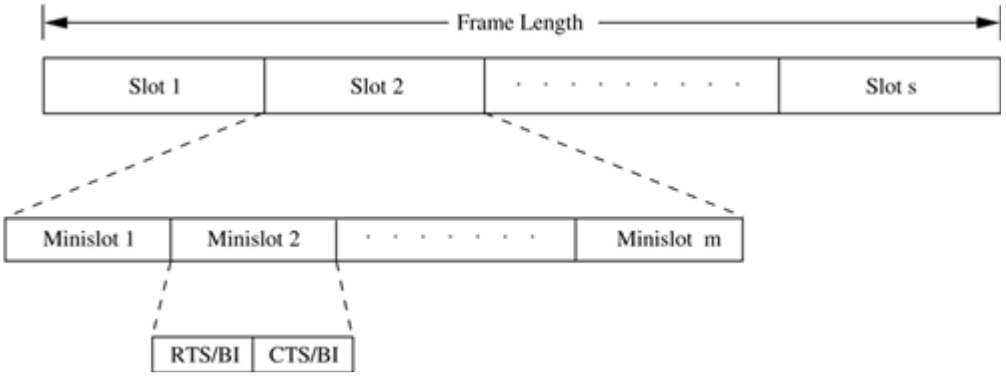
CONTENTION-BASED PROTOCOLS WITH RESERVATION MECHANISMS

Protocols described in this section have certain mechanisms that aid the nodes in effecting bandwidth reservations. Though these protocols are contention-based, contention occurs only during the resource (bandwidth) reservation phase. Once the bandwidth is reserved, the node gets exclusive access to the reserved bandwidth. Hence, QoS support can be provided for real-time traffic

1 Distributed Packet Reservation Multiple Access Protocol

The distributed packet reservation multiple access protocol (D-PRMA) [9] extends the earlier centralized packet reservation multiple access (PRMA) [10] scheme into a distributed scheme that can be used in ad hoc wireless networks. PRMA was proposed for voice support in a wireless LAN with a base station, where the base station serves as the fixed entity for theMAC operation. D-PRMA extends this protocol for providing voice support in ad hoc wireless networks. D-PRMA is a TDMA-based scheme. The channel is divided into fixed- and equal-sized frames along the time axis (Figure 6.15). Each frame is composed of *s* slots, and each slot consists of *m* minislots. Each minislot can be further divided into two control fields, RTS/BI and CTS/BI(BI stands for busy indication), as shown in the figure. These control fields are used for slot reservation and for overcoming the hidden terminal problem. Details on how this is done will be explained later in this section. All nodes having packets ready for transmission contend for the first minislot of each slot. The remaining (*m* - 1) minislots are granted to the node that wins the contention. Also, the same slot in each subsequent frame can be reserved for this winning terminal until it completes its packet transmission session. If no node wins the first minislot, then the remaining minislots are continuously used for contention, until a contending node wins any minislot. Within a reserved slot, communication between the source and receiver nodes takes place by means of either time division duplexing (TDD) or frequency division duplexing (FDD). Any node that wants to transmit packets has to first reserve slots, if they have not been reserved already. A certain period at the beginning of each minislot is reserved for carrier-sensing. If a sender node detects the channel to be idle at the beginning of a slot (minislot 1), it transmits an RTS packet (slot reservation request) to the intended destination through the RTS/BI part of the current minislot. On successfully receiving this RTS packet, thereceiver node responds by sending a CTS packet through theCTS/BI of the same minislot. If the sender node receives this CTS successfully, then it gets the reservation for the current slot and can use the remaining minislots, that is, minislots 2 to*m*. Otherwise, it continues the contention process through the subsequent minislots of the same slot.

Frame structure in D-PRMA.



In order to prioritize nodes transmitting voice traffic (voice nodes) over nodes transmitting normal data traffic (data nodes), two rules are followed in D-PRMA. According to the first rule, the voice nodes are allowed to start contending from minislot 1 with probability $p = 1$; data nodes can start contending only with probability $p < 1$. For the remaining $(m - 1)$ minislots, both the voice nodes and the data nodes are allowed to contend with probability $p < 1$. This is because the reservation process for a voice node is triggered only after the arrival of voice traffic at the node; this avoids unnecessary reservation of slots. According to the second rule, only if the node winning the minislot contention is a voice node, is it permitted to reserve the same slot in each subsequent frame until the end of the session. If a data node wins the contention, then it is allowed to use only one slot, that is, the current slot, and it has to make fresh reservations for each subsequent slot.

Nodes that are located within the radio coverage of the receiver should not be permitted to transmit simultaneously when the receiver is receiving packets. If permitted, packets transmitted by them may collide with the packets of the on-going traffic being received at the receiver. Though a node which is located outside the range of a receiver is able to hear packets transmitted by the sender, it should still be allowed to transmit simultaneously. The above requirements, in essence, mean that the protocol must be free of the hidden terminal and exposed terminal problems. In D-PRMA, when a node wins the contention in minislot 1, other terminals must be prevented from using any of the remaining $(m - 1)$ minislots in the same slot for contention (*requirement 1*). Also, when a slot is reserved in subsequent frames, other nodes should be prevented from contending for those reserved slots (*requirement 2*). The RTS-CTS exchangemechanism taking place in the reservation process helps in trying to satisfy *requirement 1*. A node that wins the contention in minislot 1 starts transmitting immediately from minislot 2. Any other node that wants to transmit will find the channel to be busy from minislot 2. Since an

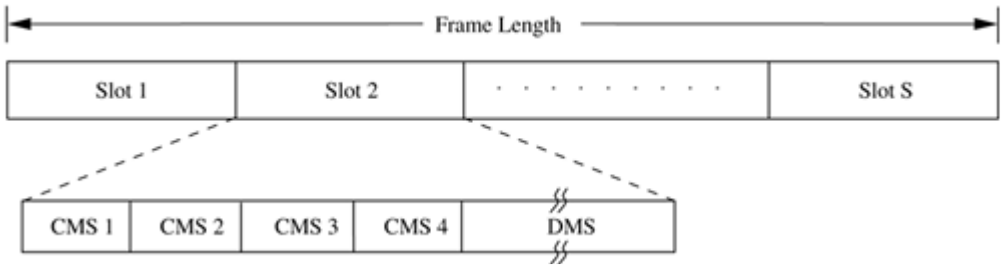
RTS can be sent only when the channel is idle, other neighboring nodes would not contend for the channel until the on-going transmission gets completed. A node sends an RTS in the RTS/BI part of a minislot. Only a node that receives an RTS destined to it is allowed to use the CTS/BI part of the slot for transmitting the CTS. So the CTS packet does not suffer any collision due to simultaneous RTS packet transmissions. This improves the probability for a successful reservation. In order to avoid the hidden terminal problem, all nodes hearing the CTS sent by the receiver are not allowed to transmit during the remaining period of that same slot. In order to avoid the exposed terminal problem, a node hearing the RTS but not the CTS (sender node's neighbor) is still allowed to transmit. But, if the communication is duplex in nature, where a node may transmit and receive simultaneously, even such exposed nodes (that hear RTS alone) should not be allowed to transmit. Therefore, D-PRMA makes such a node defer its transmissions for the remaining time period of the same slot. If an RTS or CTS packet collides, and a successful reservation cannot be made in the first minislot, then the subsequent (m - 1) minislots of the same slot are used for contention. ***6.6.2 Collision Avoidance Time Allocation Protocol*** The collision avoidance time allocation protocol (CATA) [11] is based on dynamic topology dependent ransmission scheduling. Nodes contend for and reserve time slots by means of a distributed reservation and handshake mechanism. CATA supports broadcast, unicast, and multicast transmissions simultaneously. The operation of CATA is based on two basic principles:

• The receiver(s) of a flow must inform the potential source nodes about the reserved slot on which it is currently receiving packets. Similarly, the source node must inform the potential destination node(s) about interferences in the slot.

• Usage of negative acknowledgments for reservation requests, and control packet transmissions at the beginning of each slot, for distributing slot reservation information to senders of broadcast or multicast sessions.

Time is divided into equal-sized frames, and each frame consists of S slots (Figure 6.16). Each slot is further divided into five minislots. The first four minislots are used for transmitting control packets and are called control minislots (CMS1, CMS2, CMS3, and CMS4). The fifth and last minislot, called data minislot (DMS), is meant for data transmission. The data minislot is much longer than the control minislots as the control packets are much smaller in size compared to data packets.

Frame format in CATA.



Each node that receives data during the DMS of the current slot transmits a slot reservation (SR) packet during the CMS1 of the slot. This serves to inform other neighboring potential sender nodes about the currently active reservation. The SR packet is either received without error at the neighbor nodes or causes noise at those nodes, in both cases preventing such neighbor nodes from attempting to reserve the current slot. Every node that transmits data during the DMS of the current slot transmits a request-to-send (RTS) packet during CMS2 of the slot. This RTS packet, when received by other neighbor nodes or when it collides with other RTS packets at the neighbor nodes, causes the neighbor nodes to understand that the source node is scheduled to transmit during the DMS of the current slot. Hence they defer their transmissions during the current slot.

The control minislots CMS3 and CMS4 are used as follows. The sender of an intended reservation, if it senses the channel to be idle during CMS1, transmits an RTS packet during CMS2. The receiver node of a unicast session transmits a clear-to-send (CTS) packet during CMS3. On receiving this packet, the source node understands that the reservation was successful and transmits data during the DMS of that slot, and during the same slot in subsequent frames, until the unicast flow gets terminated. Once the reservation has been made successfully in a slot, from the next slot onward, both the sender and receiver do not transmit anything during CMS3, and during CMS4 the sender node alone transmits a not-to-send (NTS) packet. The purpose of the NTS packet is explained below. If a node receives anRTS packet for broadcast or multicast during CMS2, or if it finds the channel to be free during CMS2, it remains idle and does not transmit anything during CMS3 and CMS4. Otherwise, it sends a not-to-send (NTS) packet during CMS4. The NTS packet serves as a negative acknowledgment; a potential multicast or broadcast source node that receives the NTS packet during CMS4, or that detects noise during CMS4, understands that its reservation request had failed, and it does not transmit during the

DMS of the current slot. If it finds the channel to be free during CMS4, which implies that its reservation request was successful, it starts transmitting the multicast or broadcast packets during the DMS of the slot. The length of the frame is very important in CATA. For any node (say, node A) to broadcast successfully, there must be no other node (say, node B) in its two-hop neighborhood that transmits simultaneously. If such a node B exists, then if node B is within node A's one-hop neighborhood, node A and node B cannot hear the packets transmitted by each other. If node B is within the two-hop neighborhood of node A, then the packets transmitted by nodes A and B would collide at their common neighbor nodes. Therefore, for any node to transmit successfully during one slot in every frame, the number of slots in each frame must be larger than the number of two-hop neighbor nodes of the transmitting node. The worst-case value of the frame length, that is, the number of slots in the frame, would be $Min(d2 + 1, N)$, where d is the maximum degree (degree of a node refers to the count of one-hop neighbors of the node) of a node in the network, and N is the total number of nodes in the network.

CATA works well with simple single-channel half-duplex radios. It is simple and provides support for collision-free broadcast and multicast traffic.

6.6.3 Hop Reservation Multiple Access Protocol

The hop reservation multiple access protocol (HRMA) [12] is a multichannel MAC protocol which is based on simple half-duplex, very slow frequency-hopping spread spectrum (FHSS) radios. It uses a reservation and handshake mechanism to enable a pair of communicating nodes to reserve a frequency hop, thereby guaranteeing collision-free data transmission even in the presence of hidden terminals. HRMA can be viewed as a time slot reservation protocol where each time slot is assigned a separate frequency channel.

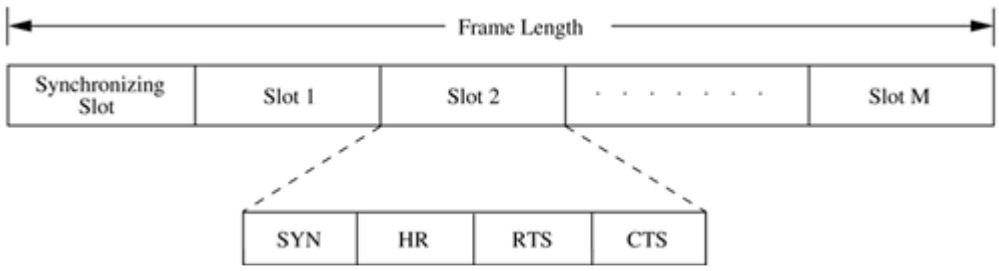
Out of the available L frequency channels, HRMA uses one frequency channel, denoted by $f0$, as

a dedicated synchronizing channel. The nodes exchange synchronization information on$f0$. The remaining L - 1 frequencies are divided into frequency pairs (denoted by (fi,

), i = 1, 2, 3, .., M), thereby restricting the length of the hopping sequence to M. fi is used for transmitting and receiving hop-reservation (HR) packets, request-to-send (RTS) packets, clearto-send (CTS) packets, and data packets. is used for sending and receiving acknowledgment (ACK) packets for the data packets received or transmitted on frequency fi. In HRMA, time is

slotted, and each slot is assigned a separate frequency hop, which is one among the *M* frequency hops in the hopping sequence. Each time slot is divided into four periods, namely, synchronizing period, HR period, RTS period, and CTS period, each period meant for transmitting or receiving the synchronizing packet, FR packet, RTS packet, andCTS packet, respectively. All idle nodes, that is, nodes that do not transmit or receive packets currently, hop together. During the synchronizing period of each slot, all idle nodes hop to the synchronizing frequency *f*0 and exchange synchronization information. During the HR,RTS, and CTS periods, they just stay idle, dwelling on the common frequency hop assigned to each slot. In addition to the synchronization period used for synchronization purposes, an exclusive synchronization slot is also defined at the beginning of each HRMA frame (Figure 6.17). This slot is of the same size as that of the other normal slots. All idle nodes dwell on the synchronizing frequency *f*0 during the synchronizing slot and exchange synchronization information that may be used to identify the beginning of a frequency hop in the common hopping sequence, and also the frequency to be used in the immediately following hop. Thus the HRMA frame, as depicted in Figure 6.17, is composed of the single synchronizing slot, followed by *M* consecutive normal slots.

Frame format in HRMA.

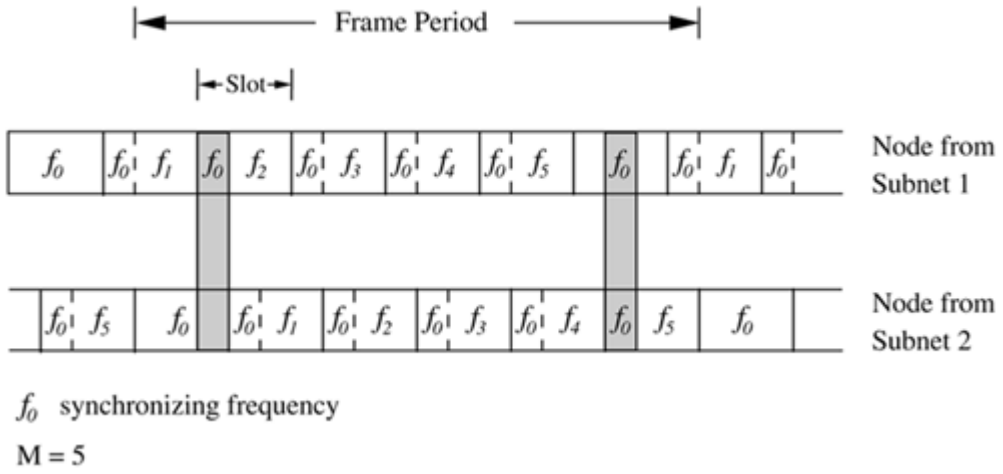


When a new node enters the network, it remains on the synchronizing frequency *f*0 for a long enough period of time so as to gather synchronization information such as the hopping pattern and the timing of the system. If it receives no synchronization information, it assumes that it is the only node in the network, broadcasts its own synchronization information, and forms a onenode

system. Since synchronization information is exchanged during every synchronization slot, new nodes entering the system can easily join the network. If μ is the length of each slot and μs the length of the synchronization period on each slot, then the dwell time of *fo* at the beginning of each frame would be $\mu + \mu s$. Consider the case where nodes from two different

disconnected network partitions come nearby. Figure 6.18depicts the worst-case frequency overlap scenario. In the figure, the maximum number of frequency hops $M = 5$. It is evident from the figure that within any time period equal to the duration of a HRMA frame, any two nodes from the two disconnected partitions always have at least two overlapping time periods of length μs on the synchronizing frequency $f0$. Therefore, nodes belonging to disconnected network
components can easily merge into a single network.

Figure 6.18. Merging of subnets.



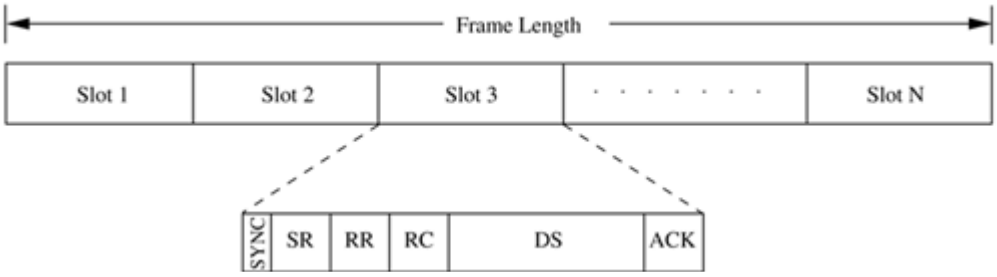
When a node receives data to be transmitted, it first listens to the HR period of the immediately following slot. If it hears an HR packet, it backs off for a randomly chosen period (which is a multiple of slot time). If it finds the channel to be free during the SR period, it transmits an RTS packet to the destination during the RTS period of the slot and waits for theCTS packet. On receiving the RTS, the destination node transmits the CTS packet during theCTS period of the same slot, stays on the same frequency currently being used, and waits for the data packet. If the source node receives the CTS packet correctly, it implies that the source and receiver nodes have successfully reserved the current hop. In case the source node does not receive
any CTS packet, it backs off for a random number of time slots and repeats the entire process again. The source and receiver nodes dwell on the same reserved frequency throughout the data transmission process, which starts immediately after the CTS period. As mentioned earlier, a separate frequency (, $i = 1, 2, ..., M$) is used for transmitting acknowledgments.

6.6.4 Soft Reservation Multiple Access with Priority Assignment

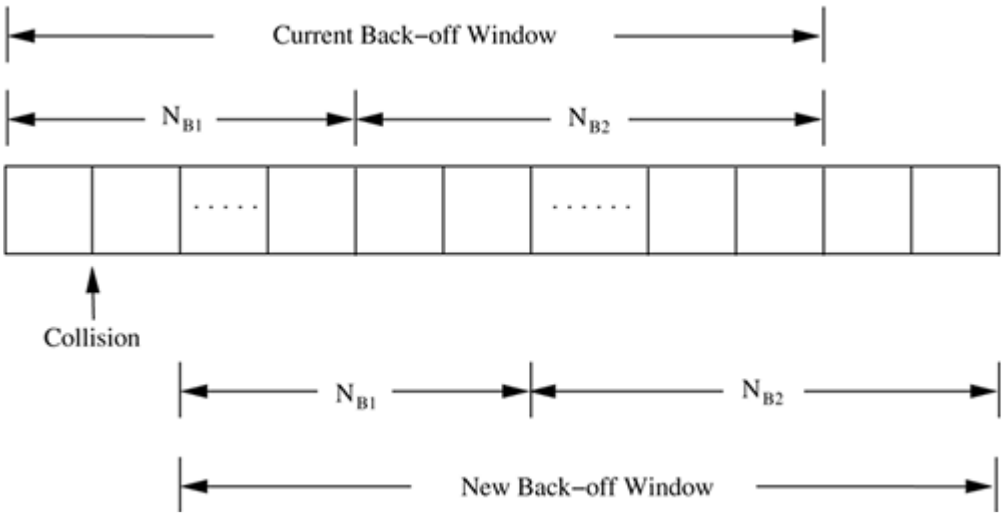
Soft reservation multiple access protocol with priority assignment (SRMA/PA) [13] was developed with the main objective of supporting integrated services of real-time and non-realtime
applications in ad hoc wireless networks, at the same time maximizing the statistical multiplexing gain. Nodes use a *collision-avoidance* handshake mechanism and a*soft reservation* mechanism in order to contend for and effect reservation of time slots. The soft reservation mechanism allows any urgent node, transmitting packets generated by a real-time application, to take over the radio resource from another node of a non-real-time application on an on-demand basis. SRMA/PA is a TDMA-based protocol in which nodes are allocated different time slots so that the transmissions are collision-free. The main features ofSRMA/PA are a unique frame structure and soft reservation capability for distributed and dynamic slot scheduling, dynamic and distributed access priority assignment and update policies, and a time-constrained back-off algorithm.

Time is divided into frames, with each frame consisting of a fixed number (*N*) of time slots. The frame structure is shown in Figure 6.19. Each slot is further divided into six different fields, SYNC, soft reservation (SR), reservation request (RR), reservation confirm (RC), data sending (DS), and acknowledgment (ACK). The SYNC field is used for synchronization purposes. The SR, RR, RC, and ACK fields are used for transmitting and receiving the corresponding control packets. The DS field is used for data transmission. The SR packet serves as a busy tone. It informs the nodes in the neighborhood of the transmitting node about the reservation of the slot. The SR packet also carries the access priority value assigned to the node that has reserved the slot. When an idle node receives a data packet for transmission, the node waits for a free slot and transmits the RR packet in the RR field of that slot. A node determines whether or not a slot is free through the SR field of that slot. In case of a voice terminal node, the node tries to take control of the slot already reserved by a data terminal if it finds its priority level to be higher than that of the data terminal. This process is called *soft reservation.* This makes
the SRMA/PA different from other protocols where even if a node has lower access priority compared to other ready nodes, it proceeds to complete the transmission of the entire data burst once it has reserved the channel.

Frame structure in SRMA/PA.



Priority levels are initially assigned to nodes based on the service classes (real-time or non-realtime)
in a static manner. Once the node acquires the channel, the corresponding slot stays
reserved for the node until the node completes transmitting the entire data burst. The node is assigned a prespecified priority, or , respectively, for voice and data terminals. R
denotes that the node is a reserved node, that is, a node that has successfully reserved the slot. It is required that , such that delay-sensitive voice applications get preference over
normal data applications. Whenever the reservation attempt fails due to collision, the access priority of the node is updated based on the urgency of its packets.



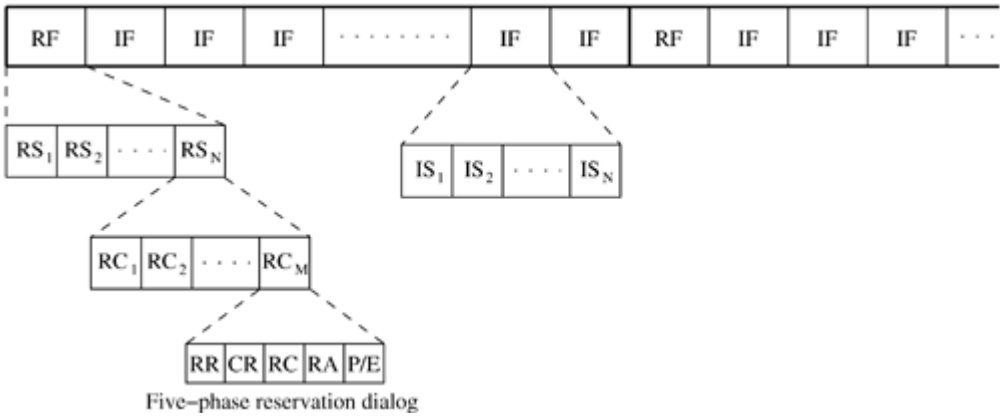
Five-Phase Reservation Protocol

The five-phase reservation protocol (FPRP) [14] is a single-channel time division multiple access (TDMA)-based broadcast scheduling protocol. Nodes use a contention mechanism in order to acquire time slots. The protocol is fully distributed, that is, multiple reservations can be simultaneously made throughout the network. No ordering among nodes is followed; nodes need

not wait for making time slot reservations. The slot reservations are made using a five-phase reservation process. The reservation process is localized; it involves only the nodes located within the two-hop radius of the node concerned. Because of this, the protocol is insensitive to the network size, that is, it is scalable. FPRP also ensures that no collisions occur due to the hidden terminal problem.

Time is divided into frames. There are two types of frames: reservation frame (RF) and information frame (IF). Each RF is followed by a sequence of IFs. Each RF has *N* reservation slots (RS), and each IF has *N* information slots (IS). In order to reserve an IS, a node needs to contend during the corresponding RS. Based on these contentions, a TDMA schedule is generated in the RF and is used in the subsequent IFs until the next RF. The structure of the frames is shown in Figure 6.21. Each RS is composed of *M* reservation cycles (RC). Within each RC, a five-phase dialog takes place, using which a node reserves slots. If a node wins the contention in an RC, it is said to have reserved the IS corresponding to the current RS in the subsequent IFs of the current frame. Otherwise, the node contends during the subsequent RCs of the current RS until itself or any other node (1-hop or 2-hop neighbor) succeeds. During the corresponding IS, a node would be in one of the following three states: transmit (T), receive (R), or blocked (B). The five-phase dialog ensures that the protocol is free from the hidden node problem, and also ensures that once a reservation is made by a node with a high probability, it gets sole access to the slot within its neighborhood

Frame structure in FPRP.



The protocol assumes the availability of global time at all nodes. Each node therefore knows when a five-phase cycle would start. The five phases of the reservation process are as follows:

1. Reservation request phase: Nodes that need to transmit packets send reservation request (RR) packets to their destination nodes.

2. Collision report phase: If a collision is detected by any node during the reservation request phase,

then that node broadcasts a collision report (CR) packet. The corresponding source nodes, upon receiving the CR packet, take necessary action.

3. Reservation confirmation phase: A source node is said to have won the contention for a slot if it

does not receive any CR messages in the previous phase. In order to confirm the reservation request made in the reservation request phase, it sends a reservation confirmation (RC) message to the destination node in this phase.

4. Reservation acknowledgment phase: In this phase, the destination node acknowledges reception

of the RC by sending back a reservation acknowledgment (RA) message to the source. The hidden nodes that receive this message defer their transmissions during the reserved slot.

5. Packing and elimination (P/E) phase: Two types of packets are transmitted during this phase: packing packet and elimination packet. The details regarding the use of these packets will be described later in this section.

Each of the above five phases is described below.

Reservation request phase:

In this phase, each node that needs to transmit packets sends an RR packet to the intended destination node with a contention probability *p,* in order to reserve an IS. Such nodes that send RR packets are called requesting nodes (RN). Other nodes just keep listening during this phase.

Collision report phase:

If any of the listening nodes detects collision of RR packets transmitted in the previous phase, it broadcasts a collision report (CR) packet. By listening for CR packets in this phase, an RNcomes to know about collision of the RR packet it had sent. If no CR is heard by the RN in this phase, then it assumes that the RR packet did not collide in its neighborhood. It then becomes a transmitting node (TN). Once it becomes a transmitting node, the node proceeds to the next phase, the reservation confirmation phase. On the other hand, if it hears a CRpacket in this phase,

it waits until the next reservation request phase, and then tries again. Thus, if two RNs are hidden from each other, their RR packets collide, both receive CRpackets, and no reservation is made, thereby eliminating the hidden terminal problem.

Reservation confirmation phase:

An RN that does not receive any CR packet in the previous phase, that is, a TN, sends an RCpacket to the destination node. Each neighbor node that receives this packet understands that the slot has been reserved, and defers its transmission during the corresponding information slots in the subsequent information frames until the next reservation frame.

Reservation acknowledgment phase:

On receiving the RC packet, the intended receiver node responds by sending an RA packet back to the TN. This is used to inform the TN that the reservation has been established. In case the TN is isolated and is not connected to any other node in the network, then it would not receive the RA packet, and thus becomes aware of the fact that it is isolated. Thus the RCpacket prevents such isolated nodes from transmitting further. The reservation acknowledgment phase also serves another purpose. Other two-hop neighbor nodes that receive this RA packet get blocked from transmitting. Therefore, they do not disturb the transmission that is to take place in the reserved slots.

When more than two TNs are located nearby, it results in a deadlock condition. Such situations may occur when there is no common neighbor node present when the RNs transmit RR packets. Collisions are not reported in the next phase, and so each node claims success and becomes a TN. Deadlocks are of two types: isolated and non-isolated. An isolated deadlock is a condition where none of the deadlocked nodes is connected to any non-deadlocked node. In the nonisolated

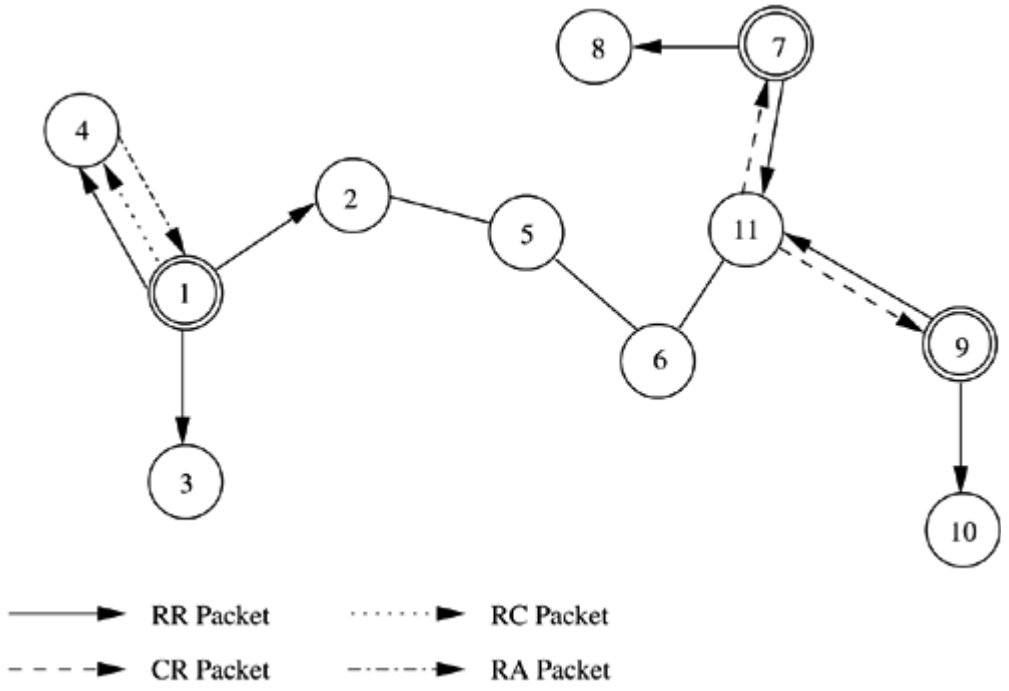
deadlock situation, at least one deadlocked node is connected to a non-deadlocked neighbor node. The RA phase can resolve isolated deadlocks. None of the nodes transmits RA, and hence the TNs abort their transmissions.

Packing/elimination (P/E) phase:

In this phase, a packing packet (PP) is sent by each node that is located within two hops from a TN, and that had made a reservation since the previous P/E phase. A node receiving a PPunderstands that there has been a recent success in slot reservation three hops away from it, and because of this some of its neighbors would have been blocked during this slot. The node

can take advantage of this and adjust its contention probability *p*, so that convergence is faster. In an attempt to resolve a non-isolated deadlock, each TN is required to transmit an elimination packet (EP) in this phase, with a probability 0.5. A deadlocked TN, on receiving an EP before transmitting its own EP, gets to know about the deadlock. It backs off by marking the slot as reserved and does not transmit further during the slot.

Consider Figure 6.22. Here nodes 1, 7, and 9 have packets ready to be transmitted to nodes 4, 8, and 10, respectively. During the reservation request phase, all three nodes transmit RRpackets. Since no other node in the two-hop neighborhood of node 1 transmits simultaneously, node 1 does not receive any CR message in the collision report phase. So node 1 transmits an RC message in the next phase, for which node 4 sends back an RAmessage, and the reservation is established. Node 7 and node 9 both transmit the RR packet in the reservation request phase. Here node 9 is within two hops from node 7. So if both nodes 7 and 9 transmit simultaneously, their RR packets collide at common neighbor node 11. Node 11 sends a CR packet which is heard by nodes 7 and 9. On receiving the CR packet, nodes 7 and 9 stop contending for the current slot.



6.6.6MACA with Piggy-Backed Reservation

MACA with piggy-backed reservation (MACA/PR) [15] is a protocol used to provide real-time

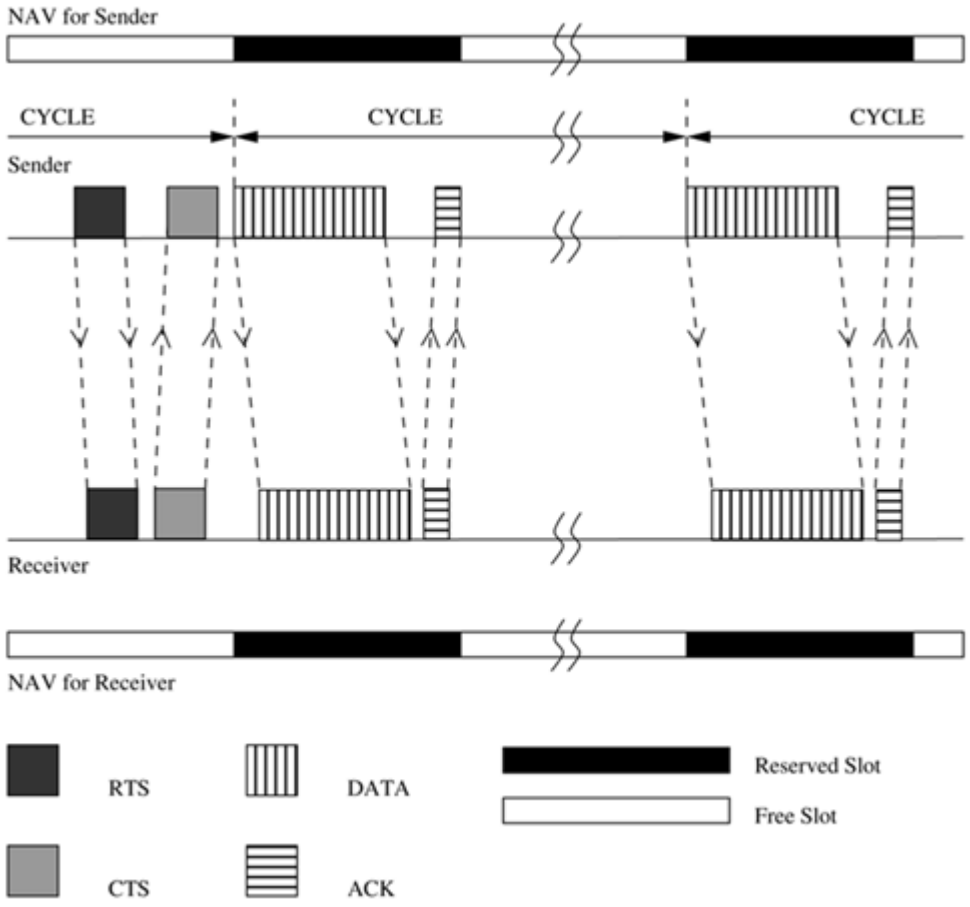
traffic support in multi-hop wireless networks. The MAC protocol used is based on the MACAW protocol [2], with the provisioning of non-persistent CSMA (as in FAMA [3]). The main components of MACA/PR are: a MAC protocol, a reservation protocol, and a QoS routing protocol. MACA/PR differentiates real-time packets from the best-effort packets. While providing guaranteed bandwidth support for real-time packets, at the same time it provides reliable transmission of best-effort packets. Time is divided into slots. The slots are defined by the reservations made at nodes, and hence are asynchronous in nature with varying lengths. Each node in the network maintains a reservation table (RT) that records all the reserved transmit and receive slots/windows of all nodes within its transmission range.

In order to transmit a non-real-time packet, a MACAW [2]-based MAC protocol is used. The ready node (a node which has packets ready for transmission) first waits for a free slot in the RT. Once it finds a free slot, it again waits for an additional random time of the order of a single-hop round-trip delay time, after which it senses the channel. If the channel is found to be still free, the node transmits an RTS packet, for which the receiver, if it is ready to receive packets, responds with a CTS packet. On receiving the CTS packet, the source node sends a DATA packet, and the receiver, on receiving the packet without any error, finally sends an ACK packet back to the source. The RTS and CTS control packets contain, in them, the time duration in which the DATA packet is to be transmitted. A nearby node that hears these packets avoids transmission during that time. If, after the random waiting time, the channel is found to be busy, the node waits for the channel to become idle again, and then repeats the same procedure.

For real-time traffic, the reservation protocol of MACA/PR functions as follows. The sender is assumed to transmit real-time packets at certain regular intervals, say, every CYCLE time period. The first data packet of the session is transmitted in the usual manner just as a best-effort packet would be transmitted. The source node first sends an RTS packet, for which the receiver node responds with a CTS packet. Now the source node sends the first DATA packet of the real-time session. Reservation information for the next DATA packet to be transmitted (which is scheduled to be transmitted after CYCLE time period) is piggy-backed on this current DATA packet. On receiving this DATA packet, the receiver node updates its reservation table with the piggy-backed reservation information. It then sends an ACK packet back to the source. The receiver node uses the ACK packet to confirm the reservation request that was piggy-backed on the previous DATA packet. It piggy-backs the reservation confirmation information on the ACK

packet. Neighbor nodes that hear the DATA and ACK packets update their reservation tables with the reservation information carried by them, and refrain from transmitting when the next packet is to be transmitted. Unlike MACAW [2],MACA/PR does not make use of RTS/CTS packets for transmission of the subsequent DATA packets. After receiving the ACK, the source node directly transmits the next DATA packet at its scheduled transmission time in the next CYCLE. This DATA packet in turn would carry reservation information for the next DATA packet. Real-time data transmission, hence, occurs as a series of DATA-ACK packet exchanges. The real-time packets (except for the first packet of the session that is used to initiate the reservation process) are transmitted only once. If an ACK packet is not received for a DATA packet, the source node just drops the packet. The ACK packet therefore serves the purpose of renewing the reservation, in addition to recovering from packet loss. If the source node fails to receive ACK packets for a certain number of consecutive DATA packets, it then assumes the reservation to have been lost. It restarts the real-time session again with an RTS-CTS control packet exchange, either on a different slot on the same link, or on a different link in case of a path break. In order to transmit an RTS to the receiver node, the source needs to find a slot that is free at both the nodes. For maintaining consistent information regarding free slots at all nodes, MACA/PRuses periodic exchange of reservation tables. This periodic table exchange automatically overcomes the hidden terminal problem. When a hidden terminal receives a reservation table from a node, it refrains from transmitting in the reserved slots of that node. Slot reservation information maintained in the reservation tables is refreshed every cycle. If the reservation is not refreshed for a certain number of consecutive cycles, it is then dropped. The transmission of packets in MACA/PR is depicted in Figure 6.23. It can be seen from the figure that theRTS-CTS exchange is used only for transmitting the first packet of the session. Since each DATA packet carries reservation information for the next DATA packet that would be transmitted after CYCLE time period, RTS-CTS exchange is not required for the subsequent DATA packets. Neighbor nodes that receive DATA packets update their reservation tables accordingly and do not contend for the channel during the reserved slots. The network allocation vector (NAV) at each node reflects the current and future state of the channel as perceived by the node.

Packet transmission in MACA/PR.



with higher priority being given to real-time packets. For real-time packets, MACA/PReffectively works as a TDM system, with a superframe time of CYCLE. The best-effort packets are transmitted in the empty slots (which have not been reserved) of the cycle. When a new node joins the network, it initially remains in the listening mode during which it receives reservation tables from each of its neighbors and learns about the reservations made in the network. After this initial period, the node shifts to its normal mode of operation.

The QoS routing protocol used with MACA/PR is the destination sequenced distance vector (DSDV) routing protocol [16] (described in detail in Chapter 7). Bandwidth constraint has been introduced in the routing process. Each node periodically broadcasts to its neighbors the (bandwidth, hop distance) pairs for each preferred path, that is, for each bandwidth value, to each destination. The number of preferred paths is equal to the maximum number of slots in a cycle.

After this is done, if a node receives a real-time packet with a certain bandwidth requirement that cannot be satisfied using the current available paths, the packet is dropped and no ACK packet is sent. The sender node would eventually reroute the packet.

Thus, MACA/PR is an efficient bandwidth reservation protocol that can support real-time traffic sessions. One of the important advantages of MACA/PR is that it does not require global synchronization among nodes. A drawback of MACA/PR is that a free slot can be reserved only if it can fit in the entire RTS-CTS-DATA-ACK exchange. Therefore, there is a possibility of many fragmented free slots not being used at all, reducing the bandwidth efficiency of the protocol.

6.6.7 Real-Time Medium Access Control Protocol

The real-time medium access control protocol (RTMAC) [17] provides a bandwidth reservation mechanism for supporting real-time traffic in ad hoc wireless networks. RTMACconsists of two components, a MAC layer protocol and a QoS routing protocol. The MAClayer protocol is a real-time extension of the IEEE 802.11 DCF. The QoS routing protocol is responsible for end-toend

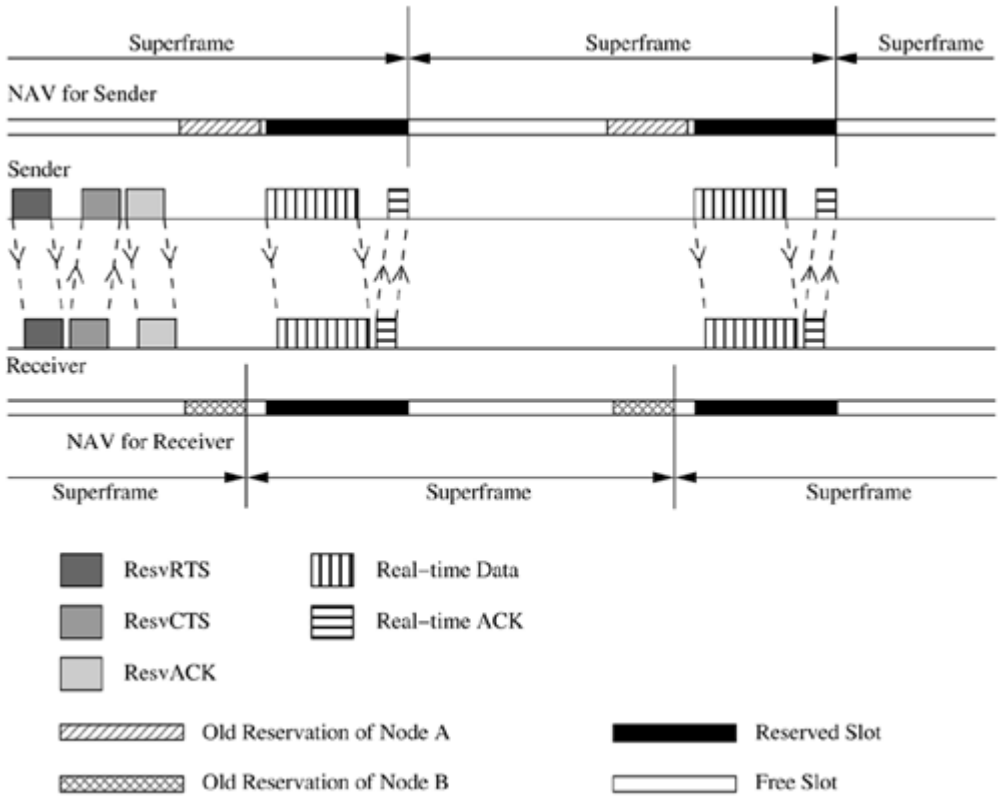
reservation and release of bandwidth resources. The MAC layer protocol has two parts: a medium-access protocol for best-effort traffic and a reservation protocol for real-time traffic.

A separate set of control packets, consisting of *ResvRTS*, *ResvRTSResvCTS*, and *ResvACK*, is used for effecting bandwidth reservation for real-time packets. RTS, CTS, and ACK control packets are used for transmitting best-effort packets. In order to give higher priority for real-time packets, the wait time for transmitting a *ResvRTS* packet is reduced to half of DCF inter-frame space (DIFS), which is the wait time used for best-effort packets.

Time is divided into superframes. As can be seen from Figure 6.24, the superframe for each node may not strictly align with the other nodes. Bandwidth reservations can be made by a node by reserving variable-length time slots on superframes, which are sufficient enough to carry the traffic generated by the node. The core concept of RTMAC is the flexibility of slot placement in the superframe. Each superframe consists of a number of reservation-slots (resv-slots). The time duration of each resv-slot is twice the maximum propagation delay. Data transmission normally requires a block of resv-slots. A node that needs to transmit real-time packets first reserves a set of resv-slots. The set of resv-slots reserved by a node for a connection on a superframe is called a

connection-slot. A node that has made reservations on the current superframe makes use of the same connection-slot in the successive superframes for transmitting packets. Each node maintains a reservation table containing information such as the sender id, receiver id, and starting and ending times of reservations that are currently active within its direct transmission range.

Reservation mechanism in RTMAC.



In RTMAC, no time synchronization is assumed. The protocol uses relative time for all reservation purposes. When a node receives this relative-time-based information, it converts the relative time to absolute time by adding its current time maintained in its clock. A three-way handshake protocol is used for effecting the reservation. For example, node A, which wants to reserve a slot with node B, sends a *ResvRTS* packet which contains the relative time information of starting and ending of the connection-slot (a number of resv-slots) to be reserved. Node B, on

receiving this packet, first checks its reservation table to see whether it can receive on those resvslots.

If it can, it replies by sending a *ResvCTS* packet containing the relative time information of the same resv-slots to be reserved. Neighbor nodes of the receiver, on receiving the *ResvCTS*, update their reservation tables accordingly. Source node A, on receiving the *ResvCTS* packet, responds by sending a *ResvACK* packet. This packet also carries relative time information regarding the reserved slots. The *ResvACK* packet is meant for the neighbor nodes of the source node (node A) which are not aware of the reservation as they may not receive the *ResvCTS* packet. Such nodes update their reservation tables on receiving the *ResvACK* packet. Transmission of the *ResvACK* packet completes the reservation process. Once the reservation is made, real-time packets are transmitted in these reserved slots. Transmission of each real-time packet is followed by the transmission of a real-time ACK (RTACK) packet by the receiver.

The bandwidth reservation process is illustrated in Figure 6.24. In the figure, NAV indicates the network allocation vector maintained at each node. As mentioned earlier in Section 6.6.6, the NAV at a node reflects the current and future state of the channel as perceived by the node. The sender node first transmits a *ResvRTS* packet indicating the connection-slot (represented by the offset time from the current time for the beginning and end of the connection-slot) to be reserved. The receiver node on receiving this packet checks its NAVand finds that the requested connection-slot is free. So it responds by sending a *ResvCTS*packet carrying the same connection-slot information. The sender node, on receiving this packet, completes the reservation process by sending a *ResvACK* packet. The corresponding connection-slot is marked as reserved at both the sender and the receiver. This is indicated inFigure 6.24 by the dark-shaded regions in the NAVs of the sender and receiver. Once the reservation is made, the real-time session gets started, and packets are transmitted in the reserved connection-slot by means of *Real-timeData – Real-timeACK* exchanges.

If the receiver node receives the *ResvRTS* packet on a slot which has already been reserved by one of its neighbor nodes, it does not respond with a *ResvCTS* packet. It just discards the received *ResvRTS* packet. This is because, if the node responds with a negative or positive ACK, the ACK packet may cause collisions with the reservations made by its neighbor. The sender node times out and retries later. In case the *ResvRTS* is received on a free slot, but the requested

connection-slot is not free at the receiver node, the receiver sends a negative CTS(*ResvNCTS*) back to the sender node. On receiving this, the sender node reattempts following the same procedure but with another free connection-slot.

If the real-time session gets finished, or a route break is detected by the sender node, the node releases the resources reserved for that session by sending a reservation release RTS(*ResvRelRTS*) packet. The *ResvRelRTS* packet is a broadcast packet. Nodes hearing this packet update their reservation tables in order to free the corresponding connection slots. In case the receiver node receives this (*ResvRelRTS*) packet, it responds by broadcasting a (*ResvRelCTS*) packet. The receiver's neighbor nodes, on receiving this (*ResvRelCTS*) packet, free up the corresponding reservation slots. In case the downstream node of the broken link does not receive the *ResvRelRTS* packet, since it also does not receive any DATA packet belonging to the corresponding connection, it times out and releases the reservations made.

A QoS routing protocol is used with RTMAC to find an end-to-end path that matches the QoS requirements (bandwidth requirements) of the user. The QoS routing protocol used here is an extension of the destination sequenced distance vector (DSDV) routing protocol. When a node receives a data packet for a new connection, the node reserves bandwidth on the forward link and forwards the packet to the next node on the path to the destination. In order to maintain a consistent view of reservation tables of neighboring nodes at each node, each node transmits its reservation information along with the route update packet, which is defined as part of the DSDV protocol. The routing protocol can specify a specific connection-slot to be reserved for a particular connection; this gives flexibility for the routing protocol to decide on the positioning of the connection-slot. But generally, the first available connection-slot is used.

One of the main advantages of RTMAC is its bandwidth efficiency. Since nodes operate in the asynchronous mode, successive reservation slots may not strictly align with each other. Hence small fragments of free slots may occur in between reservation slots. If the free slot is just enough to accommodate a DATA and ACK packet, then RTMAC can make use of the free slot by transmitting *ResvRTS-ResvCTS-ResvACK* in some other free slot.

CONTENTION-BASED MAC PROTOCOLS WITH SCHEDULING MECHANISMS

Protocols that fall under this category focus on packet scheduling at the nodes and transmission

scheduling of the nodes. Scheduling decisions may take into consideration various factors such as delay targets of packets, laxities of packets, traffic load at nodes, and remaining battery power at nodes. In this section, some of the scheduling-based MACprotocols are described.

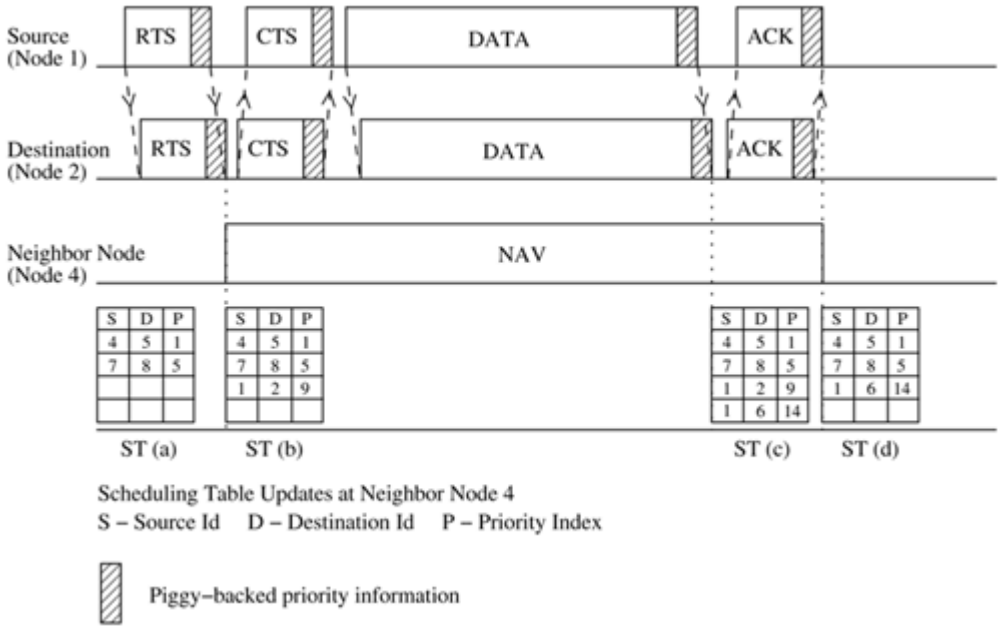
6.7.1 Distributed Priority Scheduling and Medium Access in Ad Hoc Networks

The distributed priority scheduling scheme (DPS) is based on the IEEE 802.11 distributed coordination function. DPS uses the same basic RTS-CTS-DATA-ACK packet exchange mechanism. The RTS packet transmitted by a ready node carries the priority tag/priority index for the current DATA packet to be transmitted. The priority tag can be the delay target for the DATA packet. On receiving the RTS packet, the intended receiver node responds with a CTS packet. The receiver node copies the priority tag from the received RTS packet and piggybacks it along with the source node id, on the CTS packet. Neighbor nodes receiving theRTS or CTS packets (including the hidden nodes) retrieve the piggy-backed priority tag information and make a corresponding entry for the packet to be transmitted, in their scheduling tables (STs). Each node maintains an ST holding information about packets, which were originally piggy-backed on control and data packets. The entries in the ST are ordered according to their priority tag values. When the source node transmits a DATA packet, its head-of-line packet information (consisting of the destination and source ids along with the priority tag) is piggy-backed on the DATA packet (head-of-line packet of a node refers to the packet to be transmitted next by the node). This information is copied by the receiver onto the ACK packet it sends in response to the received DATA packet. Neighbor nodes receiving the DATA or ACK packets retrieve the piggy-backed information and update their STs accordingly. When a node hears an ACK packet, it removes from its ST any entry made earlier for the corresponding DATA packet.

Figure 6.25 illustrates the piggy-backing and table update mechanism. Node 1 needs to transmit a DATA packet (with priority index value 9) to node 2. It first transmits an RTSpacket carrying piggy-backed information about this DATA packet. The initial state of the STof node 4 which is a neighbor of nodes 1 and 2 is shown in ST (a). Node 4, on hearing thisRTS packet, retrieves the piggybacked priority information and makes a corresponding entry in its ST, as shown in ST (b). The destination node 2 responds by sending a CTS packet. The actual DATA packet is sent by

the source node once it receives the CTS packet. This DATA packet carries piggy-backed priority information regarding the head-of-line packet at node 1. On hearing this DATA packet, neighbor node 4 makes a corresponding entry for the head-of-line packet of node 1, in its ST. ST (c) shows the new updated status of the ST at node 4. Finally, the receiver node sends an ACK packet to node 1. When this packet is heard by node 4, it removes the entry made for the corresponding DATA packet from its ST. The state of the scheduling table at the end of this data transfer session is depicted in ST (d).

Piggy-backing and scheduling table update mechanism in DPS



In essence, each node's scheduling table gives the rank of the node with respect to other nodes in its neighborhood. This rank information is used to determine the back-off period to be taken by the node. The back-off distribution is given by

$$\text{back} - \text{off} = \begin{cases} Uniform[0, (2^t CW_{min}) - 1] & r = 1, n < nmax \\ \alpha \times CW_{min} + Uniform[0, \gamma CW_{min} - 1] & r > 1, n = 0 \\ Uniform[0, (2^n \gamma CW_{min}) - 1] & r > 1, n \geq 1 \end{cases}$$

6.7.2 Distributed Wireless Ordering Protocol

The distributed wireless ordering protocol (DWOP) [19] consists of a media access scheme along with a scheduling mechanism. It is based on the distributed priority scheduling scheme

proposed in [18]. DWOP ensures that packets access the medium according to the order specified by an ideal reference scheduler such as first-in-first-out (FIFO), virtual clock, or earliest deadline first. In this discussion, FIFO is chosen as the reference scheduler. In FIFO, packet priority indices are set to the arrival times of packets. Similar to DPS [18], control packets are used in DWOP to piggy-back priority information regarding head-of-line packets of nodes. As the targeted FIFO schedule would transmit packets in order of the arrival times, each node builds up a scheduling table (ST) ordered according to the overheard arrival times.

The key concept in DWOP is that a node is made eligible to contend for the channel only if its locally queued packet has a smaller arrival time compared to all other arrival times in its ST(all other packets queued at its neighbor nodes), that is, only if the node finds that it holds the next region-wise packet in the hypothetical FIFO schedule. Two additional table management techniques, receiver participation and stale entry elimination, are used in order to keep the actual schedule close to the reference FIFO schedule. DWOP may not suffer due to information asymmetry. Since in most networks all nodes are not within the radio range of each other, a transmitting node might not be aware of the arrival times of packets queued atanother node which is not within its direct transmission range. This information asymmetry might affect the fair sharing of bandwidth. For example, in Figure 6.26 (a), the sender of flow B would be aware of the packets to be transmitted by the sender of flow A, and so it defers its transmission whenever a higher priority packet is queued at the sender of flow A. But the sender of flow A is not aware of the arrival times of packets queued at the sender of flow B and hence it concludes that it has the highest priority packet in its neighborhood. Therefore, node 1 unsuccessfully tries to gain access to the channel continuously. This would result in flow B receiving an unfair higher share of the available bandwidth. In order to overcome this information asymmetry problem, the *receiver participation* mechanism is used.

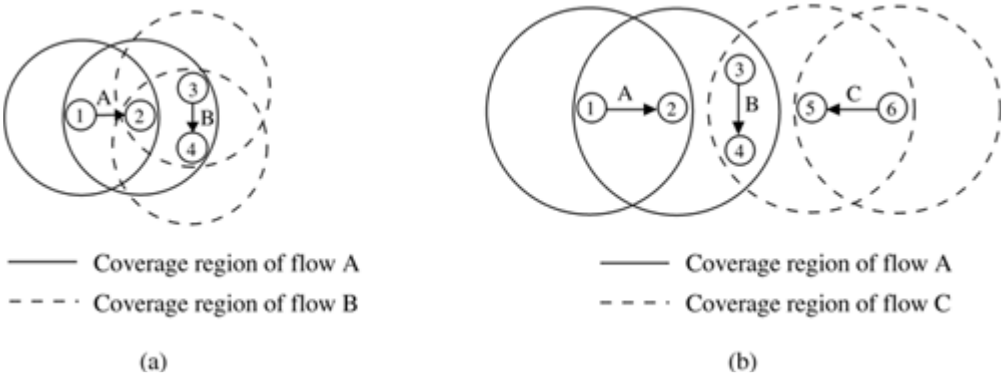


Figure 6.26. (a) Information asymmetry. (b) Perceived collisions.

In the receiver participation mechanism, a receiver node, when using its ST information, finds that the sender is transmitting out of order, that is, the reference FIFO schedule is being violated, an *out-of-order notification* is piggy-backed by the receiver on the control packets (CTS/ACK) it sends to the sender. In essence, information regarding the transmissions taking place in the twohop
neighborhood of the sender is propagated by the receiver node whenever it detects
a FIFO schedule violation. Since the notification is sent only when a FIFOviolation is detected, the actual transmission may not strictly follow the FIFO schedule; rather, it approximates
the FIFO schedule. On receiving an out-of-order packet from a sender node, the receiver node transmits a notification to the sender node carrying the actual rank *R*of the sender with respect to the receiver's local ST. On receiving this out-of-order notification, the sender node goes into a back-off state after completing the transmission of its current packet

6.7.3 Distributed Laxity-Based Priority Scheduling Scheme

The distributed laxity-based priority scheduling (DLPS) scheme [20] is a packet scheduling scheme, where scheduling decisions are made taking into consideration the states of neighboring nodes and the feedback from destination nodes regarding packet losses. Packets are reordered based on their uniform laxity budgets (ULBs) and the packet delivery ratios of the flows to which they belong.

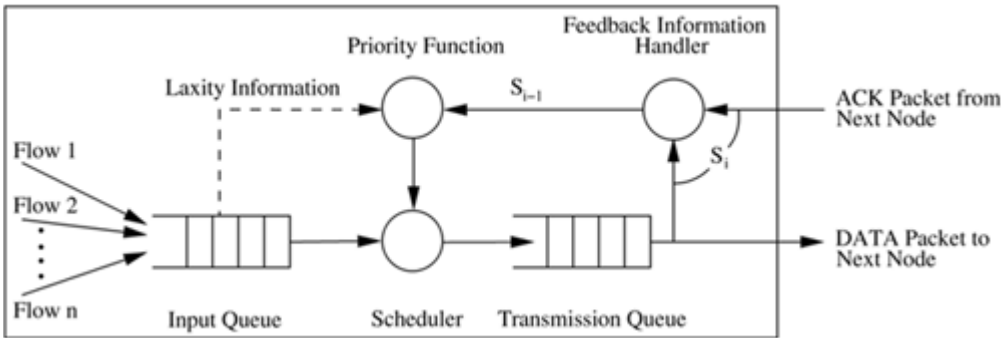
Each node maintains two tables: scheduling table (ST) and packet delivery ratio table (PDT). The ST contains information about packets to be transmitted by the node and packets overheard by the node, sorted according to their *priority index* values. Priority index expresses the priority of a packet. The lower the priority index, the higher the packet's priority. The PDT contains the

count of packets transmitted and the count of acknowledgment (ACK) packets received for every flow passing through the node. This information is used for calculating current packet delivery ratio of flows (explained later in this section).

A node keeps track of packet delivery ratios (used for calculating priority index of packets) of all flows it is aware of by means of a feedback mechanism. Figure 6.27 depicts the overall functioning of the feedback mechanism. Incoming packets to a node are queued in the node's input queue according to their arrival times. The scheduler sorts them according to their priority values and inserts them into the transmission queue. The highest priority packet from this queue is selected for transmission. The node, after transmitting a packet, updates the count of packets transmitted so far in its PDT. The destination node of a flow, on receiving data packets, initiates a feedback by means of which the count of DATA packets received by it is conveyed to the source through ACK packets traversing the reverse path. These two pieces of information, together denoted by *Si* in Figure 6.27, are received by the feedback information handler (FIH). The FIH, in parallel, also sends the previous state information *Si* −1 to the priority function module

(PFM). The ULB of each packet in ST is available at the node (ULB calculation will be explained later in this section). This information is also sent to PFM, which uses the information fed to it to calculate the priority indices of packets in the ST.

Feedback mechanism.



6.8 MAC PROTOCOLS THAT USE DIRECTIONAL ANTENNAS

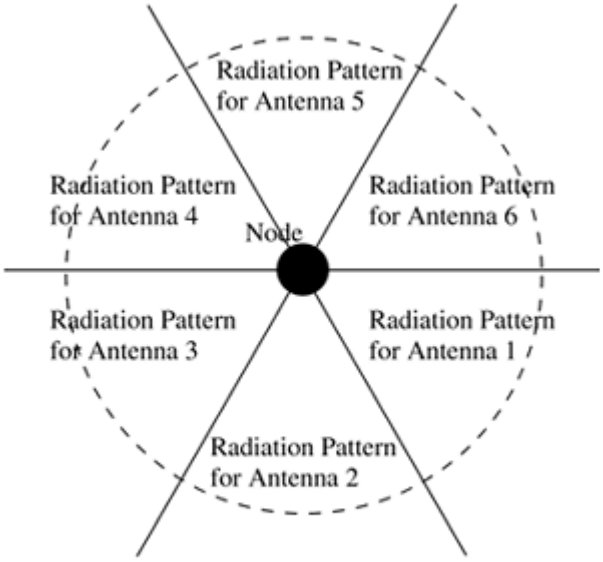
MAC protocols that use directional antennas for transmissions have several advantages over

those that use omnidirectional transmissions. The advantages include reduced signal interference, increase in the system throughput, and improved channel reuse that leads to an increase in the overall capacity of the channel. In this section, some of the MAC layer protocols that make use of directional antennas are discussed

6.8.1MAC Protocol Using Directional Antennas

The MAC protocol for mobile ad hoc networks using directional antennas that was proposed in [21] makes use of directional antennas to improve the throughput in ad hoc wireless networks. The mobile nodes do not have any location information by means of which the direction of the receiver and sender nodes could be determined. The protocol makes use of anRTS/CTS exchange mechanism, which is similar to the one used in MACA [1]. The nodes use directional antennas for transmitting and receiving data packets, thereby reducing their interference to other neighbor nodes. This leads to an increase in the throughput of the system. Each node is assumed to have only one radio transceiver, which can transmit and receive only one packet at any given time. The transceiver is assumed to be equipped with *M* directional antennas, each antenna having a conical radiation pattern, spanning an angle of radians (Figure 6.28). It is assumed that the transmissions by adjacent antennas never overlap, that is, the complete attenuation of the transmitted signal occurs outside the conical pattern of the directional antenna. The MAC protocol is assumed to be able to switch every antenna individually or all the antennas together to the *active* or *passive* modes. The radio transceiver uses only the antennas that are in the active mode. If a node transmits when all its antennas are active, then the transmission's radiation pattern is similar to that of an omnidirectional antenna. The receiver node uses receiver diversity while receiving on all antennas. This means that the receiver node uses the signal from the antenna which receives the incoming signal at maximum power. In the normal case, this selected antenna would be the one whose conical pattern is directed toward the source node whose signal it is receiving. It is assumed that the radio range is the same for all directional antennas of the nodes. In order to detect the presence of a signal, a threshold signal power value is used. A node concludes that the channel is active only if the received signal strength is higher than this threshold value.

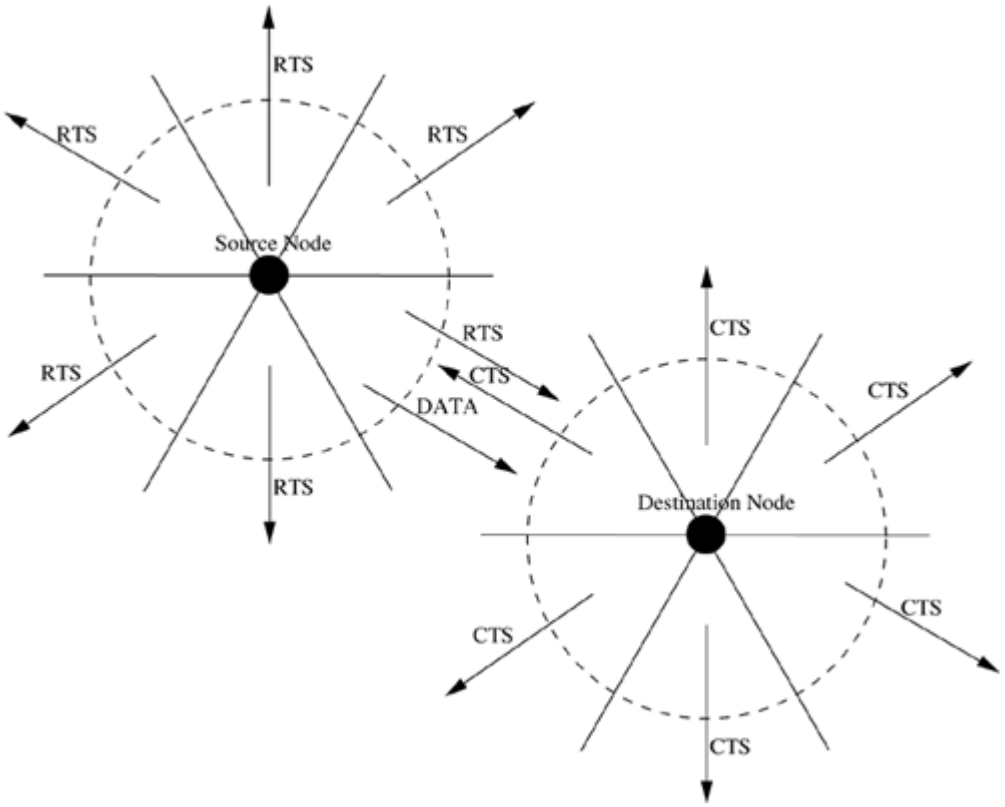
Radiation patterns of directional antennas.



Node with 6 antennas (M=6)

The protocol works as follows. The packet-exchange mechanism followed for transmitting each data packet is depicted in Figure 6.29. In the example shown in the figure, each node is assumed to have six directional antennas. The main concept in this protocol is the mechanism used by the transmitting and receiving nodes to determine the directions of each other. The MAC layer at the source node must be able to find the direction of the intended next-hop receiver node so that the data packet could be transmitted through a directional antenna. It is the same case with the receiver. The receiver node must be able to determine the direction of the sender node before starting to receive data packets. This is performed in the following manner. An idle node is assumed to be listening to the on-going transmissions on all its antennas. The sender node first transmits an RTS packet addressed to the receiver. This RTS is transmitted through all the antennas of the node (omnidirectional transmission). The intended receiver node, on receiving this RTS packet, responds by transmitting a CTSpacket, again on all its antennas (omnidirectional transmission). The receiver node also notes down the direction of the sender by identifying the antenna that received the RTS packet with maximum power. The source, on receiving the CTS packet, determines the direction of the receiver node in a similar manner. The neighbor nodes that receive the RTS or CTS packets defer their transmissions for appropriate periods of time. After receiving the CTS, the source node transmits the next data packet through the chosen directional antenna. All other antennas are switched off and remain idle. The receiver node receives this data packet only through its selected antenna.

Figure 6.29. Packet transmission.



Since a node transmits packets only through directional antennas, the interference caused to nodes in its direct transmission range is reduced considerably, which in turn leads to an increase in the overall throughput of the system

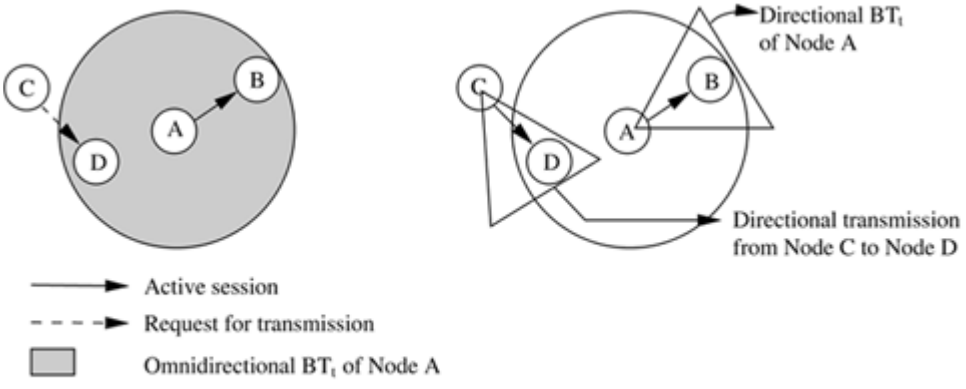
6.8.2 Directional Busy Tone-Based MACProtocol

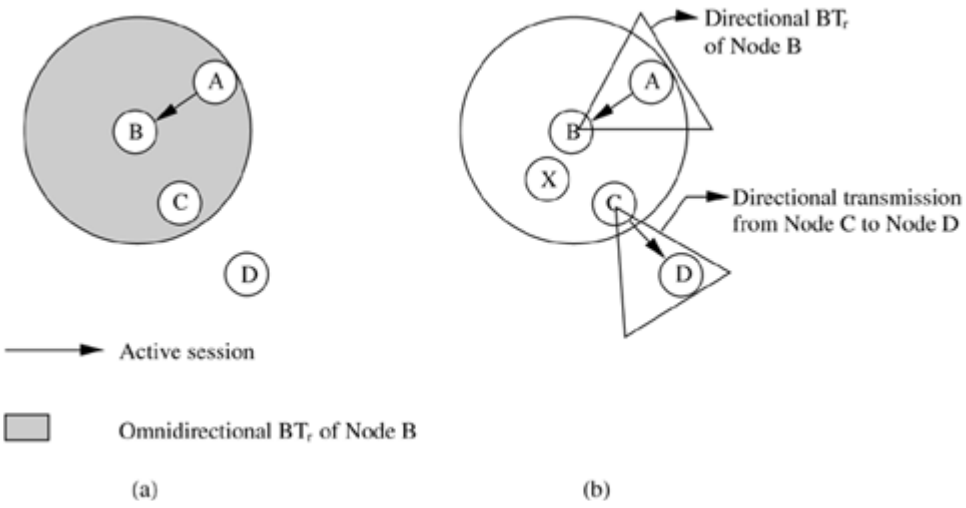
The directional busy tone-based MAC protocol [22] adapts the DBTMA protocol [5] for use with directional antennas. It uses directional antennas for transmitting the RTS, CTS, and data frames, as well as the busy tones. By doing so, collisions are reduced significantly. Also, spatial reuse of the channel improves, thereby increasing the capacity of the channel.

Each node has a directional antenna which consists of *N* antenna elements, each covering a fixed sector spanning an angle of (360/*N*) degrees. For a unicast transmission, only a single antenna element is used. For broadcast transmission, all the *N* antenna elements transmit simultaneously. When a node is idle (not transmitting packets), all antenna elements of the node keep sensing the channel. The node is assumed to be capable of identifying the antenna element on which the

incoming signal is received with maximum power. Therefore, while receiving, exactly one antenna element collects the signals. In an ad hoc wireless network, nodes may be mobile most of the time. It is assumed that the orientation of sectors of each antenna element remains fixed. The protocol uses the same two busy tones *BTt* and *BTr* used in the DBTMA protocol. The purpose of the busy tones is the same. Before transmitting anRTS packet, the sender makes sure that the *BTr* tone is not active in its neighborhood, so that its transmissions do not interfere with packets being received at a neighboring receiver node. Similarly, a receiver node, before transmitting a CTS, verifies that a *BTt* is not active in its neighborhood. This is done to make sure that the data the node is expected to receive does not collide with any other on-going transmission. The modified directional DBTMA protocol operates as follows.

A node that receives a data packet for transmission first transmits an RTS destined to the intended receiver in all directions (omnidirectional transmission). On receiving this RTS, the receiver node determines the antenna element on which the RTS is received with maximum gain. The node then sends back a directional CTS to the source using the selected antenna element (which points toward the direction of the sender). It also turns on the busy tone *BTr* in the direction toward the sender. On receiving the CTS packet, the sender node turns on the*BTt* busy tone in the direction of the receiver node. It then starts transmitting the data packet through the antenna element on which the previous CTS packet was received with maximum gain. Once the packet transmission is over, it turns off the *BTt* signal. The receiver node, after receiving the data packet, turns off the *BTr* signal.





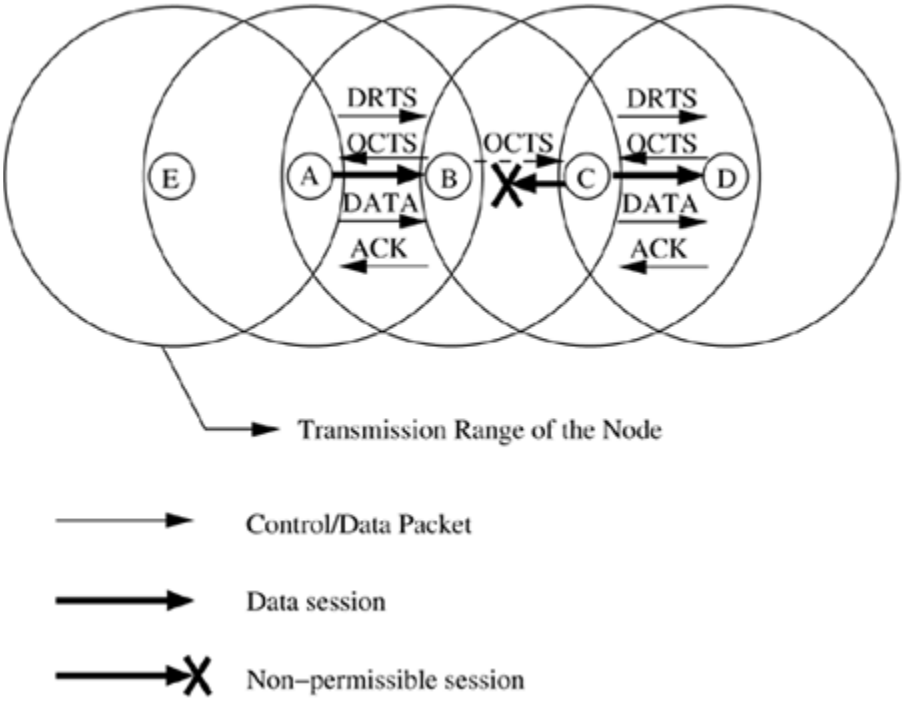
6.8.3 Directional MAC Protocols for Ad Hoc Wireless Networks

Two MAC schemes using directional antennas are proposed in [23]. It is assumed that each node knows about the location of its neighbors as well as its own location. The physical location information can be obtained by a node using the global positioning system (GPS). In the IEEE 802.11 DCF scheme, a node that is aware of a nearby on-going transmission will not participate in a transmission itself. In the directional MAC (D-MAC) protocols proposed in [23], a similar logic is applied on a per-antenna basis. If a node has received an RTS or CTSpacket related to an on-going transmission on a particular antenna, then that particular antenna is not used by the node till the other transmission is completed. This antenna stays blocked for the duration of that transmission. The key concept here is, though a particular antenna of a node may remain blocked, the remaining antennas of the node can be used for transmissions. This improves the throughput of the system. An omnidirectional transmission is possible only if none of the antennas of the node is blocked.

In the first directional MAC scheme (DMAC-1), a directional antenna is used for transmittingRTS packets. CTS packet transmissions are omnidirectional. Consider Figure 6.32. Here node A, which needs to transmit a packet to node B, first transmits a directional RTS (DRTS) packet to node B. Node B, on receiving this packet, responds by transmitting an omnidirectional CTS (OCTS) packet. Once the OCTS is received without any error by node A, node A sends a data packet using a directional antenna. When node B receives the data packet, it immediately transmits a directional ACK (DACK) packet. Node C would receive theOCTS packet from node B. At node C, only the directional antenna pointing toward

node B would be blocked due to this. Node C can freely transmit to node D using another directional antenna. Thus it can be seen that in DMAC-1, usage of directional antennas improves the performance by allowing simultaneous transmissions, which are not permitted when only omnidirectional antennas are used.

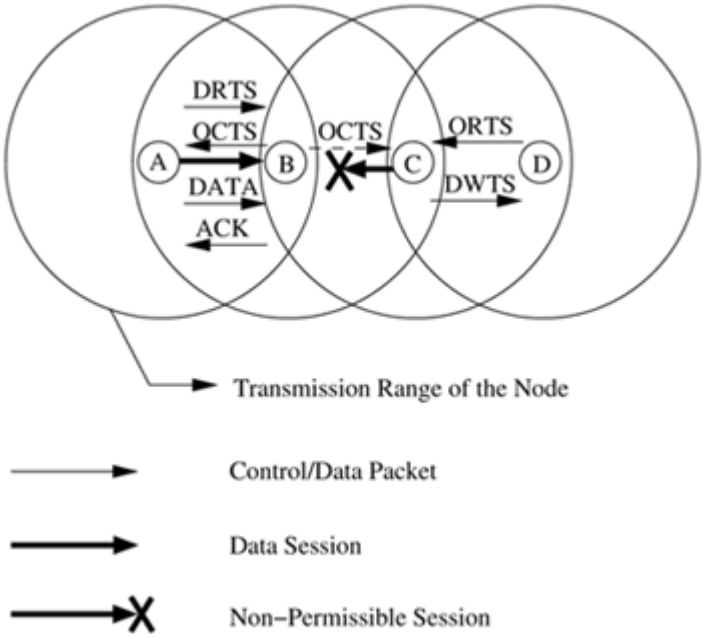
Operation of DMAC protocol.



In the second directional MAC scheme (DMAC-2) proposed in [23], both directional RTS(DRTS) as well as omnidirectional RTS (ORTS) transmissions are used. In DMAC-1, the usage of DRTS may increase the probability of control packet collisions. For example, consider Figure 6.32. Here node A initiates a DRTS transmission to node B. The DRTS packet is not heard by node E, and so it is not aware of the transmission between nodes A and B. Suppose node E sends a packet to node A, then that packet may collide with the OCTS orDACK packets transmitted by node B. The probability of control packet collisions is reduced in DMAC-2. In DMAC-2, a node that wants to initiate a data transfer may send an ORTS or aDRTS as per the following two rules. (1) If none of the directional antennas at the node are blocked, then the node sends an ORTS packet. (2) Otherwise, the node sends a DRTS packet, provided the desired directional antenna is not blocked. Consider the same example in Figure 6.32. Here when node A initiates data transfer to node B, assuming all its

antennas are not blocked, it sends an ORTS packet to node B. Node E would now receive this packet, and the antenna on which the ORTS packet was received would remain blocked for the duration of the transmission from node A to node B. If node E wants to send a packet to node A, it needs to wait for the duration of the transmission between nodes A and B, so that its directional antenna pointing toward node A becomes unblocked; only then can it start transmitting packets to node A. Thus, the combination of ORTS and DRTS packets in DMAC-2 reduces collisions between control packets.

Operation of DMAC protocol.



OTHER MAC PROTOCOLS

6.9.1 Multichannel MAC Protocol

The multichannel MAC protocol (MMAC) [24] uses multiple channels for data transmission. There is no dedicated control channel. *N* channels that have enough spectral separation between each other are available for data transmission.

Each node maintains a data structure called *PreferableChannelList* (PCL). The usage of the channels within the transmission range of the node is maintained in the PCL. Based on their usage, channels can be classified into three types.

• High preference channel (HIGH): The channel has been selected by the current node and is

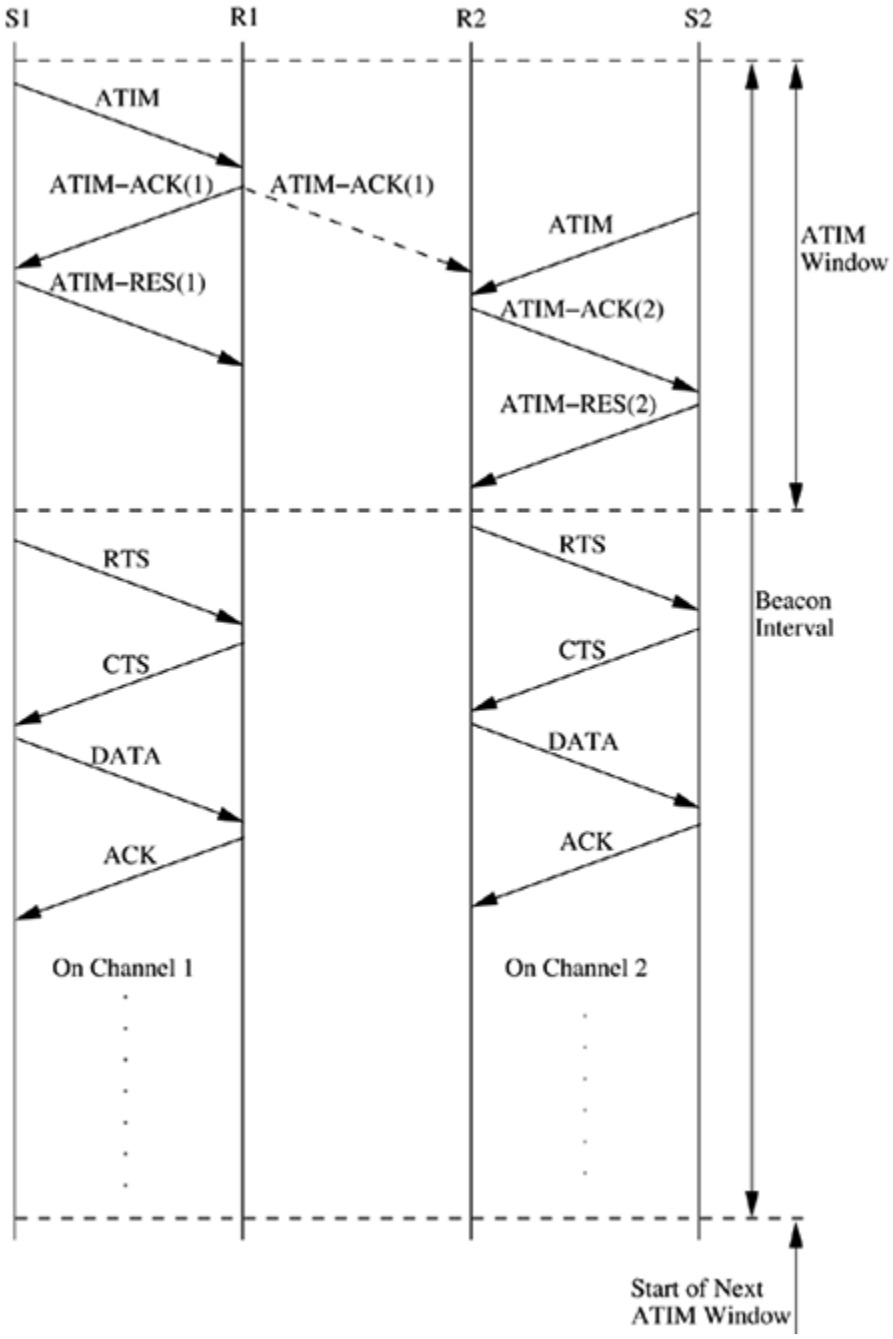
being used by the node in the current beacon interval (beacon interval mechanism will be explained later). Since a node has only one transceiver, there can be only one HIGH channel at a time.

• Medium preference channel (MID): A channel which is free and is not being currently used in the transmission range of the node is said to be a medium preference channel. If there is no HIGH channel available, a MID channel would get the next preference.

• Low preference channel (LOW): Such a channel is already being used in the transmission range of the node by other neighboring nodes. A counter is associated with each LOW state channel. For each LOW state channel, the count of source-destination pairs which have chosen the channel for data transmission in the current beacon interval is maintained.

Time is divided into beacon intervals and every node is synchronized by periodic beacon transmissions. So, for every node, the beacon interval starts and ends almost at the same time

Operation of MMAC protocol.



6.9.2 Multichannel CSMAMAC Protocol

In the multichannel CSMA MAC protocol (MCSMA) [25], the available bandwidth is divided into several channels. A node with a packet to be transmitted selects an idle channel randomly. The protocol also employs the notion of *soft* channel reservation, where preference is given to

the channel that was used for the previous successful transmission. Though the principle used in MCSMA is similar to the frequency division multiple access (FDMA) schemes used in cellular networks, the major difference here is that there is no centralized infrastructure available, and channel assignment is done in a distributed fashion using carrier-sensing. The operation of the protocol is discussed below.

The total available bandwidth is divided into *N* non-overlapping channels (*N* is independent of the number of hosts in the network), each having a bandwidth of (*W/N*), where *W* is the total bandwidth available for communication. The channels may be created in the frequency domain (FDMA) or in the code domain (CDMA). Since global synchronization between nodes is not available in ad hoc wireless networks, channel division in the time domain (TDMA) is not used. An idle node (which is not transmitting packets) continuously monitors all the *N* channels. A channel whose total received signal strength1 (TRSS) is below the sensing threshold (ST) of the node is marked IDLE by the node. The time at which TRSS drops below ST is also noted for each IDLE channel. Such channels are put in the *free-channels* list.

6.9.3 Power Control MAC Protocol for Ad Hoc Networks

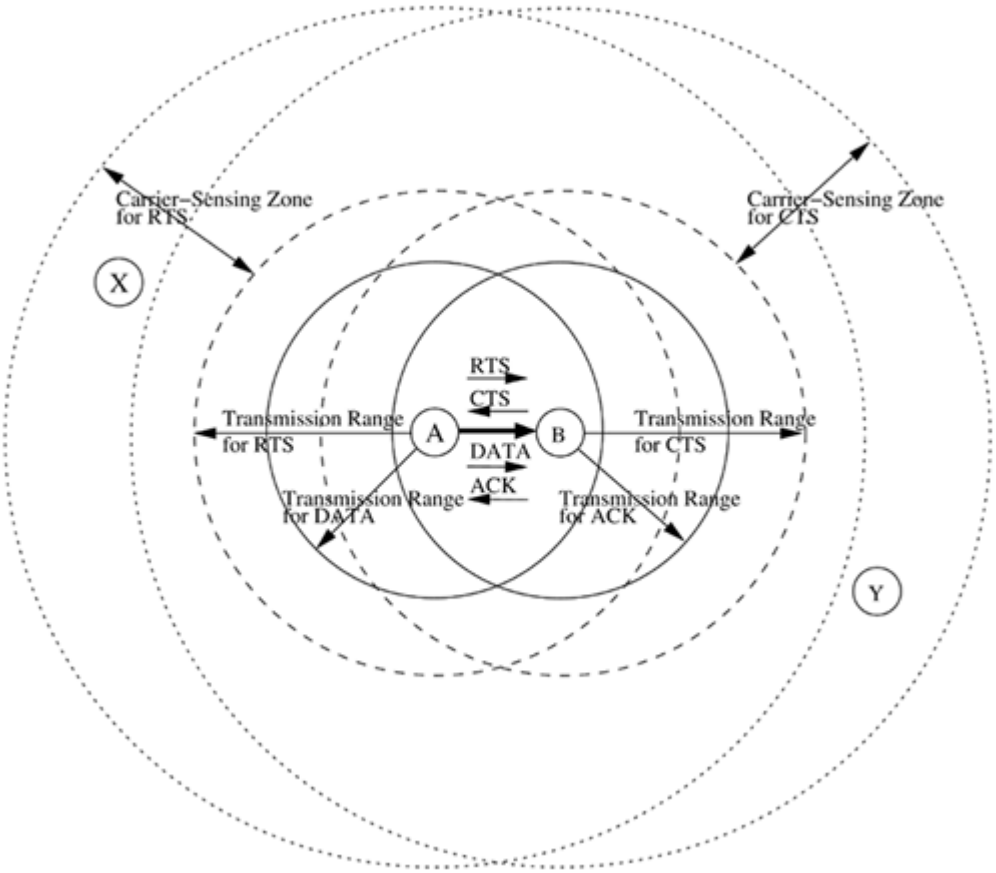
The power control MAC protocol (PCM) [26] allows nodes to vary their transmission power levels on a per-packet basis. The PCM protocol is based on the power control protocol used in [27], which is referred to as the *BASIC* protocol in this section. In what follows, the working of the BASIC power control protocol is briefly described. This is followed by a discussion of the PCM protocol.

In the BASIC scheme, the RTS and CTS packets are transmitted with maximum power *pmax*.The RTS-CTS handshake is used for deciding upon the transmission power for the subsequent DATA and ACK packet transmissions. This can be done using two methods. In the first method, source node A transmits the RTS with maximum power *pmax*. This RTS is received at the receiver with signal level *pr*. The receiver node B can calculate the minimum required transmission power level *pdesired* for the DATA packet, based on the received power level *pr* , the transmitted power level *pmax* , and the noise level at receiver B. Node B then specifies this *pdesired* in

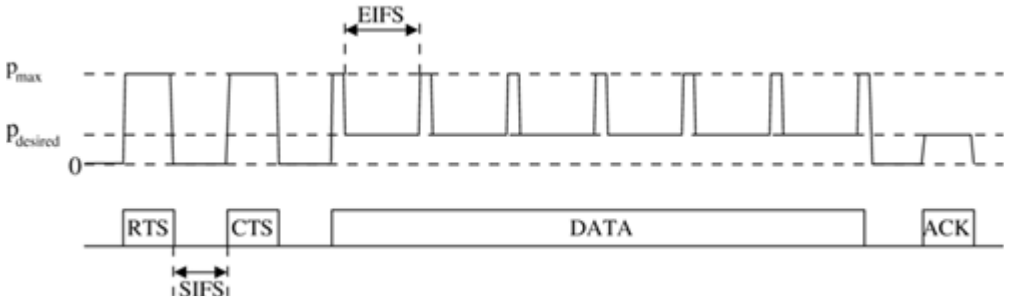
the CTS packet it transmits to node A. Node A transmits the DATA packet using power level *pdesired* . In the second method, when the receiver node B receives anRTS packet, it responds with a CTS packet at the usual maximum power level *pmax*. When the source node receives this CTS packet, it calculates *pdesired* based on the received power level*pr* and transmitted power level *pmax* as

$$p_{desired} = \frac{p_{max}}{p_r} \times Rx_{thresh} \times c$$

Packet transmission in BASIC scheme



Transmission power pattern in PCM.



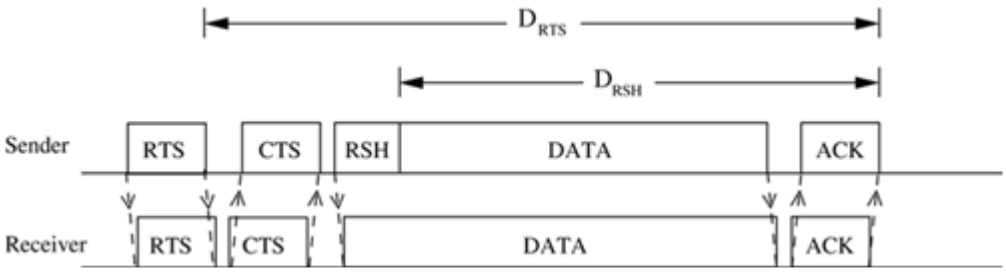
Hence with the above simple modification, the PCM protocol overcomes the problems faced in the BASIC scheme. PCM achieves throughput very close to that of the 802.11 protocol while using much less energy

6.9.4 Receiver-Based Autorate Protocol

The receiver-based autorate protocol (RBAR) [28] uses a novel rate adaptation approach. The rate adaptation mechanism is at the receiver node instead of being located at the sender. Rate adaptation is the process of dynamically switching data rates in order to match the channel conditions so that optimum throughput for the given channel conditions is achieved. Rate adaptation consists of two processes, namely, channel quality estimation and rate selection. The accuracy of the channel quality estimates significantly influences the effectiveness of the rate adaptation process. Therefore, it is important that the best available channel quality estimates are used for rate selection. Since it is the channel quality at the receiver node which determines whether a packet can be received or not, it can be concluded that the best channel quality estimates are available at the receiver. The estimates must be used as early as possible before they get stale. If the sender is to implement the rate adaptation process, significant delay would be involved in communicating the channel quality estimates from the receiver to the sender, which may result in the estimates becoming stale before being used. Therefore, the RBAR protocol advocates for rate adaptation at the receiver node rather than at the sender. Rate selection is done at the receiver on a per-packet basis during the RTS-CTS packet exchange. Since rate selection is done *during* the RTS-CTS exchange, the channel quality estimates are very close to the actual transmission times of the data packets. This improves the effectiveness of the rate selection process. The RTS and CTS packets carry the chosen modulation rate and the size of the data packet, instead of carrying the duration of the reservation. The packet transmission process is depicted in Figure 6.37. The sender node chooses

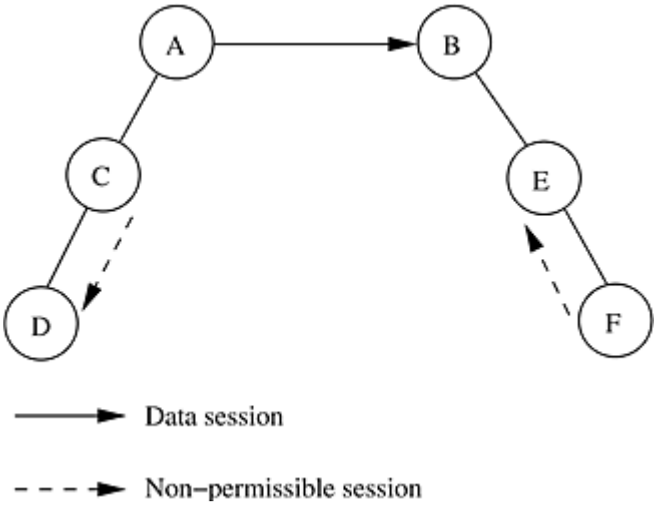
a data rate based on some heuristic and inserts the chosen data rate and the size of the data packet into the RTS. When a neighbor node receives this RTS, it calculates the duration of the reservation *DRTS* using the data rate and packet size carried on the RTS. The neighbor node then updates its NAV accordingly to reflect the reservation. While receiving the data packet, the receiver node generates an estimate of the channel conditions for the impending data transfer. Based on this estimate, it chooses an appropriate data rate. It stores the chosen data rate and the size of the packet on the CTS packet and transmits the CTS to the sender. Neighbor nodes receiving the CTS calculate the expected duration of the transmission and update their NAVs accordingly. The source node, on receiving the CTSpacket, responds by transmitting the data packet at the rate chosen by the receiver node.

Packet transmission in RBAR.

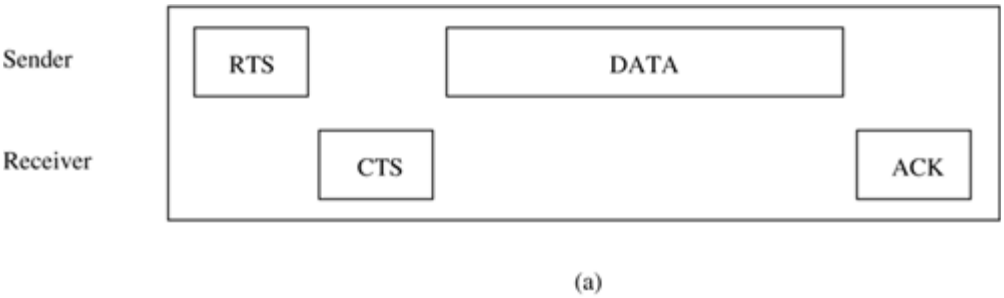


6.9.5 Interleaved Carrier-Sense Multiple Access Protocol

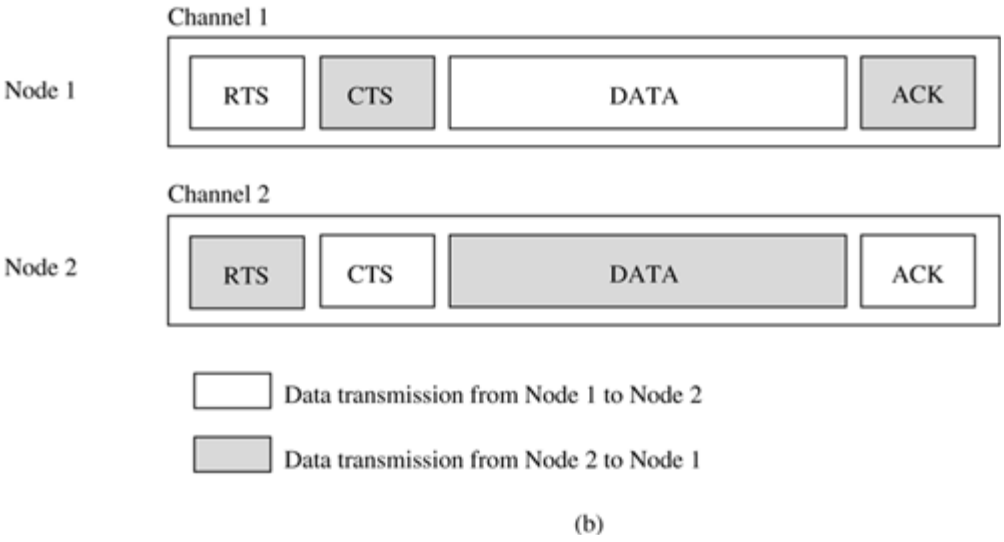
The interleaved carrier-sense multiple access protocol (ICSMA) [29] efficiently overcomes the exposed terminal problem faced in ad hoc wireless networks. The inability of a source node to transmit, even though its transmission may not affect other ongoing transmissions, is referred to as the exposed terminal problem. For example, consider the topology shown inFigure 6.38. Here, when a transmission is going from node A to node B, nodes C and F would not be permitted to transmit to nodes D and E, respectively. Node C is called a sender-exposed node, and node E is called a receiver-exposed node. The exposed terminal problem reduces the bandwidth efficiency of the system.



Exposed terminal problem.



(a)



(b)

Packet transmissions in (a) 802.11 DCF and (b) ICSMA

UNIT 4

ROUTING PROTOCOLS FOR AD HOC WIRELESS NETWORKS

ISSUES IN DESIGNING A ROUTING PROTOCOL FOR AD HOC WIRELESS NETWORKS

The major challenges that a routing protocol designed for ad hoc wireless networks faces are mobility of nodes, resource constraints, error-prone channel state, and hidden and exposed terminal problems. A detailed discussion on each of the following is given below

1 Mobility

The network topology in an ad hoc wireless network is highly dynamic due to the movement of nodes, hence an on-going session suffers frequent path breaks. Disruption occurs either due to the movement of the intermediate nodes in the path or due to the movement of end nodes. Such situations do not arise because of reliable links in wired networks where all the nodes are stationary.

Even though the wired network protocols find alternate routes during path breaks, their convergence is very slow. Therefore, wired network routing protocols cannot be used in ad hoc wireless networks where the mobility of nodes results in frequently changing network topologies. Routing protocols for ad hoc wireless networks must be able to perform efficient and effective mobility management.

2 Bandwidth Constraint

Abundant bandwidth is available in wired networks due to the advent of fiber optics and due to the exploitation of wavelength division multiplexing (WDM) technologies. But in a wireless network, the radio band is limited, and hence the data rates it can offer are much less than what a wired network can offer. This requires that the routing protocols use the bandwidth optimally by keeping the overhead as low as possible. The limited bandwidth availability also imposes a constraint on routing protocols in maintaining the topological information. Due to the frequent changes in topology, maintaining a consistent topological information at all the nodes involves more control overhead which, in turn, results in more bandwidth wastage. As efficient routing protocols in wired networks require the complete topology information, they may not be suitable for routing in the ad hoc wireless networking environment.

3 Error-Prone Shared Broadcast Radio Channel

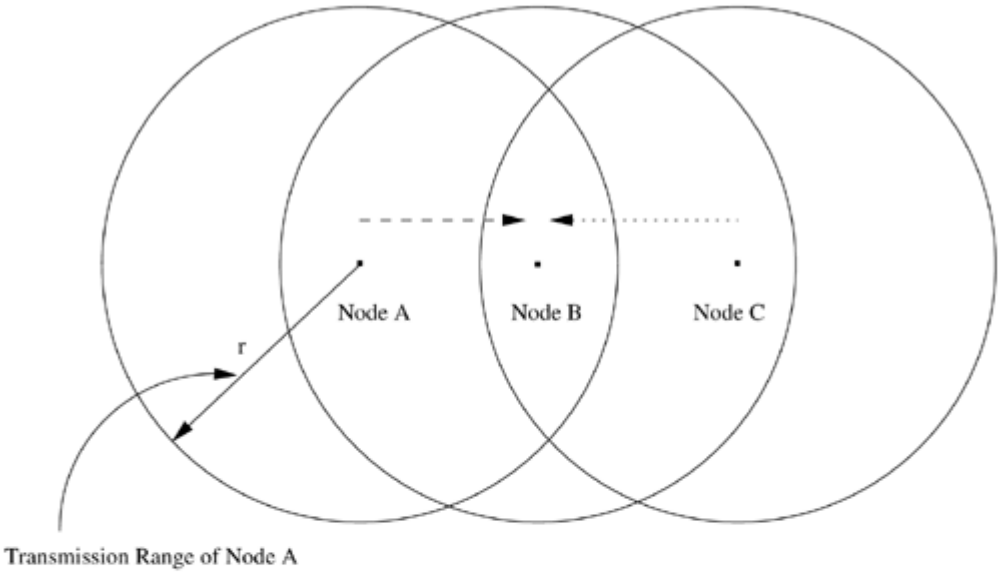
The broadcast nature of the radio channel poses a unique challenge in ad hoc wireless networks. The wireless links have time-varying characteristics in terms of link capacity and link-error probability. This requires that the ad hoc wireless network routing protocol interacts with the MAC layer to find alternate routes through better-quality links. Also, transmissions in ad hoc wireless networks result in collisions of data and control packets. This is attributed to the hidden terminal problem [1]. Therefore, it is required that ad hoc wireless network routing protocols find paths with less congestion.

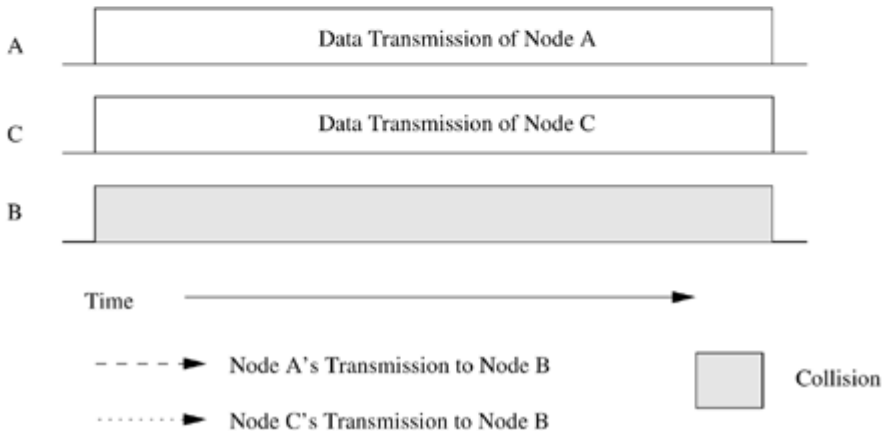
4 Hidden and Exposed Terminal Problems

The hidden terminal problem refers to the collision of packets at a receiving node due to the simultaneous transmission of those nodes that are not within the direct transmission range of the sender, but are within the transmission range of the receiver. Collision occurs when both nodes transmit packets at the same time without knowing about the transmission of each other. For example, consider Figure. Here, if both node A and node C transmit to node B at the same time, their packets collide at node B. This is due to the fact that both nodes A and C are hidden from each other, as they are not within the direct transmission range of each other and hence do not know about the presence of each other. Solutions for this problem include medium access collision avoidance (MACA) [2], medium access collision avoidance for wireless (MACAW) [3], floor acquisition multiple access (FAMA) [4], and dual busy tone multiple access (DBTMA) [5]. MACA requires that a transmitting node first explicitly notifies all potential hidden nodes about the forthcoming transmission by means of a two-way handshake control protocol called the RTS-CTS protocol exchange. Note that this may not solve the problem completely, but it reduces the probability of collisions. To increase the efficiency, an improved version of the MACA protocol known as MACAW [3] has been proposed. This protocol requires that the receiver acknowledges each successful reception of a data packet. Hence, successful transmission is a fourway exchange mechanism, namely, RTS-CTS-Data-ACK. Even in the absence of bit errors and mobility, the RTS-CTS control packet exchange cannot ensure collision-free data transmission that has no interference from hidden terminals. One very important assumption made is that every node in the capture area of the receiver (transmitter) receives the CTS (RTS) cleanly. Nodes that do not hear either of these clearly can disrupt the successful transmission of the Data or the ACK packet. One particularly troublesome situation

occurs when node A, hidden from the transmitter T and within the capture area of the receiver R, does not hear the CTS properly because it is within the capture area of node B that is transmitting and that is hidden from both R and T, as illustrated inFigure 7.2. In this case, node A did not successfully receive the CTS originated by node R and hence assumes that there is no on-going transmission in the neighborhood. Since node A is hidden from node T, any attempt to originate its own RTS would result in collision of the on-going transmission between nodes T and R.

Hidden terminal problem.

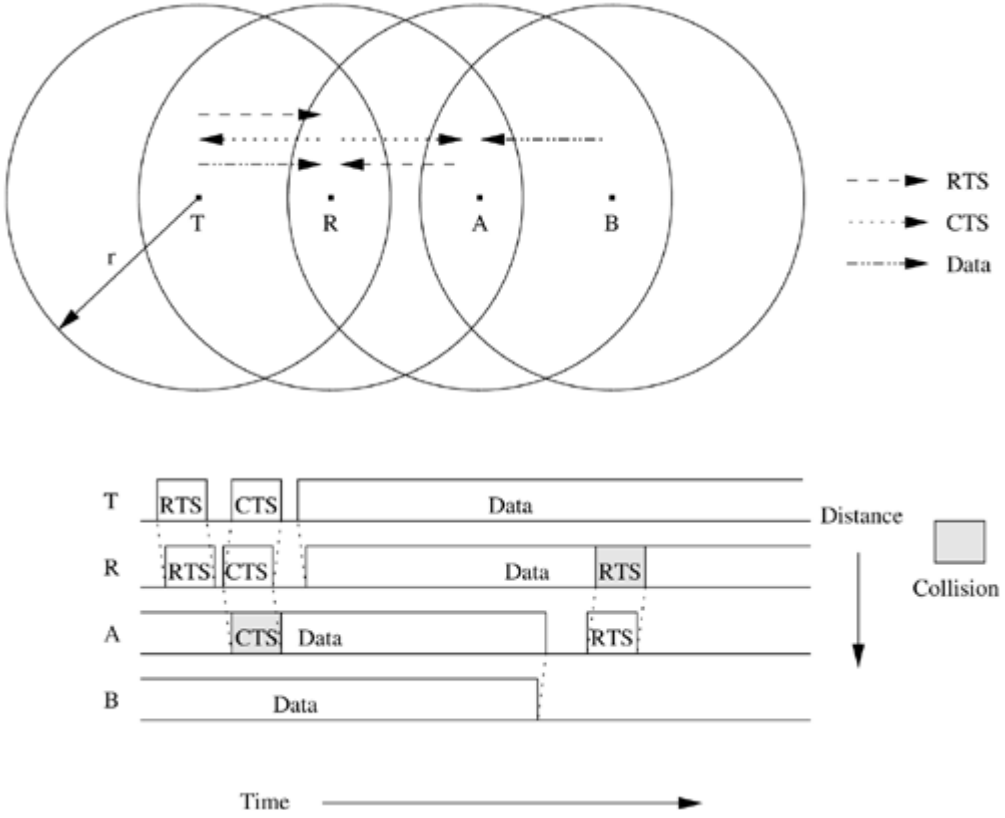




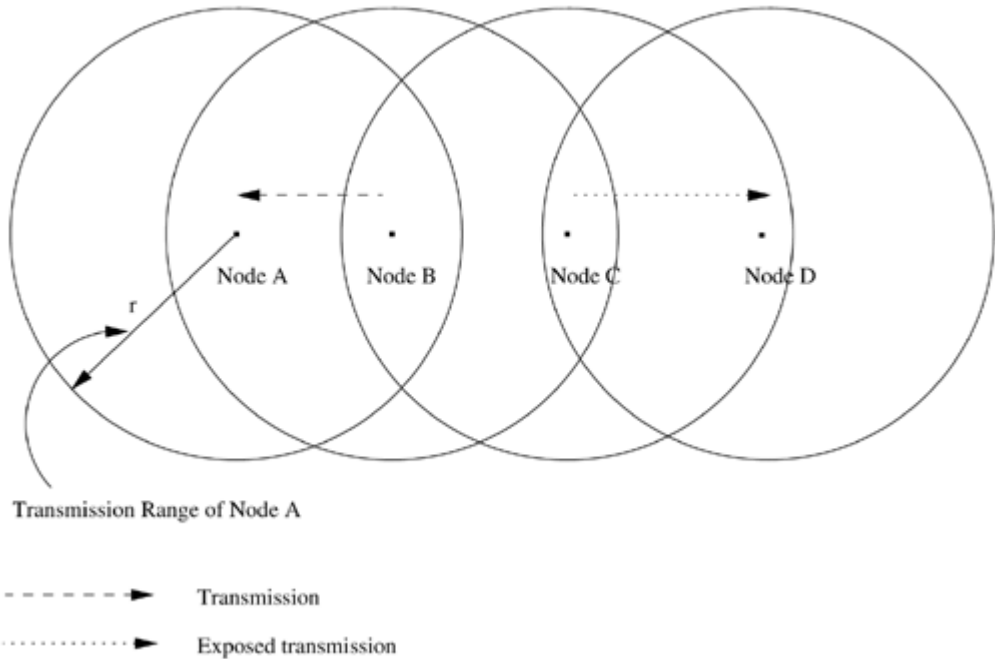
The exposed terminal problem refers to the inability of a node which is blocked due to transmission by a nearby transmitting node to transmit to another node. Consider the example in Figure 7.3. Here, if a transmission from node B to another node A is already in progress, node C cannot transmit to node D, as it concludes that its neighbor, node B, is in transmitting mode and hence should not interfere withthe on-going transmission. Thus, reusability of the radio spectrum

is affected. For node C to transmit simultaneously when node B is transmitting, the transmitting frequency of node C must be different from its receiving frequency.

Hidden terminal problem with RTS-CTS-Data-ACK scheme



Exposed terminal problem.



7.2.5 Resource Constraints

Two essential and limited resources that form the major constraint for the nodes in an ad hoc wireless network are battery life and processing power. Devices used in ad hoc wireless networks in most cases require portability, and hence they also have size and weight constraints along with the restrictions on the power source. Increasing the battery power and processing ability makes the nodes bulky and less portable. Thus ad hoc wireless network routing protocols must optimally manage these resources.

6 Characteristics of an Ideal Routing Protocol for Ad Hoc Wireless Networks

Due to the issues in an ad hoc wireless network environment discussed so far, wired network routing protocols cannot be used in ad hoc wireless networks. Hence ad hoc wireless networks require specialized routing protocols that address the challenges described above. A routing protocol for ad hoc wireless networks should have the following characteristics:

1. It must be fully distributed, as centralized routing involves high control overhead and hence is not scalable. Distributed routing is more fault tolerant than centralized routing, which involves the risk of single point of failure.
2. It must be adaptive to frequent topology changes caused by the mobility of nodes.
3. Route computation and maintenance must involve a minimum number ofnodes. Each node in the network must have quick access to routes, that is, minimum connection setup time is desired.

4. It must be localized, as global state maintenance involves a huge state propagation control overhead.

5. It must be loop-free and free from stale routes.

6. The number of packet collisions must be kept to a minimum by limiting the number of broadcasts made by each node. The transmissions should be reliable to reduce message loss and to prevent the occurrence of stale routes.

7. It must converge to optimal routes once the network topology becomes stable. The convergence must be quick.

8. It must optimally use scarce resources such as bandwidth, computing power, memory, and battery power.

9. Every node in the network should try to store information regarding the stable local topology only. Frequent changes in local topology, and changes in the topology of parts of the network with which the node does not have any traffic correspondence, must not in any way affect the node, that is, changes in remote parts of the network must not cause updates in the topology information maintained by the node.

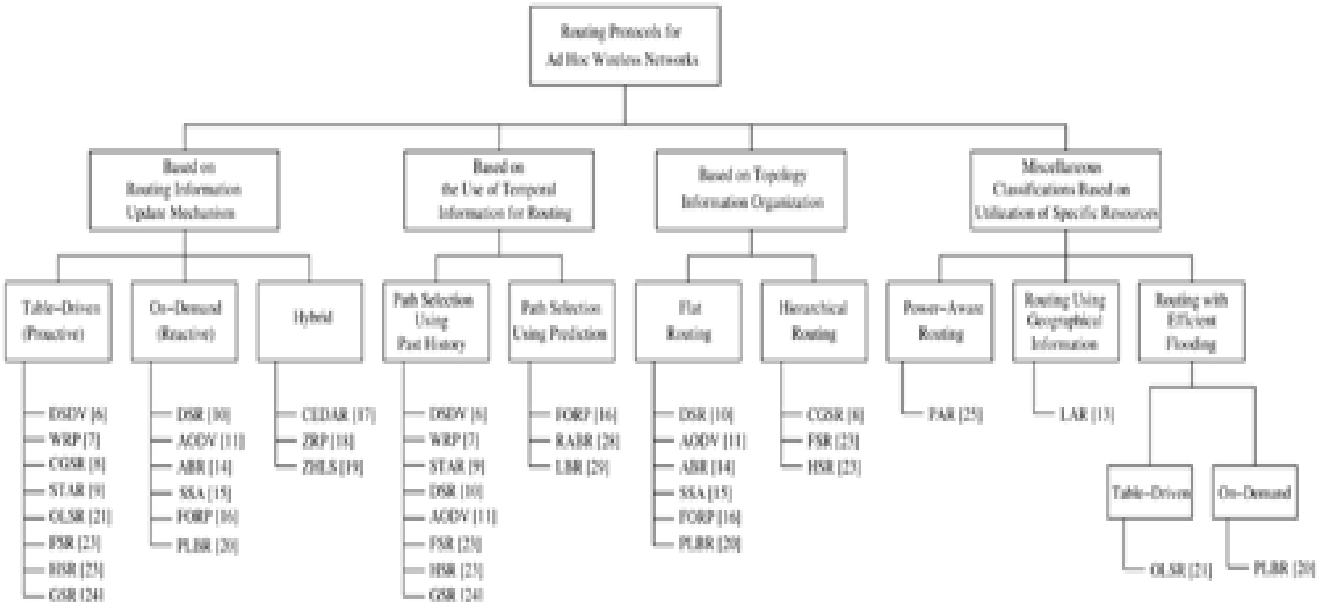
10. It should be able to provide a certain level of quality of service (QoS) as demanded by the applications, and should also offer support for time-sensitive traffic.

3 CLASSIFICATIONS OF ROUTING PROTOCOLS

Routing protocols for ad hoc wireless networks can be classified into several types based on different criteria. A classification tree is shown in Figure. Some of the classifications, their properties, and the basis of classifications are discussed below. The classification is not mutually exclusive and some protocols fall in more than one class. The deviation from the traditional routing metrics and path-finding processes that are employed in wired networks makes it worth further exploration in this direction. The routing protocols for ad hoc wireless networks can be broadly classified into four categories based on

- Routing information update mechanism
- Use of temporal information for routing
- Routing topology
- Utilization of specific resources

Figure. Classifications of routing protocols.



7.3.1 Based on the Routing Information Update Mechanism

Ad hoc wireless network routing protocols can be classified into three major categories based on the routing information update mechanism. They are:

1. **Proactive or table-driven routing protocols:** In table-driven routing protocols, every node maintains the network topology information in the form of routing tables by periodically exchanging routing information. Routing information is generally flooded in the whole network. Whenever a node requires a path to a destination, it runs an appropriate path-finding algorithm on the topology information it maintains. Table-driven routing protocols are further explored in Section 7.4.

2. **Reactive or on-demand routing protocols:** Protocols that fall under this category do not maintain the network topology information. They obtain the necessary path when it is required, by using a connection establishment process. Hence these protocols do not exchange routing information periodically. Some of the existing routing protocols that belong to this category are discussed in Section 7.5.

3. **Hybrid routing protocols:** Protocols belonging to this category combine the best features of the above two categories. Nodes within a certain distance from the node concerned, or with in a

particular geographical region, are said to be within the routing zone of the given node. For routing within this zone, a table-driven approach is used. For nodes that are located beyond this zone, an on-demand approach is used. Section 7.6 describes the protocols belonging to this category in detail.

7.3.2 Based on the Use of Temporal Information for Routing

This classification of routing protocols is based on the use of temporal information used for routing. Since ad hoc wireless networks are highly dynamic and path breaks are much more frequent than in wired networks, the use of temporal information regarding the lifetime of the wireless links and the lifetime of the paths selected assumes significance. The protocols that fall underthis category can be further classified into two types:

1. **Routing protocols using past temporal information:** These routing protocols use information about the past status of the links or the status of links at the time of routing to make routing decisions. For example, the routing metric based on the availability of wireless links (which is the current/present information here) along with a shortest path-finding algorithm, provides a path that may be efficient and stable at the time of path-finding. The topological changes may immediately break the path, making the path undergo a resource-wise expensive path reconfiguration process.

2. **Routing protocols that use future temporal information:** Protocols belonging to this category use information about the expected future status of the wireless links to make approximate routing decisions. Apart from the lifetime of wireless links, the future status information also includes information regarding the lifetime of the node (which is based on the remaining battery charge and discharge rate of the non-replenishable resources), prediction of location, and prediction of link availability.

7.3.3 Based on the Routing Topology

Routing topology being used in the Internet is hierarchical in order to reduce the state information maintained at the core routers. Ad hoc wireless networks, due to their relatively smaller number of nodes, can make use of either a flat topology or a hierarchical topology for routing.

1. **Flat topology routing protocols:** Protocols that fall under this category make use of a flat addressing scheme similar to the one used in IEEE 802.3 LANs. It assumes the presence of a globally unique (or at least unique to the connected part of the network) addressing mechanism

for nodes in an ad hoc wireless network.

2. **Hierarchical topology routing protocols:** Protocols belonging to this category make use of a logical hierarchy in the network and an associated addressing scheme. The hierarchy could be based on geographical information or it could be based on hop distance.

7.3.4 Based on the Utilization of Specific Resources

1. **Power-aware routing:** This category of routing protocols aims at minimizing the consumption of a very important resource in the ad hoc wireless networks: the battery power. The routing decisions are based on minimizing the power consumption either locally or globally in the network.

2. **Geographical information assisted routing:** Protocols belonging to this category improve the performance of routing and reduce the control overhead by effectively utilizing the geographical information available. The following section further explores the above classifications and discusses specific routing protocols belonging to each category in detail.

7.4 TABLE-DRIVEN ROUTING PROTOCOLS

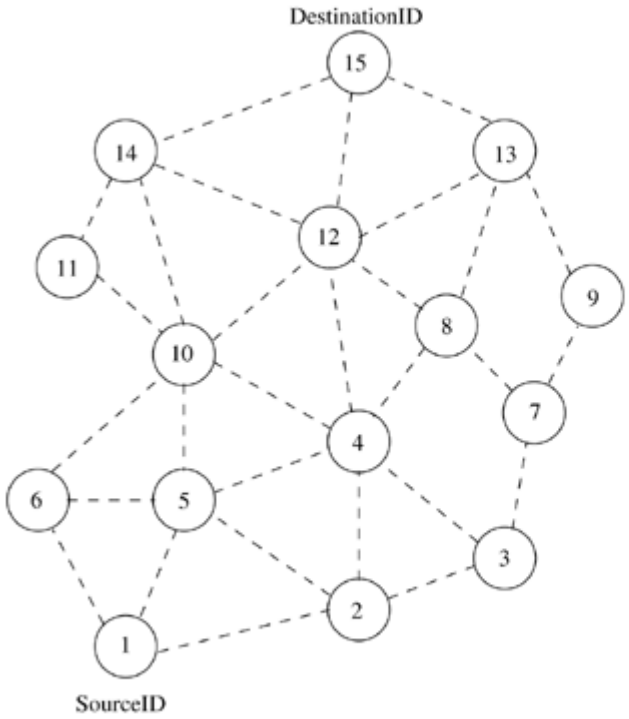
These protocols are extensions of the wired network routing protocols. They maintain the global topology information in the form of tables at every node. These tables are updated frequently in order to maintain consistent and accurate network state information. The destination sequenced distance-vector routing protocol (DSDV), wireless routing protocol (WRP), source-tree adaptive routing protocol (STAR), and cluster-head gateway switch routing protocol (CGSR) are some examples for the protocols that belong to this category.

7.4.1 Destination Sequenced Distance-Vector Routing Protocol

The destination sequenced distance-vector routing protocol (DSDV) [6] is one of the first protocols proposed for ad hoc wireless networks. It is an enhanced version of the distributed Bellman-Ford algorithm where each node maintains a table that contains the shortest distance and the first node on the shortest path to every other node in the network. It incorporates table updates with increasing sequence number tags to prevent loops, to counter the count-to-infinity problem, and for faster convergence. As it is a table-driven routing protocol, routes to all destinations are readily available at every node at all times. The tables are exchanged between neighbors at regular intervals to keep an up-to-date view of the network topology. The tables are also forwarded if a node observes a significant change in local topology. The table updates are of

two types: incremental updates and full dumps. An incremental update takes a single network data packet unit (NDPU), while a full dump may take multiple NDPUs. Incremental updates are used when a node does not observe significant changes in the local topology. A full dump is done either when the local topology changes significantly or when an incremental update requires more than a single NDPU. Table updates are initiated by a destination with a new sequence number which is always greater than the previous one. Upon receiving an updated table, a node either updates its tables based on the received information or holds it for some time to select the best metric (which may be the lowest number of hops) received from multiple versions of the same update table from different neighboring nodes. Based on the sequence number of the table update, it may forward or reject the table. Consider the example as shown in Figure 7.5 (a). Here node 1 is the source node and node 15 is the destination. As all the nodes maintain global topology information, the route is already available as shown in Figure 7.5 (b). Here the routing table of node 1 indicates that the shortest route to the destination node (node 15) is available through node 5 and the distance to it is 4 hops, as depicted in Figure 7.5 (b).

Route establishment in DSDV.



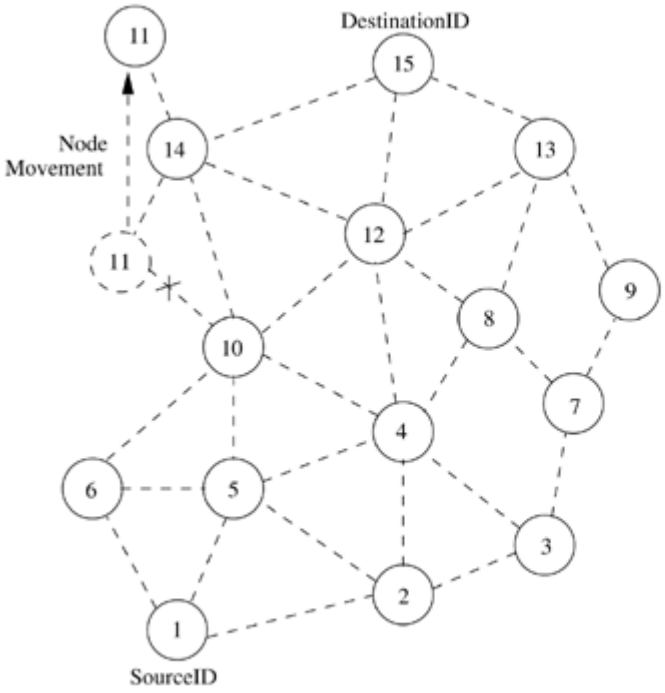
(a) Topology graph of the network

Dest	NextNode	Dist	SeqNo
2	2	1	22
3	2	2	26
4	5	2	32
5	5	1	134
6	6	1	144
7	2	3	162
8	5	3	170
9	2	4	186
10	6	2	142
11	6	3	176
12	5	3	190
13	5	4	198
14	6	3	214
15	5	4	256

(b) Routing table for Node 1

The reconfiguration of a path used by an on-going data transfer session is handled by the protocol in the following way. The end node of the broken link initiates a table update message with the broken link's weight assigned to infinity (∞) and with a sequence number greater than the stored sequence number for that destination. Each node, upon receiving an update with weight ∞, quickly disseminates it to its neighbors in order to propagate the broken-link information to the whole network. Thus a single link break leads to the propagation of table update information to the whole network. A node always assigns an odd sequence number to the link break update to differentiate it from the even sequence number generated by the destination. Consider the case when node 11 moves from its current position, as shown in Figure 7.6. When a neighbor node perceives the link break, it sets all the paths passing through the broken link with distance as ∞. For example, when node 10 knows about the mlink break, it sets the path to node 11 as ∞ and broadcasts its routing table to its neighbors. Those neighbors detecting significant changes in their routing tables rebroadcast it to their neighbors. In this way, the broken link information propagates throughout the network. Node 1 also sets the distance to node 11 as ∞. When node 14 receives a table update message from node 11, it informs the neighbors about the shortest distance to node 11. This information is also propagated throughout the network. All nodes receiving the new update message with the higher sequence number set the new distance to node 11 in their corresponding tables. The updated table at node 1 is shown in Figure 7.6, where the current distance from node 1 to node 11 has increased from three to four hops.

Route maintenance in DSDV.



Routing Table for Node 1

Dest	NextNode	Dist	SeqNo
2	2	1	22
3	2	2	26
4	5	2	32
5	5	1	134
6	6	1	144
7	2	3	162
8	5	3	170
9	2	4	186
10	6	2	142
11	5	4	180
12	5	3	190
13	5	4	198
14	6	3	214
15	5	4	256

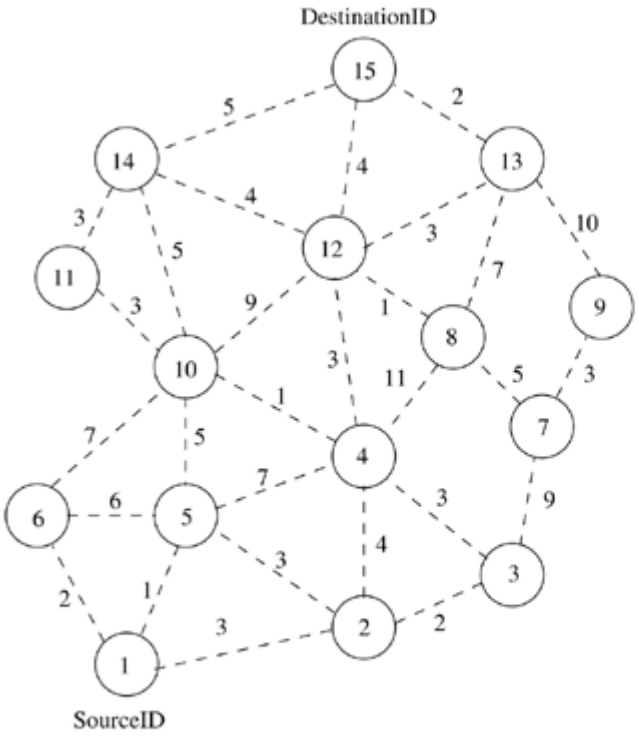
Advantages and Disadvantages

The availability of routes to all destinations at all times implies that much less delay is involved in the route setup process. The mechanism of incremental updates with sequence number tags makes the existing wired network protocols adaptable to ad hoc wireless networks. Hence, an existing wired network protocol can be applied to ad hoc wireless networks with many fewer modifications. The updates are propagated throughout the network in order to maintain an up-to-date view of the network topology at all the nodes. The updates due to broken links lead to a heavy control overhead during high mobility. Even a small network with high mobility or a large network with low mobility can completely choke the available bandwidth. Hence, this protocol suffers from excessive control overhead that is proportional to the number of nodes in the network and therefore is not scalable in ad hoc wireless networks, which have limited bandwidth and whose topologies are highly dynamic. Another disadvantage of DSDV is that in order to obtain information about a particular destination node, a node has to wait for a table update message initiated by the same destination node. This delay could result in stale routing information at nodes.

Wireless Routing Protocol

The wireless routing protocol (WRP) [7], similar to DSDV, inherits the properties of the distributed Bellman-Ford algorithm. To counter the count-toinfinity problem and to enable faster convergence, it employs a unique method of maintaining information regarding the shortest distance to every destination node in the network and the penultimate hop node on the path to every destination node. Since WRP, like DSDV, maintains an up-to-date view of the network, every node has a readily available route to every destination node in the network. It differs from DSDV in table maintenance and in the update procedures. While DSDV maintains only one topology table, WRP uses a set of tables to maintain more accurate information. The tables that are maintained by a node are the following: distance table (DT), routing table (RT), link cost table (LCT), and a message retransmission list (MRL). The DT contains the network view of the neighbors of a node. It contains a matrix where each element contains the distance and the penultimate node reported by a neighbor for a particular destination. TheRT contains the up-todate view of the network for all known destinations. It keeps the shortest distance, the *predecessor* node (penultimate node), the *successor* node (the next node to reach the destination), and a flag indicating the status of the path. The path status may be a simple path (correct), or a loop (error), or the destination node not marked (null). The LCT contains the cost (*e.g.*,the number of hops to reach the destination) of relaying messages through each link. The cost of a broken link is ∞. It also contains the number of update periods (intervals between two successive periodic updates) passed since the last successful update was received from that link. This is done to detect link breaks. The MRL contains an entry for every update message that is to be retransmitted and maintains a counter for each entry. This counter is decremented after every retransmission of an update message. Each update message contains a list of updates. A node also marks each node in theRT that has to acknowledge the update message it transmitted. Once the counter reaches zero, the entries in the update message for which no acknowledgments have been received are to be retransmitted and the update message is deleted. Thus, a node detects a link break by the number of update periods missed since the last successful transmission. After receiving an update message, a node not only updates the distance for transmitted neighbors but also checks the other neighbors' distance, hence convergence is much faster than DSDV.

Consider the example shown in Figure 7.7, where the source of the route is node 1 and the destination is node 15. As WRP proactively maintains the route to all the destinations, the route to any destination node is readily available at the source node. From the routing table shown in Figure 7.7, the route from node 1 to node 15 has the next node as node 2. The predecessor node of 15 corresponding to this route is node 12. The predecessor information helps WRP to converge quickly during link breaks.



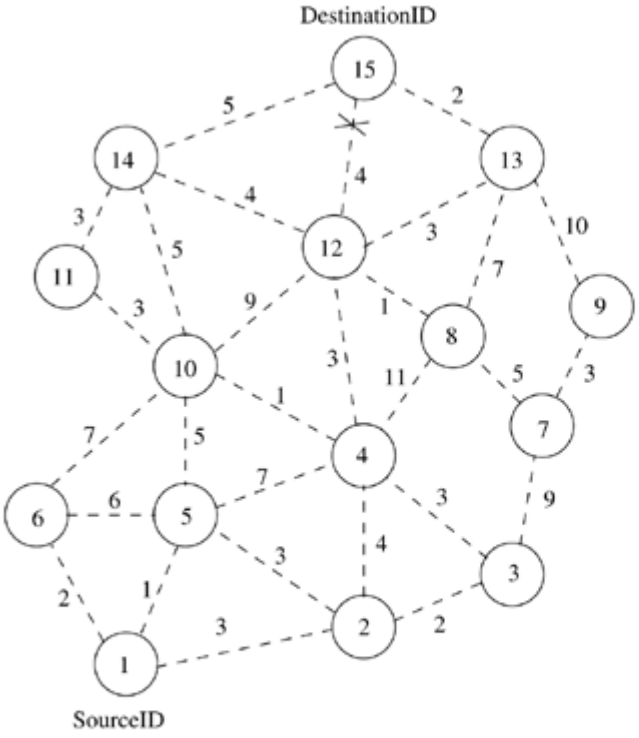
Routing Entry at Each Node for DestinationID 15

Node	NextNode	Pred	Cost
15	15	15	0
14	15	14	5
13	15	13	2
12	15	12	4
11	14	14	8
10	4	12	8
9	13	13	12
8	12	12	5
7	8	12	10
6	10	12	15
5	10	12	13
4	12	12	10
3	4	12	7
2	4	12	11
1	2	12	14

Fig:Route establishment in WRP.

When a node detects a link break, it sends an update message to its neighbors ith the link cost of the broken link set to ∞. After receiving the update message, all affected nodes update their minimum distances to the corresponding nodes (including the distance to the destination). The node that initiated the update message then finds an alternative route, if available from its DT. Note that this new route computed will not contain the broken link. Consider the scenario shown in Figure 7.8. When the link between nodes 12 and 15 breaks, all nodes having a route to the

destination with predecessor as node 12 delete their corresponding routing entries. Both node 12 and node 15 send update messages to their neighbors indicating that the cost of the link between nodes 12 and 15 is ∞. If the nodes have any other alternative route to the destination node 15, they update their routing tables and indicate the changed route to their neighbors by sending an update message. A neighbor node, after receiving an update message, updates its routing table only if the new path is better than the previously existing paths. For example, when node 12 finds an alternative route to the destination through node 13, it broadcasts an update message indicating the changed path. After receiving the update message from node 12, neighboring nodes 8, 14, 15, and 13 do not change their routing entry corresponding to destination 15 while node 4 and node 10 modify their entries to reflect the new updated path. Nodes 4 and 10 again send an update message to indicate the correct path to the destination for their respective neighbors. When node 10 receives node 4's update message, it again modifies its routing entry to optimize the path to the destination node (15) while node 4 discards the update entry it received from node 10.



Routing Entry at Each Node for DestinationID 15

Node	NextNode	Pred	Cost
15	15	15	0
14	15	14	5
13	15	13	2
12	15	13	5
11	14	14	8
10	4	13	9
9	13	13	12
8	12	13	6
7	8	13	11
6	10	13	16
5	10	13	14
4	12	13	8
3	4	13	11
2	4	13	12
1	2	13	15

Route maintenance in WRP.

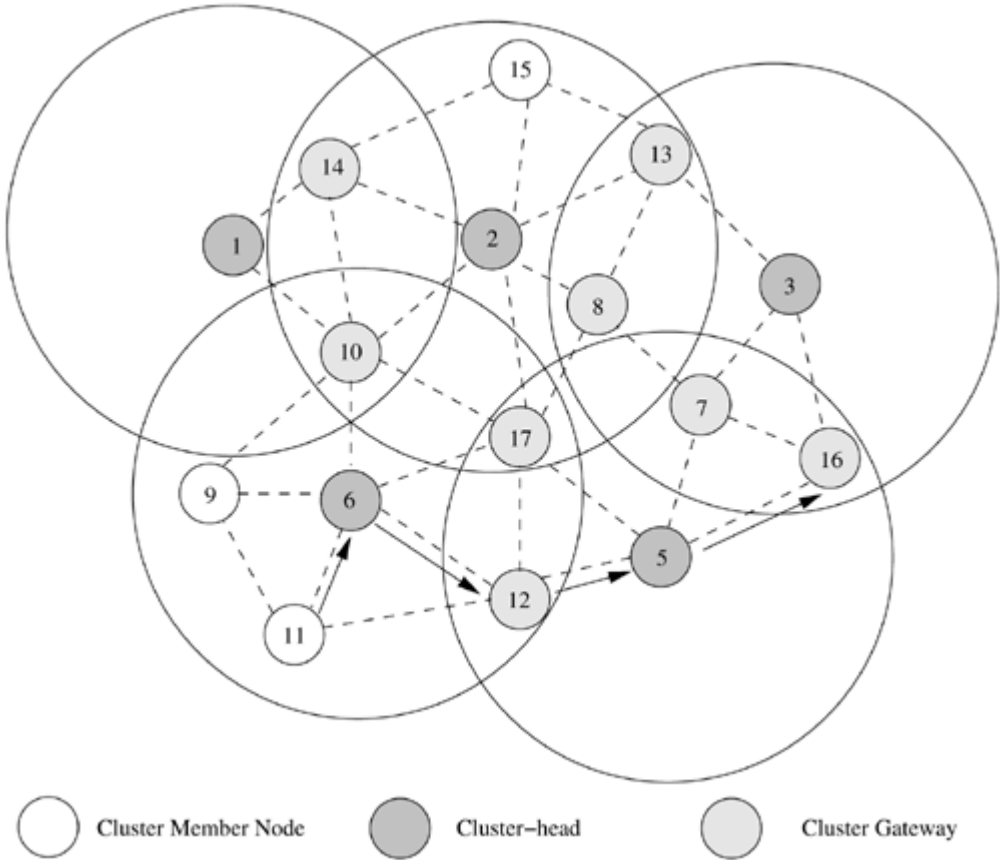
Advantages and Disadvantages

WRP has the same advantages as that of DSDV. In addition, it has faster convergence and involves fewer table updates. But the complexity of maintenance of multiple tables demands a larger memory and greater processing power from nodes in the ad hoc wireless network. At high mobility, the control overhead involved in updating table entries is almost the same as that of DSDV and hence is not suitable for highly dynamic and also for very large ad hoc wireless networks.

Cluster-Head Gateway Switch Routing Protocol

The cluster-head gateway switch routing protocol (CGSR) [8] uses a hierarchical network topology, unlike other table-driven routing approaches that employ flat topologies. CGSR organizes nodes into clusters, with coordination among the members of each cluster entrusted to a special node named*clusterhead.* This cluster-head is elected dynamically by employing a *least cluster change (LCC)*algorithm [8]. According to this algorithm, a node ceases to be a cluster-head only if it comes under the range of another cluster-head, where the tie is broken either using the lowest ID or highest connectivity algorithm. Clustering provides a mechanism to allocate bandwidth, which is a limited resource, among different clusters, thereby improving reuse. For example, different cluster-heads could operate on different spreading codes on a CDMA system. Inside a cluster, the cluster-head can coordinate the channel access based on a token-based polling protocol. All member nodes of a cluster can be reached by the cluster-head within a single hop, thereby enabling the cluster-head to provide improved coordination among nodes that fall under its cluster. A token-based scheduling (assigning access token to the nodes in a cluster) is used within a cluster for sharing the bandwidth among the members of the cluster. CGSR assumes that all communication passes through the clusterhead. Communication between two clusters takes place through the common member nodes that are members of both the clusters. These nodes which are members of more than one cluster are called*gateways.* A gateway is expected to be able to listen to multiple spreading codes that are currently in operation in the clusters in which the node exists as a member. A gateway conflict is said to occur when a cluster-head issues a token to a gateway over a spreading code while the gateway is tuned to another code. Gateways that are capable of simultaneously communicating over two interfaces

can avoid gateway conflicts. The performance of routing is influenced by token scheduling and code scheduling (assigning appropriate spreading codes to two different clusters) that are handled at cluster-heads and gateways, respectively. The routing protocol used in CGSR is an extension of DSDV. Every member node maintains a routing table containing the destination cluster-head for every node in the network. In addition to the cluster member table, each node maintains a routing table which keeps the list of next-hop nodes for reaching every destination cluster. The *cluster (hierarchical) routing protocol* is used here. As per this protocol, when a node with packets to be transmitted to a destination gets the token from its cluster-head, it obtains the destination cluster-head and the nexthop node from the cluster member table and the routing table, respectively. CGSR improves the routing performance by routing packets through the cluster-heads and gateways. A path from any node *a* to any node *b* will be similar to $a - C1 - G1 - C2 - G2 - ...Ci - Gj...Gn - b$, where *Gi* and *Cj* are the *ith* gateway and the *jth* cluster-head, respectively, in the path. Figure 7.9 shows the clusterheads, *cluster gateways*, and normal cluster member nodes in an ad hoc wireless network. A path between node 11 and node 16 would follow 11 - 6 - 12 - 5 - 16. Since the cluster-heads gain more opportunities for transmission, the clusterheads, by means of a dynamic scheduling mechanism, can make CGSR obtain better delay performance for real-time flows. Route reconfiguration is necessitated by mainly two factors: firstly, the change in cluster-head and secondly, the stale entries in the cluster member table and routing table. CGSR depends on the table update mechanism to handle the latter problem, while the least cluster change algorithm [8] handles the former.



Advantages and Disadvantages

CGSR is a hierarchical routing scheme which enables partial coordination between nodes by electing cluster-heads. Hence, better bandwidth utilization is possible. It is easy to implement priority scheduling schemes with token scheduling and gateway code scheduling. The main disadvantages ofCGSR are increase in path length and instability in the system at high mobility when the rate of change of cluster-heads is high. In order to avoid gateway conflicts, more resources (such as additional interfaces) are required. The power consumption at the cluster-head node is also a matter of concern because the battery-draining rate at the cluster-head is higher than that at a normal node. This could lead to frequent changes in the cluster-head, which may result in multiple path breaks.

Source-Tree Adaptive Routing Protocol

Source-tree adaptive routing protocol (STAR) [9] proposed by Garcia-Luna- Aceves and Spohn is a variation of table-driven routing protocols, with the least overhead routing approach (LORA) as the key concept rather than the optimum routing approach (ORA) that was employed by

earlier table-driven routing protocols. The ORA protocols attempt to update routing information quickly enough to provide optimum paths with respect to the defined metric (which may be the lowest number of hops), but withLORA, the routing protocol attempts to provide feasible paths that are not guaranteed to be optimal, but involve much less control overhead. In STAR protocol, every node broadcasts its *sourcetree*information. The source-tree of a node consists of the wireless links used by the node in its preferred path to destinations. Every node, using its adjacent links and the source-tree broadcast by its neighbors, builds a partial graph of the topology. During initialization, a node sends an update message to its neighbors. Also, every node is required to originate update messages about new destinations, the chances of routing loops, and the cost of paths exceeding a given threshold. Hence, each node will have a path to every destination node.The path, in most cases, would be sub-optimal. In the absence of a reliable link layer broadcast mechanism, STAR uses the following path-finding approach. When a node *s* has data packets to send to a particular destination *d,* for which no path exists in its source-tree, it originates an update message to all its neighbors indicating the absence of a path to *d*. This update message triggers another update message from a neighbor which has a path to *d.*Node *s* retransmits the update message as long as it does not have a path to *d* with increasing intervals between successive retransmissions. After getting the source-tree update from a neighbor, the node *s*updates its source-tree and, using this, it finds a path to all nodes in the network. The data packet contains information about the path to be traversed in order to prevent the mpossibility of routing loop formation. In the presence of a reliable broadcast mechanism, STAR assumes implicit route maintenance. The link update message about the unavailability of a next-hop node triggers an update message from a neighbor which has an alternate source tree indicating an alternate next-hop node to the destination. In addition to path breaks, the intermediate nodes are responsible for handling the routing loops. When an intermediate node *k* receives a data packet to destination *d,* and one of the nodes in the packet's traversed path is present in node *k*'s path to the destination *d,* then it discards the packet and a*RouteRepair* update message is reliably sent to the node in the head of the route repair path. The route repair path corresponds to the path *k* to *x,* where *x* is the last router in the data packet's traversed path that is first found in the path *k* to *d,* that belongs to the source tree of *k.* The *RouteRepair* packet contains the complete source tree of node *k* and the traversed path of the packet. When an

intermediate node receives a *RouteRepair* update message, it removes itself from the top of the route repair path and reliably sends it to the head of the route repair path.

Advantages and Disadvantages

STAR has very low communication overhead among all the table-driven routing protocols. The use of the LORA approach in this table-driven routing protocol reduces the average control overhead compared to several other on-demand routing protocols.

7.5 ON-DEMAND ROUTING PROTOCOLS

Unlike the table-driven routing protocols, on-demand routing protocols execute the path-finding process and exchange routing information only when a path is required by a node to communicate with a destination. This section explores some of the existing on-demand routing protocols in detail.

7.5.1 Dynamic Source Routing Protocol

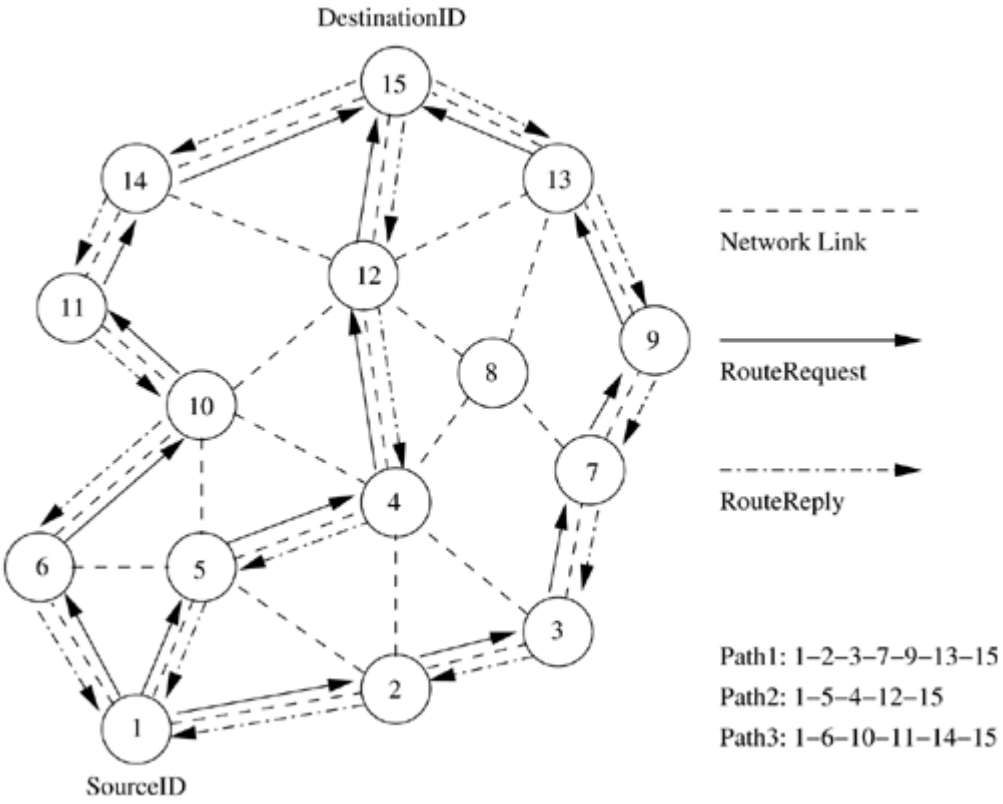
Dynamic source routing protocol (DSR) [10] is an on-demand protocol designed to restrict the bandwidth consumed by control packets in ad hoc wireless networks by eliminating the periodic table-update messages required in the table-driven approach. The major difference between this and the other ondemand routing protocols is that it is *beacon-less* and hence does not require periodic *hello*packet (*beacon*) transmissions, which are used by a node to inform its neighbors of its presence. The basic approach of this protocol (and all other on-demand routing protocols) during the route construction phase is to establish a route by flooding *RouteRequest* packets in the network. The destination node, on receiving a *RouteRequest* packet, responds by sending a *RouteReply* packet back to the source, which carries the route traversed by the *RouteRequest* packet received.

Consider a source node that does not have a route to the destination. When it has data packets to be sent to that destination, it initiates a *RouteRequest* packet. This *RouteRequest* is flooded throughout the network. Each node, upon receiving a *RouteRequest* packet, rebroadcasts the packet to its neighbors if it has not forwarded already or if the node is not the destination node, provided the packet's time to live (TTL) counter has not exceeded.

Each *RouteRequest* carries a sequence number generated by the source node and

the path it has traversed. A node, upon receiving a *RouteRequest*packet, checks the sequence number on the packet before forwarding it. The packet is forwarded only if it is not a duplicate *RouteRequest.* The sequence number on the packet is used to prevent loop formations and to avoid multiple transmissions of the same *RouteRequest* by an intermediate node that receives it through multiple paths. Thus, all nodes except the destination forward a *RouteRequest*packet during the route construction phase. A destination node, after receiving the first *RouteRequest*packet, replies to the source node through the reverse path the *RouteRequest* packet had traversed. InFigure 7.10, source node 1 initiates a *RouteRequest* packet to obtain a path for destination node 15. This protocol uses a route cache that stores all possible information extracted from the source route contained in a data packet. Nodes can also learn about the neighboring routes traversed by data packets if operated in the promiscuous mode (the mode of operation in which a node can receive the packets that are neither broadcast nor addressed to itself). This route cache is also used during the route construction phase. If an intermediate node receiving a *RouteRequest* has a route to the destination node in its route cache, then it replies to the source node by sending a *RouteReply* with the entire route information from the source node to the destination node.

Route establishment in DSR.

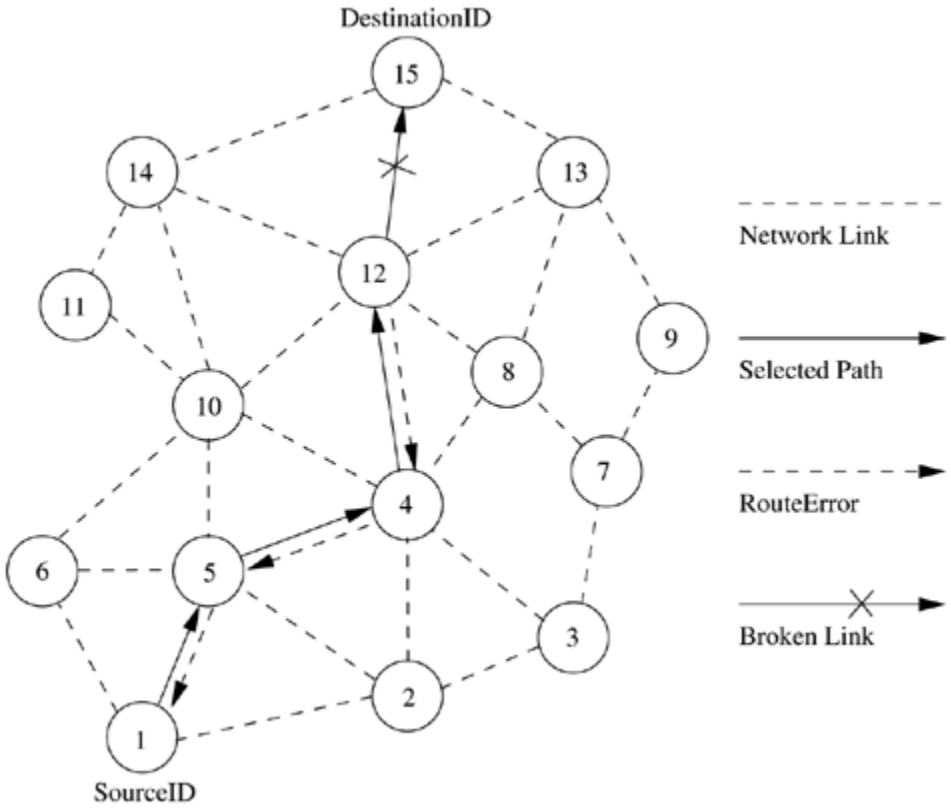


Optimizations

Several optimization techniques have been incorporated into the basic DSR protocol to improve the performance of the protocol. DSR uses the route cache at intermediate nodes. The route cache is populated with routes that can be extracted from the information contained in data packets that get forwarded. This cache information is used by the intermediate nodes to reply to the source when they receive a *RouteRequest* packet and if they have a route to the corresponding destination. By operating in the promiscuous mode, an intermediate node learns about route breaks. Information thus gained is used to update the route cache so that the active routes maintained in the route cache do not use such broken links. During network partitions, the affected nodes initiate *RouteRequest* packets. An exponential backoff algorithm is used to avoid frequent *RouteRequest* flooding in the network when the destination is in another disjoint set. DSR also allows piggy-backing of a data packet on the*RouteRequest* so that a data packet can be sent along with the *RouteRequest.* If optimization is not allowed in the DSR protocol, the route construction phase

is very simple. All the intermediate nodes flood the *RouteRequest* packet if it is not redundant. For example, after receiving the *RouteRequest* packet from node 1 (refer to Figure 7.10), all its neighboring nodes, that is, nodes 2, 5, and 6, forward it. Node 4 receives the *RouteRequest* from both nodes 2 and 5. Node 4 forwards the first *RouteRequest* it receives from any one of the nodes 2 and 5 and discards the other redundant/duplicate *RouteRequest* packets.

The *RouteRequest* is propagated till it reaches the destination which initiates the *RouteReply.* As part of optimizations, if the intermediate nodes are also allowed to originate *RouteReply* packets, then a source node may receive multiple replies from intermediate nodes. For example, in Figure 7.11, if the intermediate node 10 has a route to the destination via node 14, it also sends the *RouteReply* to the source node. The source node selects the latest and best route, and uses that for sending data packets. Each data packet carries the complete path to its destination.



Route maintenance in DSR.

When an intermediate node in the path moves away, causing a wireless link to break, for example, the link between nodes 12 and 15 in Figure 7.11, a *RouteError* message is generated from the node adjacent to the broken link to inform the source node. The source node reinitiates the route establishment procedure. The cached entries at the intermediate nodes and the source node are removed when a*RouteError* packet is received. If a link breaks due to the movement of edge nodes (nodes 1 and 15), the source node again initiates the route discovery process.

Advantages and Disadvantages

This protocol uses a reactive approach which eliminates the need to periodically flood the network with table update messages which are required in a tabledriven approach. In a reactive (on-demand) approach such as this, a route is established only when it is required and hence the need to find routes to all other nodes in the network as required by the table-driven approach is eliminated. The intermediate nodes also utilize the route cache information efficiently to reduce the control overhead. The disadvantage of this protocol is that the route maintenance mechanism does not locally repair a broken link. Stale route cache information could also result in inconsistencies during the route reconstruction phase. The connection setup delay is higher than in table-driven protocols. Even though the protocol performs well in static and low-mobility environments, the performance degrades rapidly with increasing mobility. Also, considerable routing overhead is involved due to the source-routing mechanism employed in DSR. This routing overhead is directly proportional to the path length.

2 Ad Hoc On-Demand Distance-Vector Routing Protocol

Ad hoc on-demand distance vector (AODV) [11] routing protocol uses an ondemand approach for finding routes, that is, a route is established only when it is required by a source node for transmitting data packets. It employs destination sequence numbers to identify the most recent path. The major difference between AODV and DSR stems out from the fact that DSR uses source routing

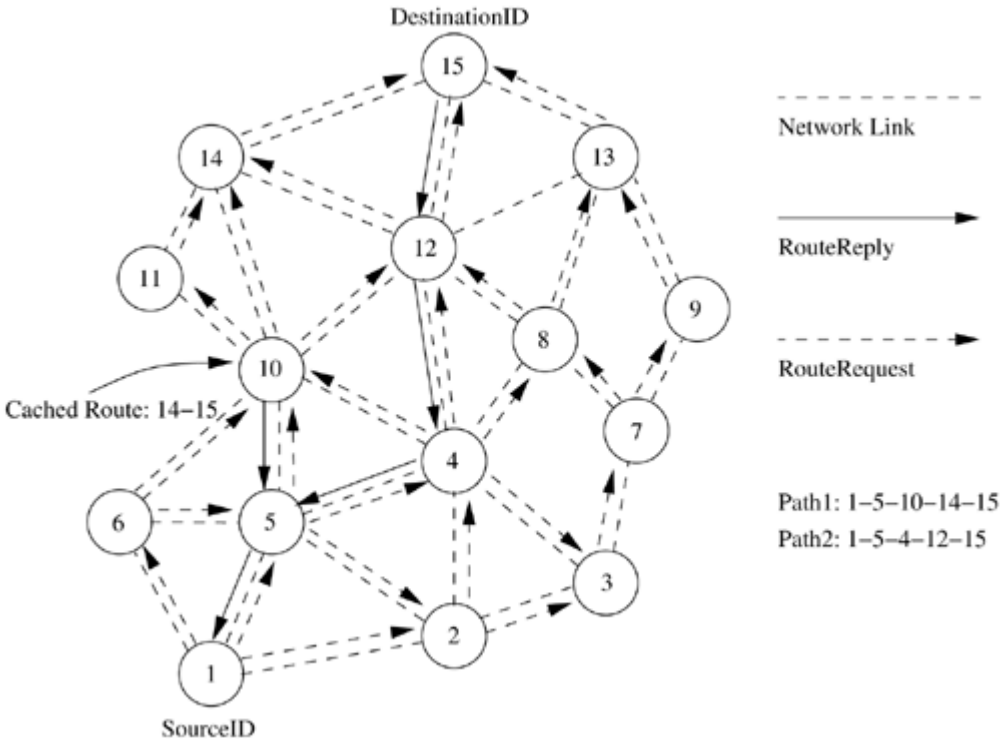
in which a data packet carries the complete path to be traversed. However, in AODV, the source node and the intermediate nodes store the next-hop information corresponding to each flow for data packet transmission. In an ondemand routing protocol, the source node floods the *RouteRequest* packet in the network when a route is not available for the desired destination. It may obtain multiple routes to different destinations from a single *RouteRequest.* The major difference between AODV and other on-demand routing protocols is that it uses a destination sequence number (DestSeqNum) to determine an up-to-date path to the destination. A node updates its path information only if the DestSeqNum of the current packet received is greater than the last DestSeqNum stored at the node.

A *RouteRequest* carries the source identifier (SrcID), the destination identifier (DestID), the source sequence number (SrcSeqNum), the destination sequence number (DestSeqNum), the broadcast identifier (BcastID), and the time to live (TTL) field. DestSeqNum indicates the freshness of the route that is accepted by the source. When an intermediate node receives a *RouteRequest,* it either forwards it or prepares a *RouteReply* if it has a valid route to the destination. The validity of a route at the intermediate node is determined by comparing the sequence number at the intermediate node with the destination sequence number in the *RouteRequest* packet. If a *RouteRequest* is received multiple times, which is indicated by the BcastID-SrcID pair, the duplicate copies are discarded. All intermediate nodes having valid routes to the destination, or the destination node itself, are allowed to send*RouteReply* packets to the source. Every intermediate node, while forwarding a *RouteRequest,* enters the previous node address and its BcastID. A timer is used to delete this entry in case a *RouteReply* is not received before the timer expires. This helps in storing an active path at the intermediate node asAODV does not employ source routing of data packets. When a node receives a *RouteReply* packet, information about the previous node from which the packet was received is also stored in order to forward the data packet to this next node as the next hop toward the destination.

Consider the example depicted in Figure 7.12. In this figure, source node 1

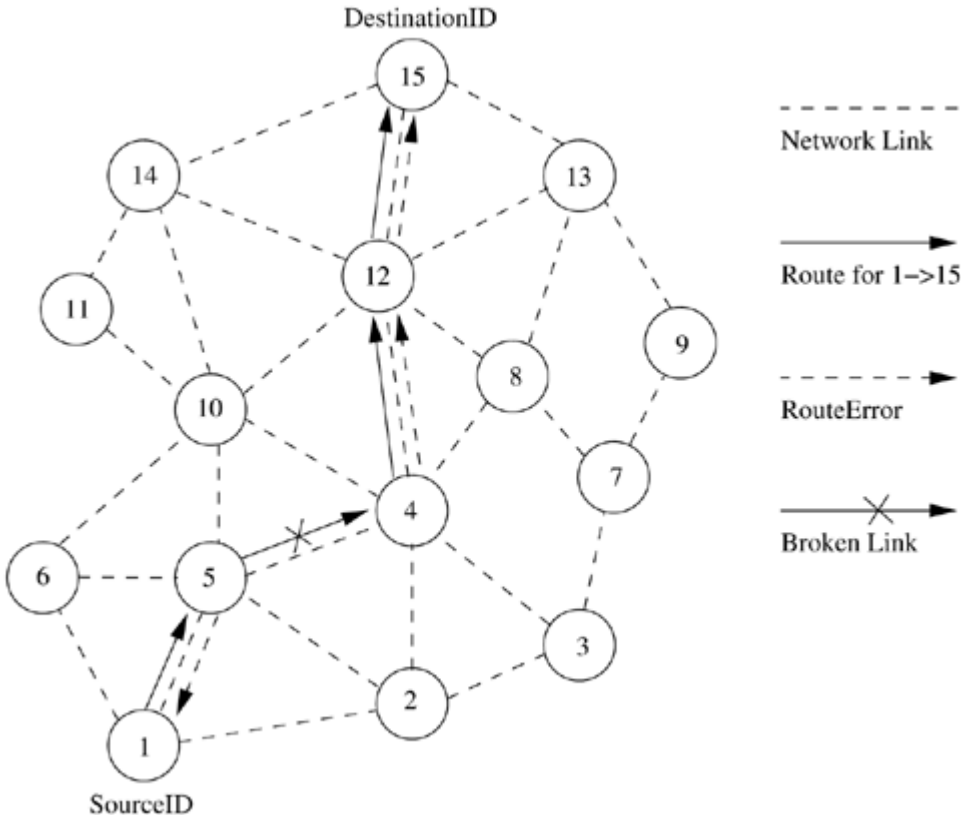
initiates a path-finding process by originating a *RouteRequest* to be flooded in the network for destination node 15, assuming that the *RouteRequest* contains the destination sequence number as 3 and the source sequence number as 1. When nodes 2, 5, and 6 receive the *RouteRequest* packet, they check their routes to the destination. In case a route to the destination is not available, they further forward it to their neighbors. Here nodes 3, 4, and 10 are the neighbors of nodes 2, 5, and 6. This is with the assumption that intermediate nodes 3 and 10 already have routes to the destination node, that is, node 15 through paths 10-14-15 and 3-7-9-13-15, respectively. If the destination sequence number at intermediate node 10 is 4 and is 1 at intermediate node 3, then only node 10 is allowed to reply along the cached route to the source. This is because node 3 has an older route to node 15 compared to the route available at the source node (the destination sequence number at node 3 is 1, but the destination sequence number is 3 at the source node), while node 10 has a more recent route (the destination sequence number is 4) to the destination. If the *RouteRequest* reaches the destination (node 15) through path 4-12-15 or any other alternative route, the destination also sends a *RouteReply* to the source. In this case, multiple*RouteReply* packets reach the source. All the intermediate nodes receiving a *RouteReply* update their route tables with the latest destination sequence number. They also update the routing information if it leads to a shorter path between source and destination.

Route establishment in AODV.



AODV does not repair a broken path locally. When a link breaks, which is determined by observing the periodical *beacon*s or through link-level acknowledgments, the end nodes (*i.e.*, source and destination nodes) are notified. When a source node learns about the path break, it reestablishes the route to the destination if required by the higher layers. If a path break is detected at an intermediate node, the node informs the end nodes by sending an unsolicited *RouteReply* with the hop count set as ∞.

Consider the example illustrated in Figure 7.13. When a path breaks, for example, between nodes 4 and 5, both the nodes initiate *RouteError* messages to inform their end nodes about the link break. The end nodes delete the corresponding entries from their tables. The source node reinitiates the pathfinding process with the new BcastID and the previous destination sequence number.



Advantages and Disadvantages

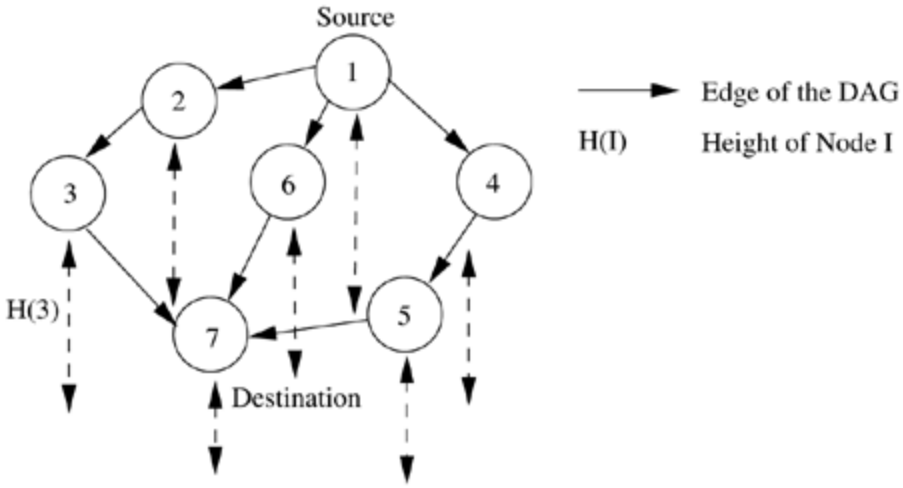
The main advantage of this protocol is that routes are established on demand and destination sequence numbers are used to find the latest route to the destination. The connection setup delay is less. One of the disadvantages of this protocol is that intermediate nodes can lead to inconsistent routes if the source sequence number is very old and the intermediate nodes have a higher but not the latest destination sequence number, thereby having stale entries. Also multiple *RouteReply* packets in response to a single *RouteRequest* packet can lead to heavy control overhead. Another disadvantage ofAODV is that the periodic *beacon*ing leads to unnecessary bandwidth consumption.

Temporally Ordered Routing Algorithm

Temporally ordered routing algorithm (TORA) [12] is a source-initiated ondemand routing protocol which uses a *link reversal algorithm* and provides loop-free multipath routes to a destination node. InTORA, each node maintains its one-hop local topology information and also has the capability to detect

partitions. TORA has the unique property of limiting the control packets to a small region during the reconfiguration process initiated by a path break. Figure 7.14 shows the distance metric used in TORAwhich is nothing but the length of the path, or the height from the destination. H(N) denotes the height of node N from the destination. TORA has three main functions: establishing, maintaining, and erasing routes.

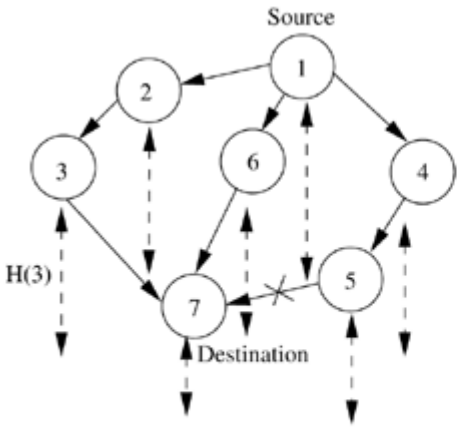
Illustration of temporal ordering in TORA.



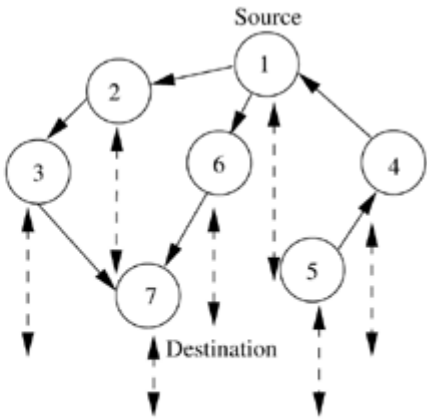
The route establishment function is performed only when a node requires a path to a destination but does not have any directed link. This process establishes a destination-oriented directed acyclic graph (DAG) using a *Query/Update* mechanism. Consider the network topology shown in Figure 7.14. When node 1 has data packets to be sent to the destination node 7, a *Query* packet is originated by node 1 with the destination address included in it. This *Query* packet is forwarded by intermediate nodes 2, 3, 4, 5, and 6, and reaches the destination node 7, or any other node which has a route to the destination. The node that terminates (in this case, node 7) the *Query* packet replies with an *Update* packet containing its distance from the destination (it is zero at the destination node). In the example, the destination node 7 originates an *Update* packet. Each node that receives the *Update* packet sets its distance to a value higher than the distance of the sender of the *Update* packet. By doing this, a set of directed links from the node which originated the *Query* to the

destination node 7 is created. This forms the DAGdepicted in Figure 7.14. Once a path to the destination is obtained, it is considered to exist as long as the path is available, irrespective of the path length changes due to the reconfigurations that may take place during the course of the data transfer session.

When an intermediate node (say, node 5) discovers that the route to the destination node is invalid, as illustrated in Figure 7.15, it changes its distance value to a higher value than its neighbors and originates an *Update* packet. The neighboring node 4 that receives the *Update* packet reverses the link between 1 and 4 and forwards the *Update* packet. This is done to update the DAG corresponding to destination node 7. This results in a change in the DAG. If the source node has no other neighbor that has a path to the destination, it initiates a fresh *Query/Update* procedure. Assume that the link between nodes 1 and 4 breaks. Node 4 reverses the path between itself and node 5, and sends an update message to node 5. Since this conflicts with the earlier reversal, a partition in the network can be inferred. If the node detects a partition, it originates a *Clear* message, which erases the existing path information in that partition related to the destination.



Link break beween Nodes 5 and 7



Nodes 4 and 5 reverse their links in order to update the path

Advantages and Disadvantages

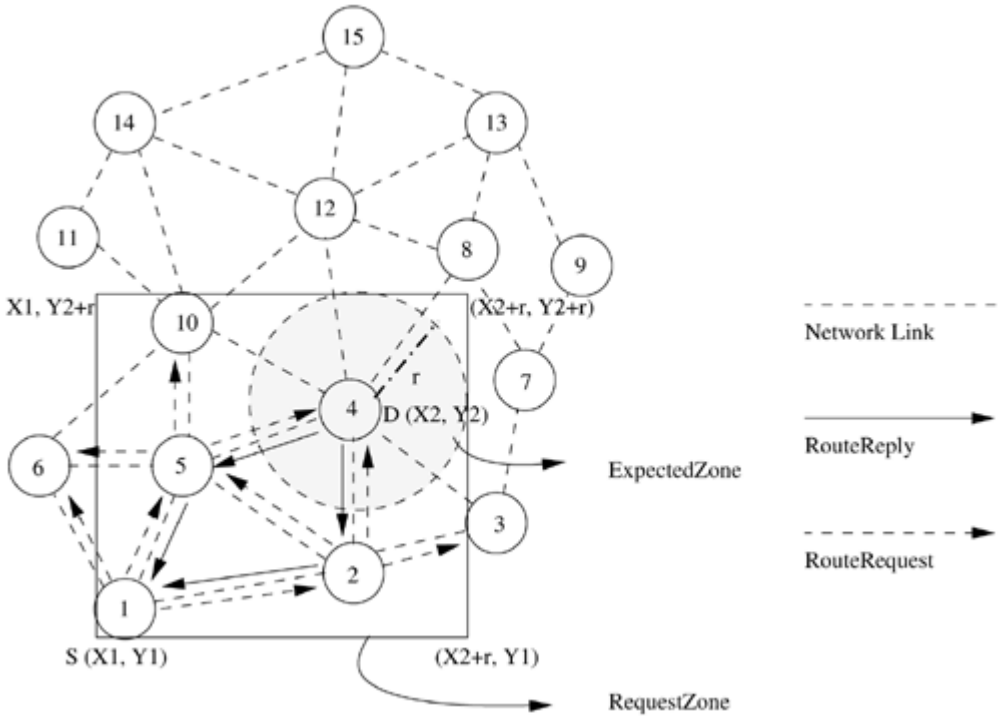
By limiting the control packets for route reconfigurations to a small region, TORA incurs less control overhead. Concurrent detection of partitions

and subsequent deletion of routes could result in temporary oscillations and transient loops. The local reconfiguration of paths results in non-optimal routes

7.5.4 Location-Aided Routing

Location-aided routing protocol (LAR) [13] utilizes the location information for improving the efficiency of routing by reducing the control overhead. LAR assumes the availability of the global positioning system (GPS) for obtaining the geographical position information necessary for routing.LAR designates two geographical regions for selective forwarding of control packets, namely,*ExpectedZone* and *RequestZone.* The *ExpectedZone* is the region in which the destination node is expected to be present, given information regarding its location in the past and its mobility information (refer to Figure 7.16). In the event of non-availability of past information about the destination, the entire network area is considered to be the *ExpectedZone* of the destination. Similarly, with the availability of more information about its mobility, the *ExpectedZone* of the destination can be determined with more accuracy and improved efficiency. The *RequestZone* is a geographical region within which the path-finding control packets are permitted to be propagated. This area is determined by the sender of a data transfer session. The control packets used for path-finding are forwarded by nodes which are present in the *RequestZone* and are discarded by nodes outside the zone. In situations where the sender or the intermediate relay nodes are not present in the *RequestZone*, additional area is included for forwarding the packets. This is done when the first attempt for obtaining a path to a destination using the initial *RequestZone* fails to yield a path within a sufficiently long waiting time. In this case, the second attempt repeats the process with increased *RequestZone* size to account for mobility and error in location estimation. LAR uses flooding, but here flooding is restricted to a small geographical region. The nodes decide to forward or discard the control packets based on two algorithms, namely, LAR1 and LAR2.

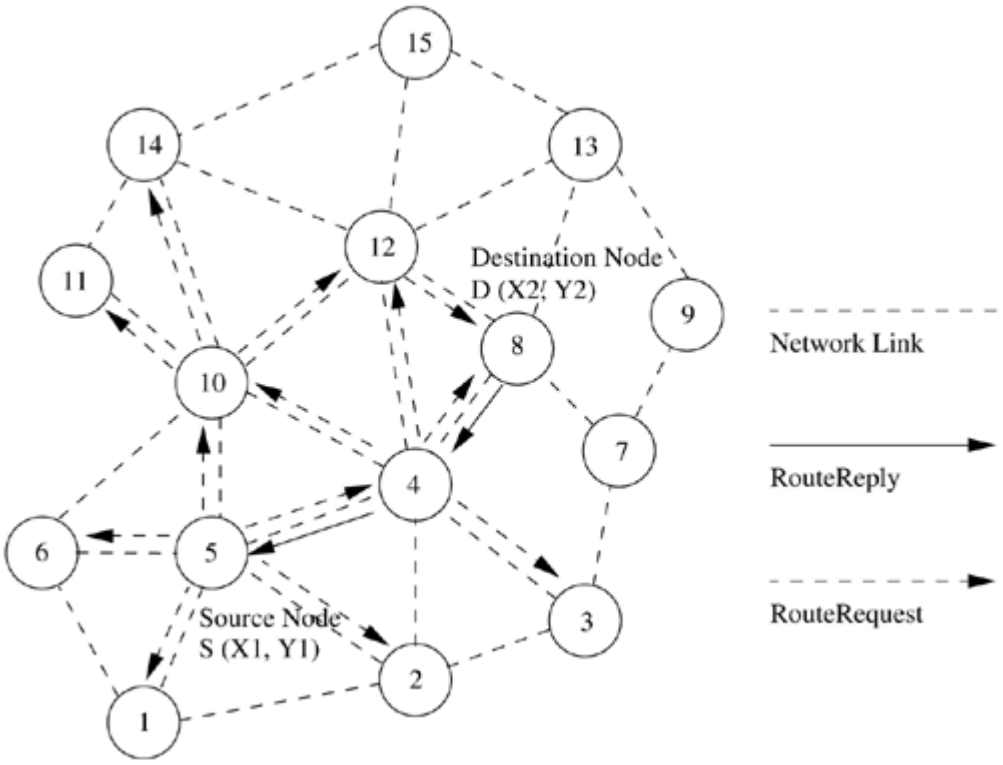
***RequestZone* and *ExpectedZone* in LAR1.**



In the LAR1 algorithm, the source node (say, *S*) explicitly specifies the *Request-Zone* in the*RouteRequest* packet. As per LAR1, as illustrated in Figure 7.16, the *RequestZone* is the smallest rectangle that includes the source node (*S*) and the *ExpectedZone,* the sides of which are parallel to the X and Y axes, when the node *S* is outside the *ExpectedZone.* When node *S* is within the *ExpectedZone,*then the *RequestZone* is reduced to the *ExpectedZone* itself. Every intermediate node that receives the*RouteRequest* packet verifies the *RequestZone* information contained in the packet and forwards it further if the node is within the *RequestZone;* otherwise, the packet is discarded. In Figure 7.16, the source node (node 1) originates a *RouteRequest,* which is broadcast to its neighbors (2, 5, and 6). These nodes verify their own geographical locations to check whether they belong to the *ExpectedZone.*Nodes 2 and 5 find that they are inside the *ExpectedZone* and hence they forward the *RouteRequest.*But node 6 discards the packet. Finally, when the *RouteRequest* reaches the destination node (node 4), it originates a *RouteReply* that contains the current location and current time of the node. Also, as an option, the current speed of movement can

be included in the *RouteReply* if that information is available with the node. Such information included in the *RouteReply* packet is used by the source node for future route establishment procedures.

In LAR2 algorithm (Figure 7.17), the source node *S* (node 5) includes the distance between itself and the destination node *D* (node 8) along with the (*X*, *Y*) coordinates of the destination node *D* in the*RouteRequest* packet instead of the explicit information about the *Expected Region.* When an intermediate node receives this *RouteRequest* packet, it computes the distance to the node *D.* If this distance is less than the distance from *S* to node $D + \delta$, where δ is a parameter of the algorithm decided based on the error in location estimation and mobility, then the *RouteRequest* packet is forwarded. Otherwise, the *RouteRequest* is discarded. Consider the example illustrated in Figure 7.17. Assume that the value of δ is 0 here. The *RouteRequest* packet originated by node 5 is received by nodes 1, 2, 4, 6, and 10. Only nodes 4 and 10 find that the distance between them and the destination is less than the distance between the node 5 and the destination node; other nodes discard the *RouteRequest.* A*RouteRequest* packet is forwarded only once and the distance between the forwarding node and *D* is updated in the *RouteRequest* packet for further relaying. When node 4 forwards the *RouteRequest*packet, it updates the packet with the distance between itself and the destination node *D.* This packet, after being received at neighbor node 3, is discarded due to the fact that the distance between node 3 and the node 8 is greater than the distance between nodes 4 and 8. Once the *RouteRequest* reaches node 8, it originates a *RouteReply* packet back to the source node 5, containing the path through which future data packets are to be propagated. In order to compensate for the location error (due to the inaccuracy of GPS information or due to changes in the mobility of the nodes), a larger *RequestZone*that can accommodate the amount of error that occurred is considered.



Route establishment in LAR2.

Advantages and Disadvantages

LAR reduces the control overhead by limiting the search area for finding a path. The efficient use of geographical position information, reduced control overhead, and increased utilization of bandwidth are the major advantages of this protocol. The applicability of this protocol depends heavily on the availability of GPS infrastructure or similar sources of location information. Hence, this protocol cannot be used in situations where there is no access to such information.

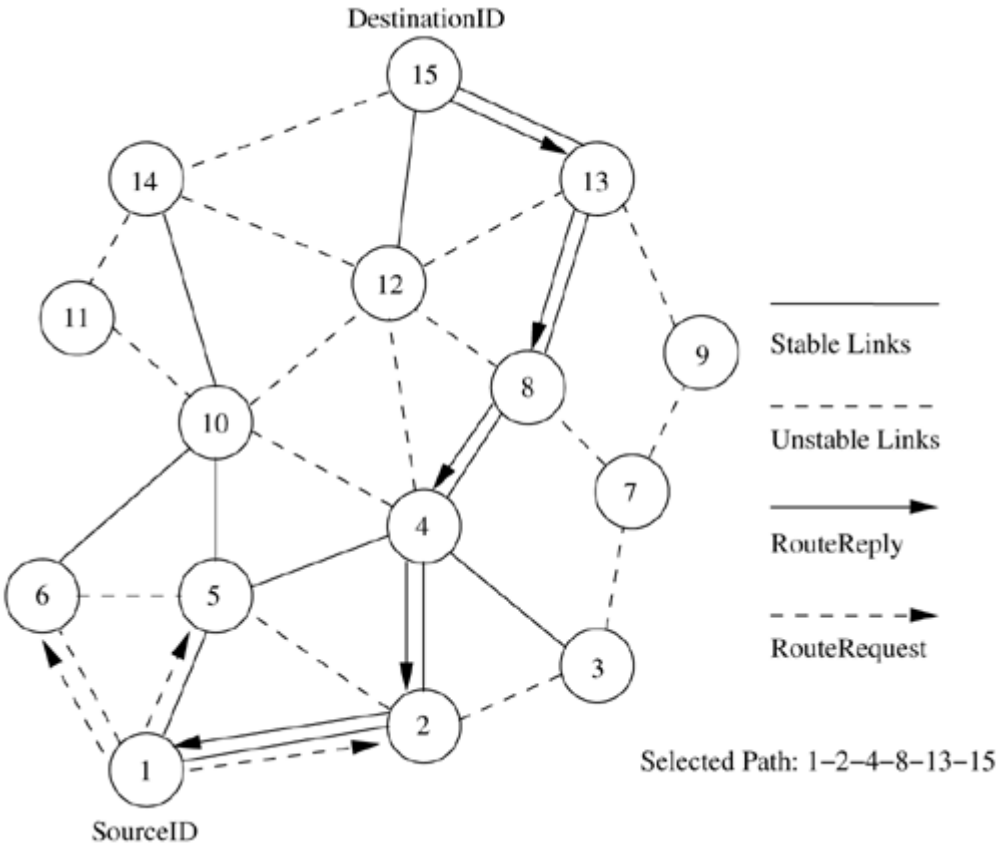
Associativity-Based Routing

Associativity-based routing (ABR) [14] protocol is a distributed routing protocol that selects routes based on the stability of the wireless links. It is a *beacon*-based, on-demand routing protocol. A link is classified as stable or unstable based on its temporal stability. The temporal stability is determined by counting the periodic *beacon*s that a node receives from its neighbors. Each

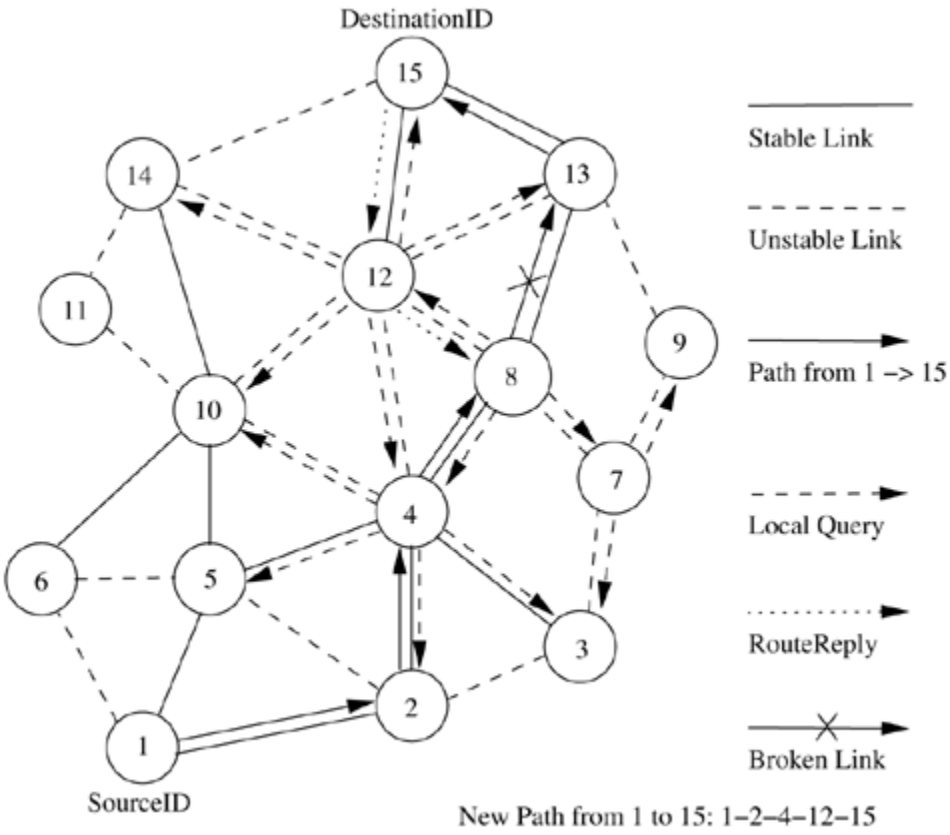
node maintains the count of its neighbors' *beacon*s and classifies each link as stable or unstable based on the *beacon* count corresponding to the neighbor node concerned. The link corresponding to a stable neighbor is termed as a stable link, while a link to an unstable neighbor is called an unstable link.

A source node floods *RouteRequest* packets throughout the network if a route is not available in its route cache. All intermediate nodes forward the *RouteRequest* packet. A *RouteRequest* packet carries the path it has traversed and the *beacon* count for the corresponding node in the path. When the first*RouteRequest* reaches the destination, the destination waits for a time period *TRouteSelectTime* to receive multiple *RouteRequest*s through different paths. After this time duration, the destination selects the path that has the maximum proportion of stable links. If two paths have the same proportion of stable links, the shorter of them is selected. If more than one shortest path is available, then a random path among them is selected as the path between source and destination.

Consider Figure 7.18, in which the source node (node 1) initiates the *RouteRequest* to be flooded for finding a route to the destination node (node 15). The solid lines represent the *stable* links that are classified based on the *beacon* count, while dotted lines represent *unstable* links. ABR does not restrict any intermediate node from forwarding a *RouteRequest* packet based on the stable or unstable link criterion. It uses stability information only during the route selection process at the destination node. As depicted in Figure 7.18, the *RouteRequest* reaches the destination through three different routes. Route 1 is 1-5-10-14-15, route 2 is 1-5-4-12-15, and route 3 is 1-2-4-8-13-15. ABR selects route 3 as it contains the highest percentage of stable links compared to route 1 and route 2. ABR gives more priority to stable routes than to shorter routes. Hence, route 3 is selected even though the length of the selected route is more than that of the other two routes.



If a link break occurs at an intermediate node, the node closer to the source, which detects the break, initiates a local route repair process. In this process, the node locally broadcasts a route repair packet, termed the local query (LQ) broadcast, with a limited time to live (TTL), as illustrated in Figure 7.19where a TTL value of 2 is used. This way a broken link is bypassed locally without flooding a new*RouteRequest* packet in the whole network. If a node fails to repair the broken link, then its uplink node (the previous node in the path which is closer to the source node) reinitiates the LQ broadcast. This route repair process continues along the intermediate nodes toward the source node until it traverses half the length of the broken path or the route is repaired. In the former case, the source node is informed, which initiates a new route establishment phase.



Consider the example in Figure 7.19. When a path break occurs between nodes 8 and 13, the node adjacent to the broken link toward the source node, that is, node 8, initiates a local query broadcast in order to locally repair the broken path. The local query has limited scope with the maximum TTL value set to the remaining path length from the broken link to the destination. In the same figure, the broken path is repaired locally by bypassing the path segment 8-13-15 through segment 8-12-15.

Advantages and Disadvantages

In ABR, stable routes have a higher preference compared to shorter routes. They result in fewer path breaks which, in turn, reduces the extent of flooding due to reconfiguration of paths in the network. One of the disadvantages of this protocol is that the chosen path may be longer than the shortest path between the source and destination because of the preference given to stable paths. Another disadvantage is that repetitive LQ broadcasts may result in high delays during route repairs.

Signal Stability-Based Adaptive Routing Protocol

Signal stability-based adaptive routing protocol (SSA) [15] is an on-demand routing protocol that uses signal stability as the prime factor for finding stable routes. This protocol is *beacon*-based, in which the signal strength of the *beacon* is measured for determining link stability. The signal strength is used to classify a link as *stable* or *unstable*. This protocol consists of two parts: forwarding protocol (FP) and dynamic routing protocol (DRP). These protocols use an extended radio interface that measures the signal strength from *beacon*s. DRP maintains the routing table by interacting with the DRP processes on other hosts. FP performs the actual routing to forward a packet on its way to the destination.

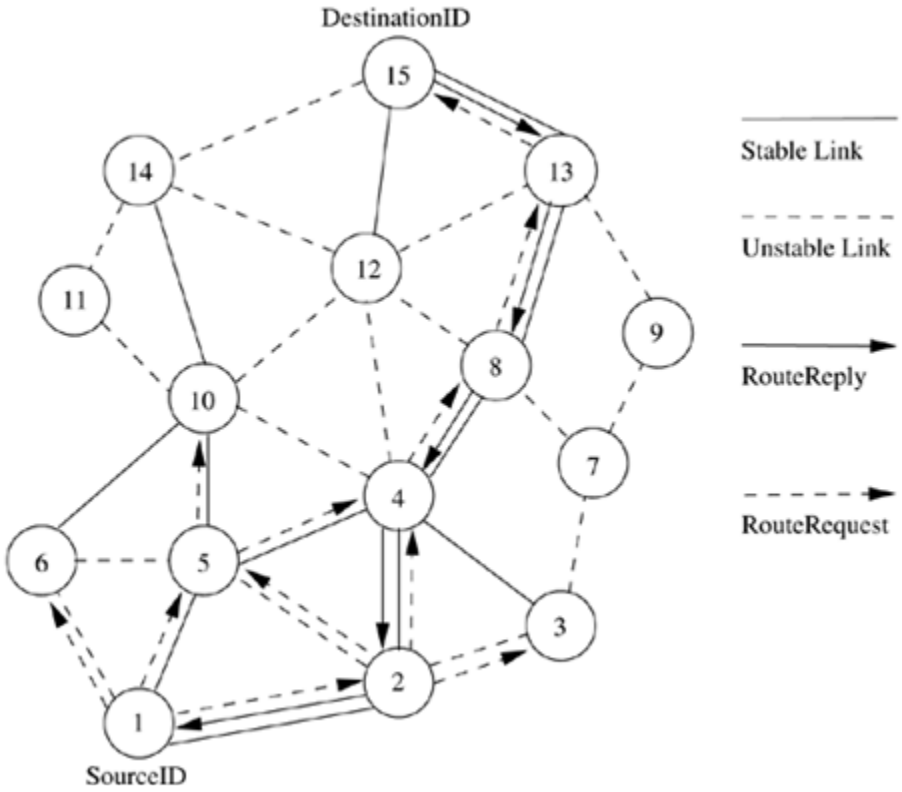
Every node maintains a table that contains the *beacon* count and the signal strength of each of its neighbors. If a node has received strong *beacon*s for the past few *beacons*, the node classifies the link as a *strong*/*stable* link. The link is otherwise classified as a *weak*/*unstable* link. Each node maintains a table called the signal stability table (SST), which is based on the signal strengths of its neighbors'*beacon*s. This table is used by the nodes in the path to the destination to forward the incoming*RouteRequest* over strong links for finding the most stable end-to-end path. If the attempt of forwarding a *RouteRequest* over the stable links fails to obtain any path to a destination, the protocol floods the *RouteRequest* throughout the network without considering the stability of links as the forwarding criterion.

A source node which does not have a route to the destination floods the network with *RouteRequest*packets. But unlike other routing protocols, nodes that employ the SSA protocol process a*RouteRequest* only if it is received over a strong link. A *RouteRequest* received through a weak link is dropped without being processed. The destination selects the first *RouteRequest* packet received over strong links. The destination initiates a *RouteReply* packet to notify the selected route to the source.

Consider Figure 7.20, where the source node (node 1) broadcasts a *RouteRequest* for finding the route to the destination node (node 15). In Figure 7.20, solid lines represent the *stable* links, while the dotted lines represent *weak* links. Unlike ABR, SSA restricts intermediate nodes from forwarding a*RouteRequest* packet if the packet had been received over a weak link. It forwards only *RouteRequest*packets received over stable links. In Figure 7.20, when the *RouteRequest* is initiated by the source, it is to be processed by all its neighbors. But before processing, each neighbor node checks whether the*RouteRequest* packet was received through a stable link. If the *RouteRequest* had been received through a stable link and had not been sent already (*i.e.*, it is not a duplicate *RouteRequest*), it is forwarded by the node; otherwise, it is dropped. For example, when the *RouteRequest* from node 1 reaches nodes 2, 5, and 6, it is forwarded only by nodes 2 and 5 as the link between nodes 1 and 6 is weak. Similarly, the *RouteRequest* forwarded by node 2 is rejected by nodes 3 and 5, while node 4 forwards it to its neighbors, provided it has not already been forwarded. In this way, nodes forward

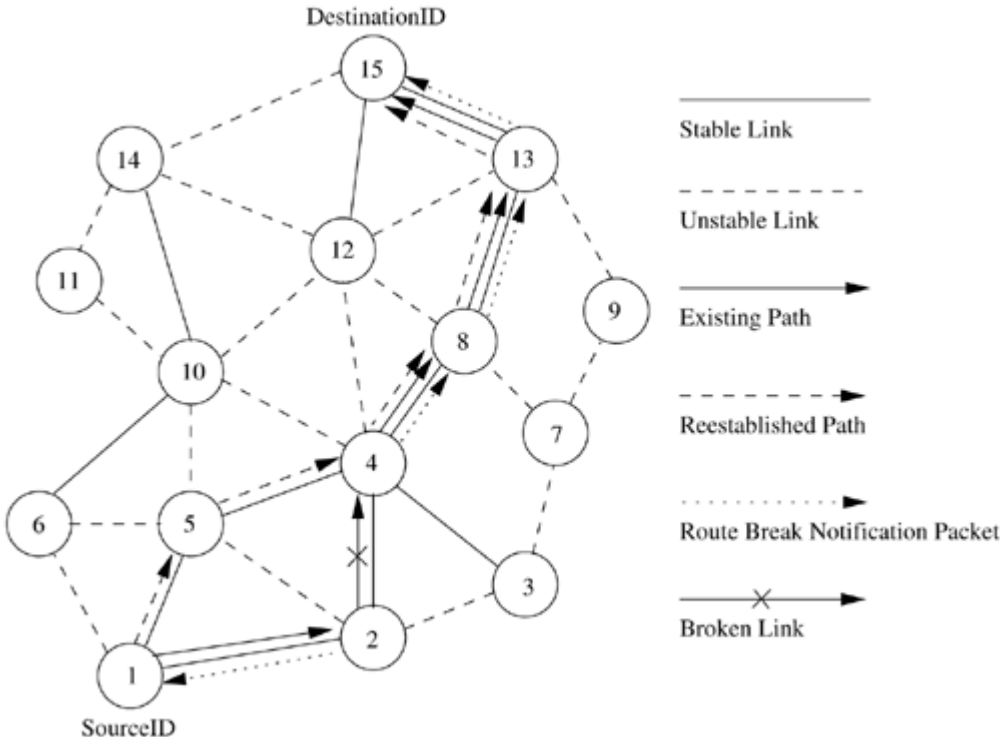
the*RouteRequest* until it reaches the destination. In this example, only one route is established based on the strong link criterion. A *RouteRequest* propagated through path 1-5-10-14-15 is rejected by the destination as it receives a *RouteRequest* from node 14 on a weak link. The stable path consisting of strong links is 1- 2 (or 5)-4-8-13-15. The first *RouteRequest* packet that reaches the destination over a stable path is selected by the destination.

Route establishment in SSA.



As shown in Figure 7.21, when a link breaks, the end nodes of the broken link (*i. e.*, nodes 2 and 4) notify the corresponding end nodes of the path (*i. e.*, nodes 1 and 15). A source node, after receiving a route break notification packet, rebroadcasts the *RouteRequest* to find another stable path to the destination. Stale entries are removed only if data packets that use the stale route information fail to reach the next node. If the link between nodes 2 and 4 breaks, a new strong path is established through 1-5-4-8-13-15. If no strong path is available when a link gets broken (*e.g.*, link 8-13), then the new route is established by considering weak links also. This is done when multiple *RouteRequest* attempts fail to obtain a path to the destination using only the stable links

Route maintenance in SSA.



Advantages and Disadvantages

The main advantage of this protocol is that it finds more stable routes when compared to the shortest path route selection protocols such as DSR and AODV. This protocol accommodates temporal stability by using *beacon* counts to classify a link as stable or weak. The main disadvantage of this protocol is that it puts a strong *RouteRequest* forwarding condition which results in *RouteRequest* failures. A failed *RouteRequest* attempt initiates a similar path-finding process for a new path without considering the stability criterion. Such multiple flooding of *RouteRequest* packets consumes a significant amount of bandwidth in the already bandwidth-constrained network, and also increases the path setup time. Another disadvantage is that the strong links criterion increases the path length, as shorter paths may be ignored for more stable paths.

Flow-Oriented Routing Protocol

Flow-oriented routing protocol (FORP) [16] is an on-demand routing protocol that employs a prediction-based *multi-hop-handoff* mechanism for supporting time-sensitive traffic in ad hoc wireless networks. This protocol has been proposed for IPv6-based ad hoc wireless networks where quality of service (QoS) needs to be provided. The *multi-hop-handoff* is aimed at alleviating the effects of path breaks on the real-time packet flows. A sender or an intermediate node initiates the route maintenance process only after detecting a link break. This reactive route maintenance procedure may result in high packet loss,

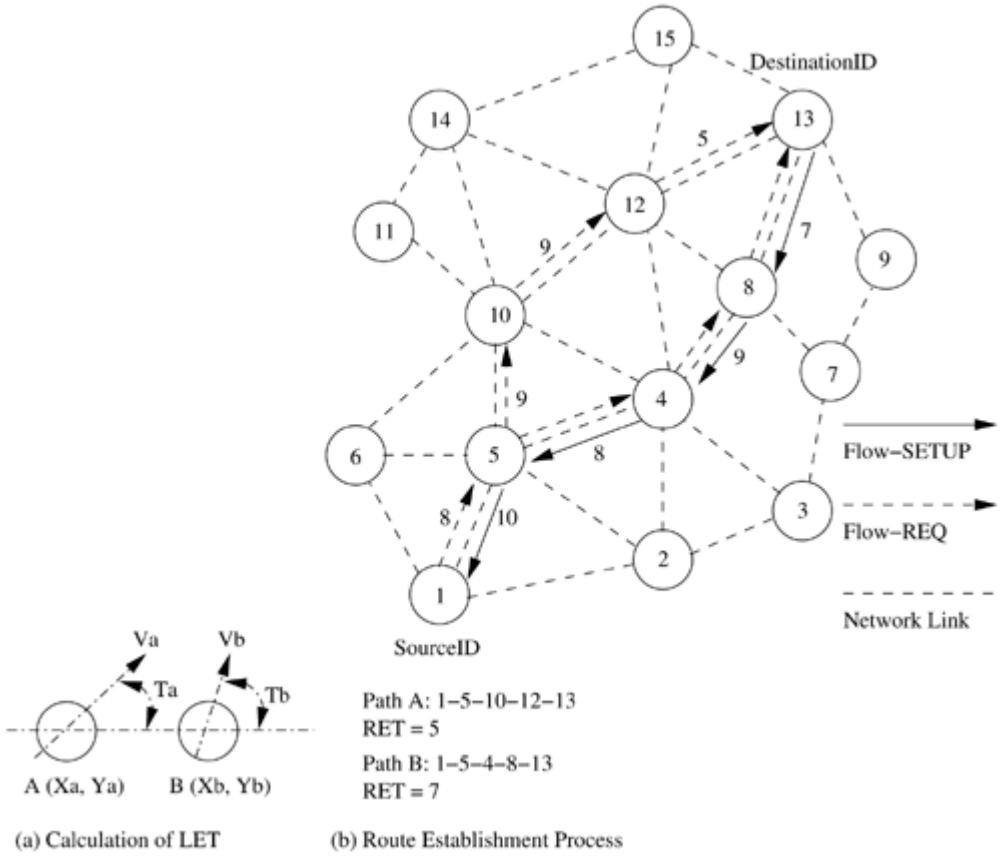
leading to a low quality of service provided to the user. FORP uses a unique prediction-based mechanism that utilizes the mobility and location information of nodes to estimate the link expiration time (LET). LET is the approximate lifetime of a given wireless link. The minimum of theLET values of all wireless links on a path is termed as the route expiry time (RET). Every node is assumed to be able to predict the LET of each of its links with its neighbors.

The LET between two nodes can be estimated using information such as current position of the nodes, their direction of movement, and their transmission ranges. FORP requires the availability of GPS information in order to identify the location of nodes.

When a sender node needs to set up a real-time flow to a particular destination, it checks its routing table for the availability of a route to that destination. If a route is available, then that is used to send packets to the destination. Otherwise, the sender broadcasts a *Flow-REQ* packet carrying information regarding the source and the destination nodes. The *Flow-REQ* packet also carries a flow identification number/sequence number which is unique for every session. A neighbor node, on receiving this packet, first checks if the sequence number of the received *Flow-REQ* is higher than the sequence number corresponding to a packet belonging to the same session that had been previously forwarded by the node. If so, then it updates its address on the packet and extracts the necessary state information out of the packet. If the sequence number on the packet is less than that of the previously forwarded packet, then the packet is discarded. This is done to avoid looping of *Flow-REQ* packets. A *Flow-REQ* with the same sequence number as that of a *Flow-REQ* belonging to the same session which had been forwarded already by the node, would be broadcast further only if it has arrived through a shorter (and therefore better) path. Before forwarding a *Flow-REQ*, the intermediate node appends its node address and the LET of the last link the packet had traversed onto the packet. The *Flow-REQ* packet, when received at the destination node, contains the list of nodes on the path it had traversed, along with theLET values of every wireless link on that path. FORP assumes all the nodes in the network to be synchronized to a common time by means of GPS information. If the calculated value of RET, corresponding to the new *Flow-REQ* packet arrived at the destination, is better than the RET value of the path currently being used, then the destination originates a *Flow-SETUP* packet. The LET of a link can be estimated given the information about the location, velocity, and transmission range of the nodes concerned. The LET of the wireless link between two nodes *a* and *b* with transmission range *Tx*, which are moving at velocity *Va* and *Vb* at angles *Ta* and *Tb* , respectively (refer to Figure 7.22 (a)), can be estimated as described below:

$$LET_{ab} = \frac{-(pq + rs) + (p^2 + r^2){T_x}^2 - (ps - qr)^2}{p^2 + q^2}$$

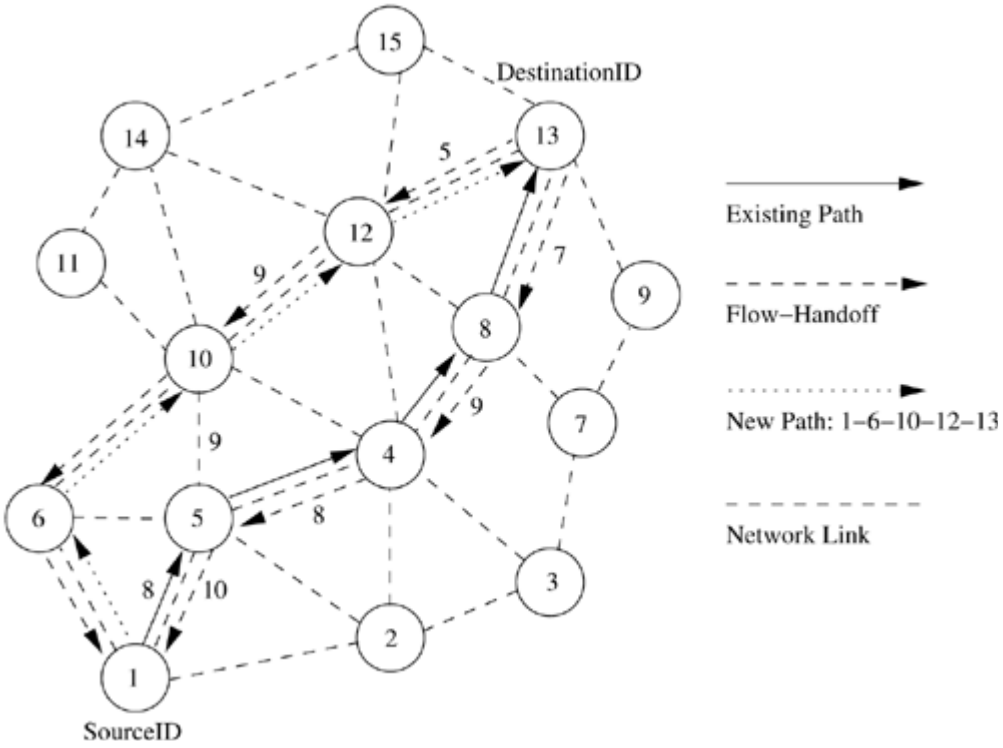
Route establishment in FORP



The route establishment procedure is shown in Figure 7.22 (b). In this case, the path 1-5-4-8-13 (path 1) has a RET value of 7, whereas the path 1-5-10-12-13 (path 2) has a RET value of 5. This indicates that path 1 may last longer than path 2. Hence the sender node originates a *Flow-SETUP* through the reverse path 13-8-4-5-1.

FORP employs a proactive route maintenance mechanism which makes use of the expected RET of the current path available at the destination. Route maintenance is illustrated in Figure 7.23. When the destination node determines (using the RET of the current path) that a route break is about to occur within a critical time period (*tc*), it originates a *Flow-HANDOFF* packet to the source node, which is forwarded by the intermediate nodes. The mechanism by which *Flow-HANDOFF* packets are forwarded is similar to the *Flow-REQ* forwarding mechanism. When many *Flow-HANDOFF* packets arrive at the source node, the source node calculates the RET values of paths taken by each of them, selects the best path, and uses this new path for sending packets to the destination. In the example shown in Figure 7.23, the *Flow-HANDOFF* packets are forwarded by every intermediate node after appending the LET information of the previous link traversed onto the packet. The existing path 1-5-4-8-13 is erased and a new path is selected by the source node based on the RETs corresponding to different paths traversed by the *Flow-HANDOFF* packets. In this case, the path 1-6-10-12-13 is chosen. The critical time (*tc*) is taken as the difference between the RET and delay encountered by

the latest packet which has traversed through the existing path from the source to the destination.



Advantages and Disadvantages

The use of LET and RET estimates reduces path breaks and their associated ill effects such as reduction in packet delivery, increase in the number of out-oforder packets, and non-optimal paths resulting from local reconfiguration attempts. The proactive route reconfiguration mechanism adopted here works well when the topology is highly dynamic. The requirements of time synchronization increases the control overhead. Dependency on the GPS infrastructure affects the operability of this protocol in environments where such infrastructure may not be available.

HYBRID ROUTING PROTOCOLS

In this section, we discuss the working of routing protocols termed as hybrid routing protocols. Here, each node maintains the network topology information up to *m* hops. The different existing hybrid protocols are presented below.

7.6.1 Core Extraction Distributed Ad Hoc Routing Protocol

Core extraction distributed ad hoc routing (CEDAR) [17] integrates routing and support for QoS. It is based on extracting core nodes (also called as dominator nodes) in the network, which together approximate the minimum dominating set. A dominating set (DS) of a graph is defined as a set of nodes in the graph such that every node in the graph is either present in the DS or is a neighbor of some node present in the DS. There exists at least one core node within three

hops. The DS of the least cardinality in a graph is called the minimum dominating set. Nodes that choose a core node as their dominating node are called core member nodes of the core node concerned. The path between two core nodes is termed a virtual link. CEDAR employs a distributed algorithm to select core nodes. The selection of core nodes represents the core extraction phase.

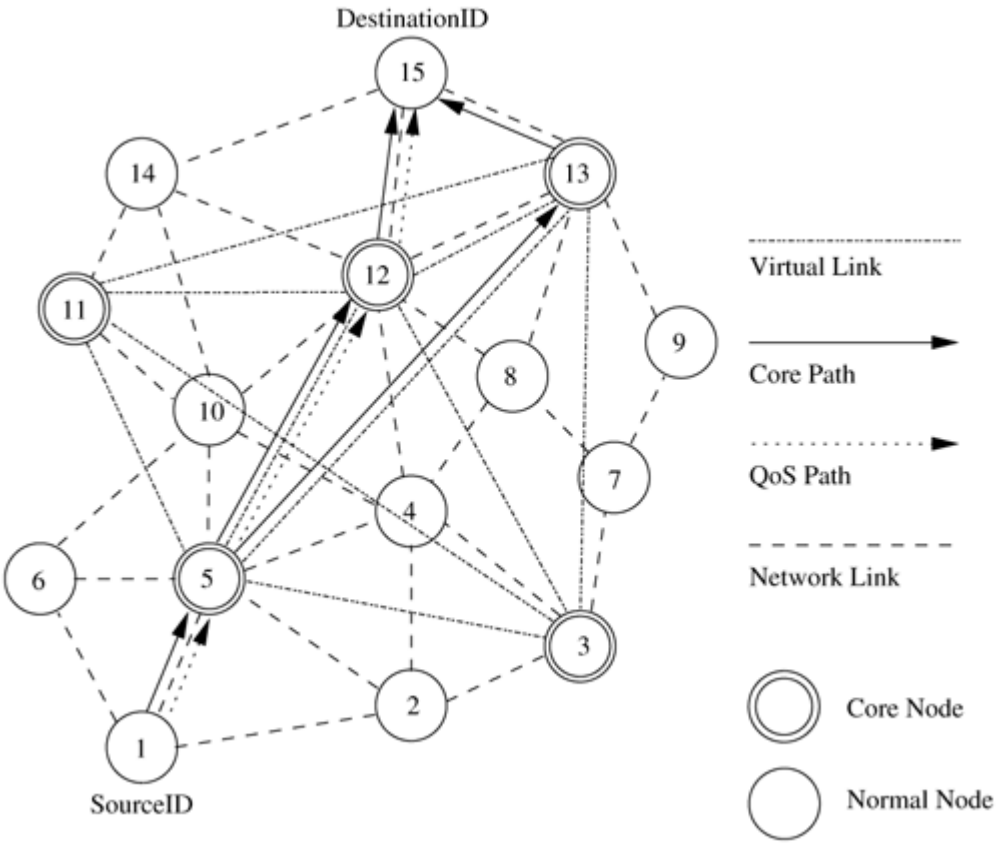
CEDAR uses the core broadcast mechanism to transmit any packet throughout the network in the unicast mode, involving as minimum number of nodes as possible. These nodes that take part in the core broadcast process are called core nodes. In order to carry out a core broadcast efficiently, each core node must know about its neighboring core nodes. The transfer of information about neighboring core nodes results in significant control overhead at high mobility. When a core node to which many nodes are attached moves away from them, each node has to reselect a new core node. The selection of core nodes, which is similar to the distributed leader election process, involves substantial control overhead.

Each core node maintains its neighborhood local topology information. CEDAR employs an efficient link state propagation mechanism in which information regarding the presence of high bandwidth and stable links is propagated to several more nodes, compared to the propagation of information regarding low bandwidth and unstable links, which is suppressed and kept local. To propagate link information, slow-moving increase-waves and fast-moving decrease-waves are used. An increase-wave carrying update information is originated when the bandwidth on the link concerned increases above a certain threshold. The fast-moving decrease-waves are propagated in order to quickly notify the nearby nodes (core nodes which are at most separated by three hops) about the reduction in available bandwidth. As bandwidth increase information moves slowly, only stable high-bandwidth link state information traverses long distances. If the high-bandwidth link is unstable, then the corresponding increase-wave is overtaken by fast-moving decrease-waves which represent the decrease in available bandwidth on the corresponding link. These waves are very adaptive to the dynamic topology of ad hoc wireless networks. Increaseand decrease-waves are initiated only when changes in link capacity cross certain thresholds, that is, only when there is a significant change in link capacity. Fast-moving decrease-waves are prevented from moving across the entire network, thereby suppressing low bandwidth unstable information to the local nodes only. Route establishment in CEDAR is carried out in two phases. The first phase finds a core path from the source to the destination. The core path is defined as a path from the dominator of the source node (source core) to the dominator of the destination node (destination core). In the second phase, a QoS feasible path is found over the core path. A node initiates a *RouteRequest* if the destination is not in the local topology table of its core node; otherwise, the path is immediately established. For establishing a route, the source core initiates a core broadcast in which the *RouteRequest* is sent to all neighboring core nodes as unicast data. Each of these recipient core nodes in turn forwards the *RouteRequest* to its neighboring core nodes if the destination is not its core

member. A core node which has the destination node as its core member replies to the source core. Once the core path is established, a path with the required QoS support is then chosen.
To find a path that can provide the required QoS, the source core first finds a path to the domain of the farthest possible core node in the core path, which can provide the bandwidth required. Among the available paths to this domain, the source core chooses the shortest-widest path (shortest path with highest bandwidth). Assume *MidCore* is the farthest possible core node found by the source core. In the next iteration, *MidCore* becomes the new source core and finds another *MidCore* node that satisfies the QoS support requirements. This iterative process repeats until a path to the destination with the required bandwidth is found. If no feasible path is available, the source node is informed about the non-availability of a QoS path

Consider Figure 7.24 where the source is node 1 and the destination is node 15. The core nodes in the network are nodes 3, 5, 11, 12, and 13. In this figure, node 5 is the dominator of nodes 1 and 6. Similarly, node 12 is the dominator of node 15. When node 1 initiates a *RouteRequest* to be flooded throughout the network, it intimates its core node the <source id, destination id> pair information. If the core node 5 does not have any information about the dominator of node 15, which is the destination node, it initiates a core broadcast. Due to this, all nearby core nodes receive the request in the unicast transmission mode. This unicast transmission is done on the virtual links. For core node 5, the virtual link with core node 3 comprises of the links 5-2 and 2-3, while the virtual link between core nodes 5 and 13 is represented by path 5-4-8-13. When a core node receives the core broadcast message, it checks whether the destination is its core member. A core node having the destination as one of its core members replies to the source core node. In our case, core node 12 replies to core node 5. The path between core nodes 12 and 5 constitutes a core path. Once a core path is established, the feasibility of the path in terms of the availability of the required bandwidth is checked. If the required bandwidth is available on the path, the connection is established; otherwise, the core path is rejected.

Route establishment in CEDAR.

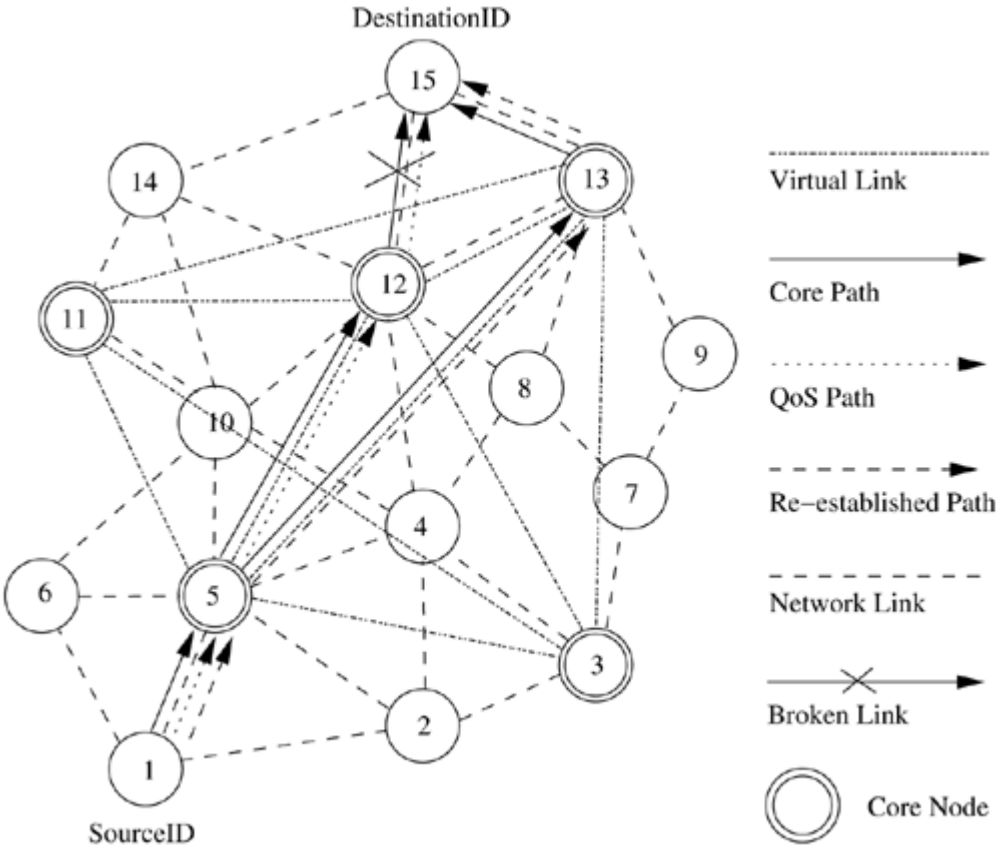


CEDAR attempts to repair a broken route locally when a path break occurs. When a link u-v on the path from source to the destination fails, node u sends back a notification to the source and starts recomputation of a route from itself to the destination node. Until the route recomputation gets completed, node u drops every subsequent packet it receives. Once the source node receives the notification sent by node u, it immediately stops transmitting packets belonging to the corresponding flow, and starts computing a new route to the destination. If the link break occurs near the source, route recomputation at node u may take a long time, but the notification sent by node u reaches the source node very rapidly and prevents large packet losses. If the broken link is very close to the destination, the notification sent by node u might take a longer time to reach the source, but the route recomputation time at node u is small and hence large packet losses are again prevented. If the link break occurs somewhere near the middle of the path, then both the local route recomputation mechanism and the route break notification mechanism are not fast enough, and hence a considerable amount of packet loss occurs in this case.

Consider the network topology shown in Figure 7.25. When the link between nodes 12 and 15 breaks, node 12 tries to reconnect to the destination using an alternate path that satisfies the bandwidth requirement. It also notifies the source node about the link break. The source node tries to reconnect to the destination by reinitiating the route establishment process. In case node 12 does not have any other feasible path, then the alternate path 1-5-4-8-13-15 found by

the source node is used for the further routing of packets.

Route maintenance in CEDAR.

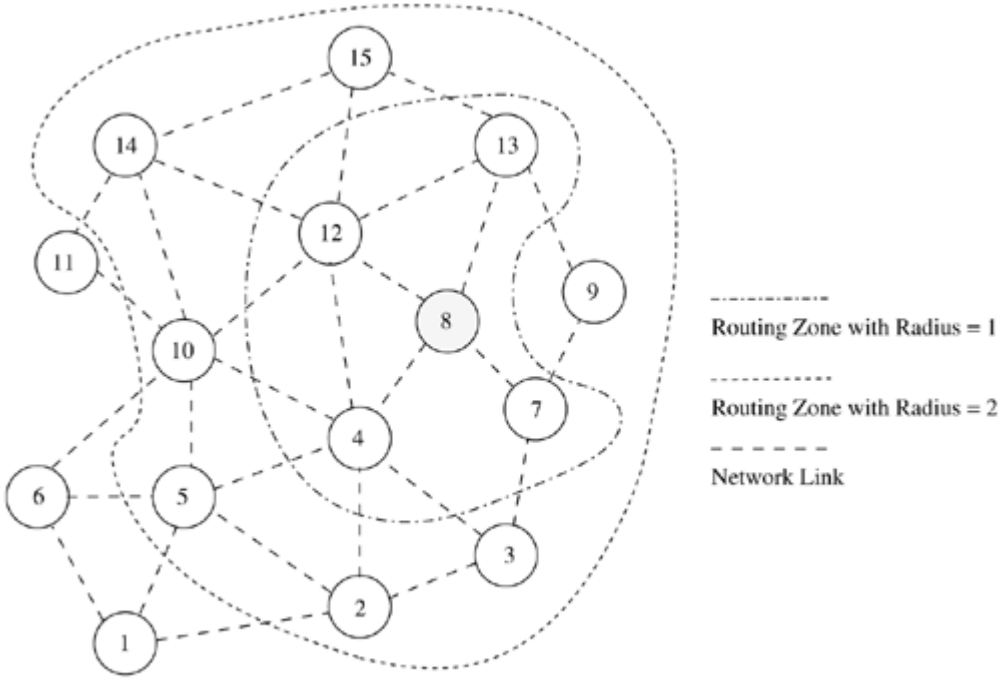


Advantages and Disadvantages
The main advantage of CEDAR is that it performs both routing and QoS path computation very efficiently with the help of core nodes. The increase- and decrease-waves help in appropriate propagation of the stable high-bandwidth link information and the unstable low-bandwidth link information, respectively. Core broadcasts provide a reliable mechanism for establishing paths with QoS support. A disadvantage of this protocol is that since route computation is carried out at the core nodes only, the movement of the core nodes adversely affects the performance of the protocol. Also, the core node update information could cause a significant amount of control overhead

Zone Routing Protocol

Zone routing protocol (ZRP) [18] is a hybrid routing protocol which effectively combines the best features of both proactive and reactive routing protocols. The key concept employed in this protocol is to use a proactive routing scheme within a limited zone in the *r*-hop neighborhood of every node, and use a

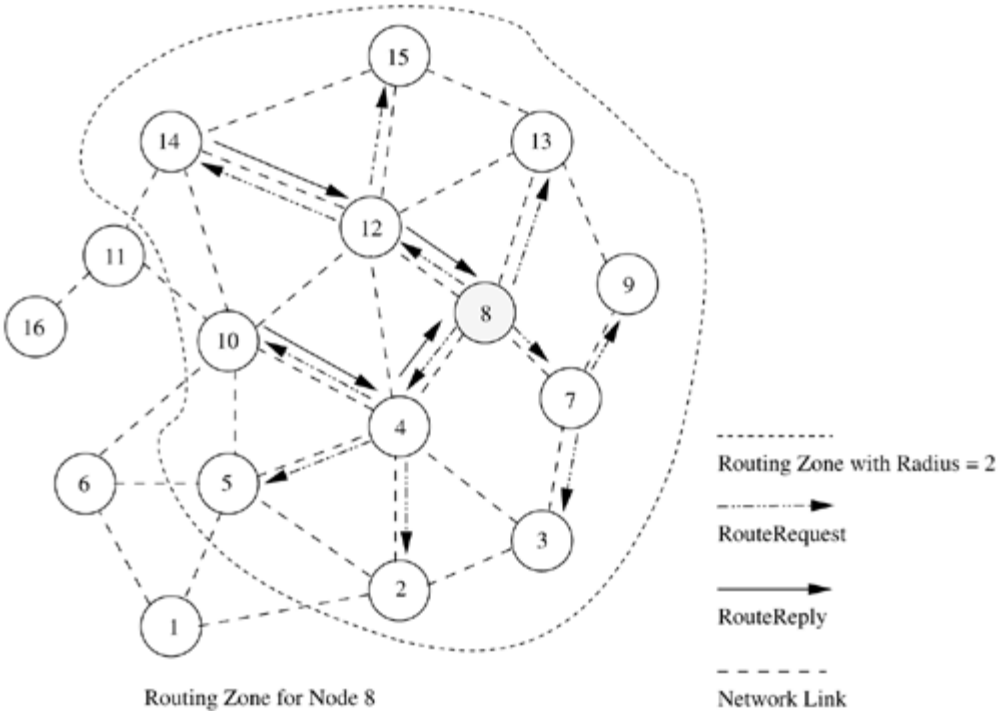
reactive routing scheme for nodes beyond this zone. An *intra-zone routing protocol* (IARP) is used in the zone where a particular node employs proactive routing. The reactive routing protocol used beyond this zone is referred to as *inter-zone routing protocol* (IERP). The *routing zone* of a given node is a subset of the network, within which all nodes are reachable within less than or equal to *zone radius*hops. Figure 7.26 illustrates *routing zones* of node 8, with r = 1 hop and r = 2 hops. With *zone radius = 2*, the nodes 7, 4, 12, and 13 are interior nodes, whereas nodes 2, 3, 5, 9, 10, 13, and 15 are peripheral nodes (nodes with the shortest distance equal to the *zone radius*). Each node maintains the information about routes to all nodes within its *routing zone* by exchanging periodic route update packets (part of IARP). Hence the larger the *routing zone*, the higher the update control traffic.



Routing Zone for Node 8

The IERP is responsible for finding paths to the nodes which are not within the *routing zone*. IERPeffectively uses the information available at every node's *routing zone*. When a node *s* (node 8 in Figure 7.27) has packets to be sent to a destination node *d* (node 15 in Figure 7.27), it checks whether node *d* is within its zone. If the destination belongs to its own zone, then it delivers the packet directly. Otherwise, node *s* bordercasts (uses unicast routing to deliver packets directly to the border nodes) the*RouteRequest* to its peripheral nodes. In Figure 7.27 node 8 bordercasts *RouteRequest*s to nodes 2, 3, 5, 7, 9, 10, 13, 14, and 15. If any peripheral node finds node *d* to be located within its *routing zone*, it sends a *RouteReply* back to node *s* indicating the path; otherwise, the

node rebordercasts the*RouteRequest* packet to the peripheral nodes. This process continues until node *d* is located. Nodes 10 and 14 find the information about node 16 to be available in their intra-zone routing tables, and hence they originate *RouteReply* packets back to node 8. During *RouteRequest* propagation, every node that forwards the *RouteRequest* appends its address to it. This information is used for delivering the*RouteReply* packet back to the source. The path-finding process may result in multiple *RouteReply*packets reaching the source, in which case the source node can choose the best path among them. The criterion for selecting the best path may be the shortest path, least delay path, etc.



Routing Zone for Node 8

Path finding between node 8 and node 16.

When an intermediate node in an active path detects a broken link in the path, it performs a local path reconfiguration in which the broken link is bypassed by means of a short alternate path connecting the ends of the broken link. A path update message is then sent to the sender node to inform it about the change in path. This results in a sub-optimal path between two end points, but achieves quick reconfiguration in case of link failures. To obtain an optimal path, the sender reinitiates the global path-finding process after a number of local reconfigurations.

Advantages and Disadvantages

By combining the best features of proactive and reactive routing schemes, ZRP reduces the control overhead compared to the *RouteRequest* flooding mechanism employed in on-demand approaches and

the periodic flooding of routing information packets in table-driven approaches. But in the absence of a query control, ZRP tends to produce higher control overhead than the aforementioned schemes. This can happen due to the large overlapping of nodes' *routing zones*. The query control must ensure that redundant or duplicate *RouteRequest*s are not forwarded. Also, the decision on the zone radius has a significant impact on the performance of the protocol.

Zone-Based Hierarchical Link State Routing Protocol

Zone-based hierarchical link state (ZHLS) routing protocol [19] is a hybrid hierarchical routing protocol that uses the geographical location information of the nodes to form non-overlapping zones. A hierarchical addressing that consists of a zone ID and a node ID is employed. Each node requires its location information, based on which it can obtain its zone ID. The information about topology inside a zone is maintained at every node inside the zone, and for regions outside the zone, only the zone connectivity information is maintained. ZHLS maintains high-level hierarchy for inter-zone routing. Packet forwarding is aided by the hierarchical address comprising of the zone ID and node ID. Similar to ZRP, ZHLS also employs a proactive approach inside the geographical zone and a reactive approach beyond the zone. A destination node's current location is identified by the zone ID of the zone in which it is present and is obtained by a route search mechanism.

In ZHLS, every node requires GPS or similar infrastructure support for obtaining its own geographical location that is used to map itself onto the corresponding zone. The assignment of zone addresses to geographical areas is important and is done during a phase called the network design phase or network deployment phase. The area of the zone is determined by several factors such as the coverage of a single node, application scenario, mobility of the nodes, and size of the network. For example, the ad hoc network formed by a set of hand-held devices with a limited mobility may require a zone radius of a few hundred meters, whereas the zone area required in the network formed by a set of ships, airplanes, or military tanks may be much larger.

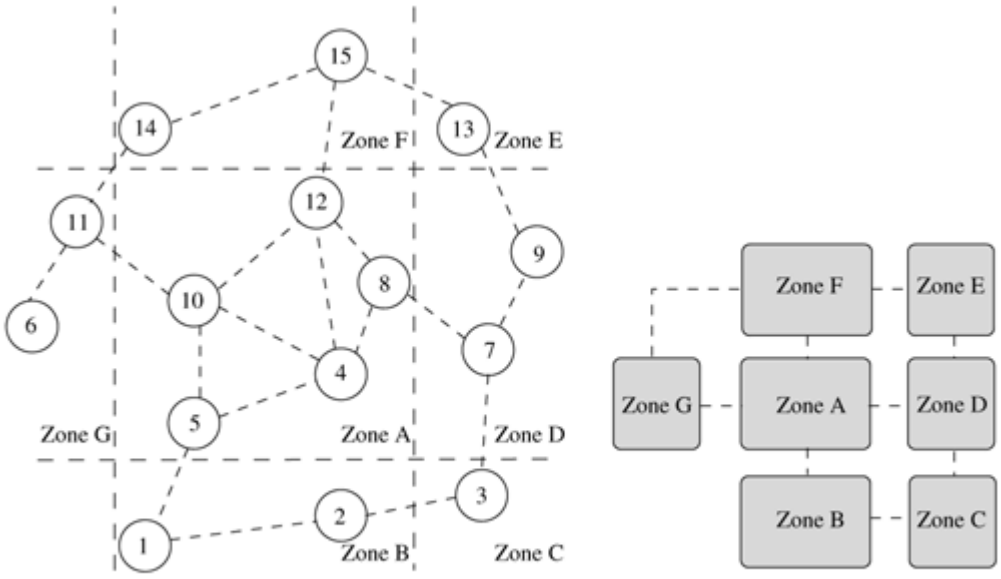
The intra-zone routing table is updated by executing the shortest path algorithm on the node-level topology of the zone. The node-level topology is obtained by using the *intra-zone clustering*mechanism, which is similar to the link state updates limited to the nodes present in the zone. Each node builds a one-hop topology by means of a link request and link response mechanism. Once the one-hop topology is available, each node prepares link state packets and propagates them to all nodes in the zone. These update packets contain the node IDs of all nodes that belong to the zone, and node IDs and zone IDs of all nodes that belong to other zones. The nodes that receive link responses from nodes belonging to other zones are called *gateway* nodes. Data traffic between two zones will be relayed through these gateway nodes. For example, nodes 5, 8, 10, and 12 are the gateway nodes for zone A in Figure 7.28 (a). Every node in a zone is aware of the neighboring zones connected to its zone and the gateway nodes to be used for reaching them. Once the node-level link state packets are

exchanged and the node-level topology is updated, every node in a zone generates a zone link state packet. The zone link state packet contains information about the zone-level connectivity. These zone link state packets are propagated throughout the network by the gateway nodes. The zone-level topology is shown in Figure 7.28 (b). The zone link state packets originated by every zone are shown inTable 7.1. Using the information obtained from zone link state packets, a node can build the zone topology. The zone routing table can be formed for any destination zone by executing the shortest path algorithm on the zone-level topology. The zone link state packets are source sequence numbered and a time-stamp field is included to avoid stale link state packets. The association of the nodes to the respective zones helps in reducing routing overhead as in ZRP, but it includes the additional requirement of determining a given destination node's present location. If a source node Src wants to communicate with a destination node Dest, Src checks whether Dest resides in its own zone. If Dest belongs to the same zone, then packets are delivered to Dest as per the intra-zone routing table. If the destination Dest does not belong to the zone, then the node Src originates a location request packet containing the sender's and destination's information. This location request packet is forwarded to every other zone. The gateway node of a zone at which the location request packet is received verifies its routing table for the destination node for which the location request was originated. The gateway node that finds the destination node required by a location request packet originates a location response packet containing the zone information to the sender.

Zone link state packets

Source Zone	Zone Link State Packet
A	B, D, F, and G
B	C and A
C	B and D
D	A, C, and E
E	A, D, and F
F	A, E, and G
G	A and F

Zone-based hierarchical link state routing protocol.



(a) Node–level topology in ZHLS

(b) Zone topology for the node–level topology in (a)

Route maintenance is easier with the presence of multiple gateway nodes between zones. If a given gateway node moves away, causing a zone-level connection failure, routing can still take place with the help of the other gateway nodes. This is due to the hierarchical addressing that makes use of zone ID and node ID. At any intermediate zone, with the most updated inter-zonal routing table, it forwards the data packets.

Advantages and Disadvantages

The hierarchical approach used in this protocol significantly reduces the storage requirements and the communication overhead created because of mobility. The zone-level topology is robust and resilient to path breaks due to mobility of nodes. Intra-zonal topology changes do not generate network-wide control packet transmissions. A main disadvantage of this protocol is the additional overhead incurred in the creation of the zone-level topology. Also the path to the destination is sub-optimal. The geographical information required for the creation of the zone level topology may not be available in all environments.

7.7 ROUTING PROTOCOLS WITH EFFICIENT FLOODING MECHANISMS

Many of the existing on-demand routing protocols employ flooding of *RouteRequest* packets in order to obtain a feasible path with the required packet-forwarding constraints. Flooding of control packets results in a significant amount of redundancy, wastage of bandwidth, increase in number of collisions, and broadcast storms1 at times of frequent topological changes. Existing routing protocols that employ efficient flooding mechanisms to counter

the requirement of flooding include preferred link-based routing (PLBR) protocols [20] and optimized link state routing (OLSR) protocol [21]. The former belongs to the on-demand routing protocols category and the latter belongs to the table-driven routing protocols category. These protocols utilize algorithms that require a minimum number of transmissions in order to flood the entire network.

1 Broadcast storm refers to the presence/origination of a large number of broadcast control packets for routing due to the high topological instability occurring in the network as a result of mobility.

7.7.1 Preferred Link-Based Routing Protocols

SSA [15] uses the preferred link approach in an implicit manner by processing a *RouteRequest* packet only if it is received through a strong link. Wired networks also employ preferred links mechanisms [22], but restrict themselves by selecting a single preferred link, based on heuristics that satisfy multiple constraints, for example, minimum cost and least delay required by the route. In ad hoc networks, the single preferred link model is not suitable due to reasons such as lack of topology information, continuously changing link characteristics, broadcast nature of the radio channel, and mobility of nodes. Sisodia *et al.* proposed two algorithms in [20] known as preferred link-based routing (PLBR) protocols that employ a different approach in which a node selects a subset of nodes from its neighbors list (*NL*). This subset is referred to as the *preferred list* (*PL*). Selection of this subset may be based on link or node characteristics. Every *RouteRequest* packet carries the list of a selected subset of neighbors. All neighbors receive *RouteRequest* packets because of the broadcast radio channel, but only neighbors present in the PL forward them further. The packet is forwarded by *K* neighbors, where *K* is the maximum number of neighbors allowed in the PL. PLBR aims to minimize control overhead in the ad hoc wireless network. All nodes operate in the promiscuous mode. Each node maintains information about its neighbors and their neighbors in a table called neighbor's neighbor table (*NNT*). It periodically transmits a *beacon* containing the changed neighbor's information. PLBR has three main phases: route establishment, route selection, and route maintenance.

The route establishment phase starts when a source node (Src) receives packets from the application layer, meant for a destination node (Dest) for which no route is currently available. If Dest is in Src's*NNT*, the route is established directly. Otherwise, Src transmits a *RouteRequest* packet containing the source node's address (SrcID), destination node's address (DestID), a unique sequence number (SeqNum) (which prevents formation of loops and forwarding of multiple copies of the same*RouteRequest* packet received from different neighbors), a traversed path (*TP*) (containing the list of nodes through which the packet has traversed so far and the weight assigned to the associated links), and a PL. It also contains a time to live (*TTL*) field that is used to avoid packets being present in the network forever, and a *NoDelay* flag, the use of which will be described later in this section. Before forwarding a *RouteRequest* packet, each eligible node recomputes the preferred list table (*PLT*) that contains the list of neighbors in the order of preference. The node inserts the first *K* entries of

the *PLT*onto the *PL* field of the *RouteRequest* packet (*K* is a global parameter that indicates the maximum size of *PL*). The old *PL* of a received packet is replaced every time with a new *PL* by the forwarding node. A node is eligible for forwarding a *RouteRequest* only if it satisfies all the following criteria: the node ID must be present in the received *RouteRequest* packet's *PL*, the *RouteRequest* packet must not have been already forwarded by the node, and the *TTL* on the packet must be greater than zero. If Dest is in the eligible node's *NNT*, the *RouteRequest* is forwarded as a unicast packet to the neighbor, which might either be Dest or whose *NL* contains Dest. If there are multiple neighbors whose *NL* have Dest, the *RouteRequest* is forwarded to only one randomly selected neighbor. Otherwise, the packet is broadcast with a new *PL* computed from the node's *NNT*. *PLT* is computed by means of one of the two algorithms discussed later in this section. If the computed *PLT* is empty, that is, there are no eligible neighbors, the *RouteRequest* packet is discarded and marked as *sent*. If the *RouteRequest* packet reaches the destination, the route is selected by the route selection procedure given below.

When multiple *RouteRequest* packets reach Dest, the route selection procedure selects the best route among them. The criterion for selecting the best route can be the shortest path, or the least delay path, or the most stable path. Dest starts a timer after receiving the first *RouteRequest* packet. The timer expires after a certain *RouteSelectWait* period, after which no more *RouteRequest* packets would be accepted. From the received *RouteRequest* packets, a route is selected as follows.

For every *RouteRequest i* that reached Dest during the *RouteSelectWait* period, Max( ) is selected, where is the minimum weight of a link in the path followed by *i*. If two or more paths have the same value for *Max* ( ), the shortest path is selected. After selecting a route, all subsequent*RouteRequest* packets from the same Src with a SeqNum less than or equal to the SeqNum of the selected *RouteRequest* are discarded. If the *NoDelay* flag is set, the route selection procedure is omitted and *TP* of the first *RouteRequest* reaching the Dest is selected as the route. The *NoDelay* flag can be set if a fast connection setup is needed.

Mobility of nodes causes frequent path breaks that should be locally repaired to reduce broadcast of the*RouteRequest*. The local route repair broadcast mechanism used in ABR [14] has a high failure rate due to the use of restricted *TTL*, which increases the average delay in route reestablishment. PLBR uses a quick route repair mechanism which bypasses the down link (Dest side) node from the broken path, using information about the next two hops from *NNT*.

Algorithms for Preferred Links Computation

Two different algorithms have been proposed by Sisodia *et al.* in [20], for finding preferred links. The first algorithm selects the route based on degree of neighbor nodes (degree of a node is the number of neighbors). Preference is given to nodes whose neighbor degree is higher. As higher degree neighbors cover more nodes, only a few of them are required to cover all the nodes of the *NNT*. This reduces the number of broadcasts. The second algorithm gives

preference to stable links. Links are not explicitly classified as stable or unstable. The notion of stability is based on the weight given to links.

Neighbor Degree-Based Preferred Link Algorithm (NDPL)

Let *d* be the node that calculates the preferred list table *PLT*. *TP* is the traversed path and *OLDPL* is the preferred list of the received *RouteRequest* packet. The *NNT* of node *d* is denoted by *NNTd* . *N*(*i*) denotes the neighbors of node *i* and itself. Include list (*INL*) is a set containing all neighbors reachable by transmitting the *RouteRequest* packet after execution of the algorithm, and the exclude list (*EXL*) is a set containing all neighbors that are unreachable by transmitting the *RouteRequest* packet after execution of the algorithm.

1. In this step, node *d* marks the nodes that are not eligible for further forwarding the *RouteRequest*packet.

1. If a node *i* of *TP* is a neighbor of node *d*, mark all neighbors of *i* as reachable, that is, add *N*(*i*) to*INL*.
2. If a node *i* of *OLDPL* is a neighbor of node *d*, and *i* < *d*, mark all neighbors of node *i* as reachable, that is, include *N*(*i*) in *INL*.
3. If neighbor *i* of node *d* has a neighbor *n* present in *TP*, mark all neighbors of *i* as reachable by adding *N*(*i*) to *INL*.
4. If neighbor *i* of node *d* has a neighbor *n* present in *OLDPL*, and *n* < *d*, here again add *N*(*i*) to *INL*, thereby marking all neighbors of node *i* as reachable.

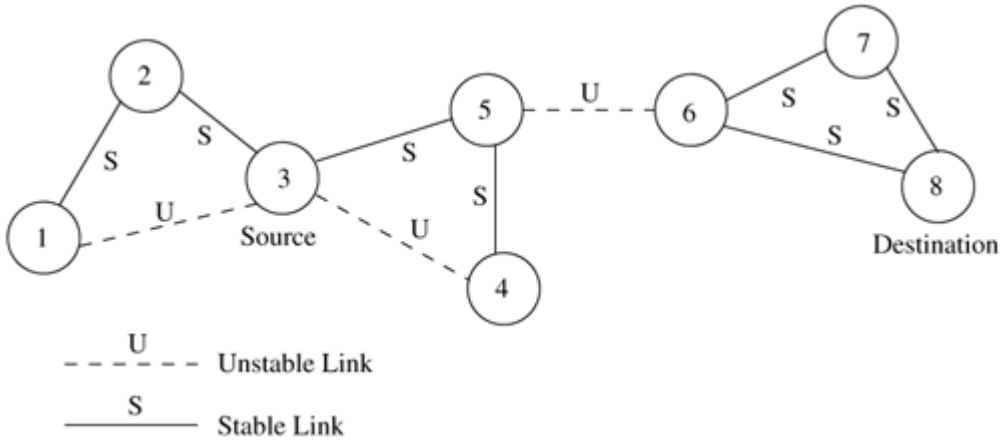
2. If neighbor *i* of node *d* is not in *INL*, put *i* in preferred list table *PLT* and mark all neighbors of *i* as reachable. If *i* is present in *INL*, mark the neighbors of *i* as unreachable by adding them to *EXL*, as*N*(*i*) may not be included in this step. Here neighbors *i* of *d* are processed in decreasing order of their degrees. After execution of this step, the *RouteRequest* is guaranteed to reach all neighbors of*d*. If *EXL* is not empty, some neighbor's neighbors *n* of node *d* are currently unreachable, and they are included in the next step.
3. If neighbor *i* of *d* has a neighbor *n* present in *EXL*, put *i* in *PLT* and mark all neighbors of *i* as reachable. Delete all neighbors of *i* from *EXL*. Neighbors are processed in decreasing order of their degrees. After execution of this step, all the nodes in *NNTd* are reachable. Apply reduction steps to remove overlapping neighbors from *PLT* without compromising on reachability.
4. Reduction steps are applied here in order to remove overlapping neighbors from *PLT* without compromising on reachability.

1. Remove each neighbor *i* from *PLT* if *N*(*i*) is covered by remaining neighbors of *PLT*. Here the minimum degree neighbor is selected every time.
2. Remove neighbor *i* from *PLT* whose *N*(*i*) is covered by node *d* itself.

Weight-Based Preferred Link Algorithm (WBPL)

In this algorithm, a node finds the preferred links based on stability, which is indicated by a weight, which in turn is based on its neighbors' temporal and spatial stability.

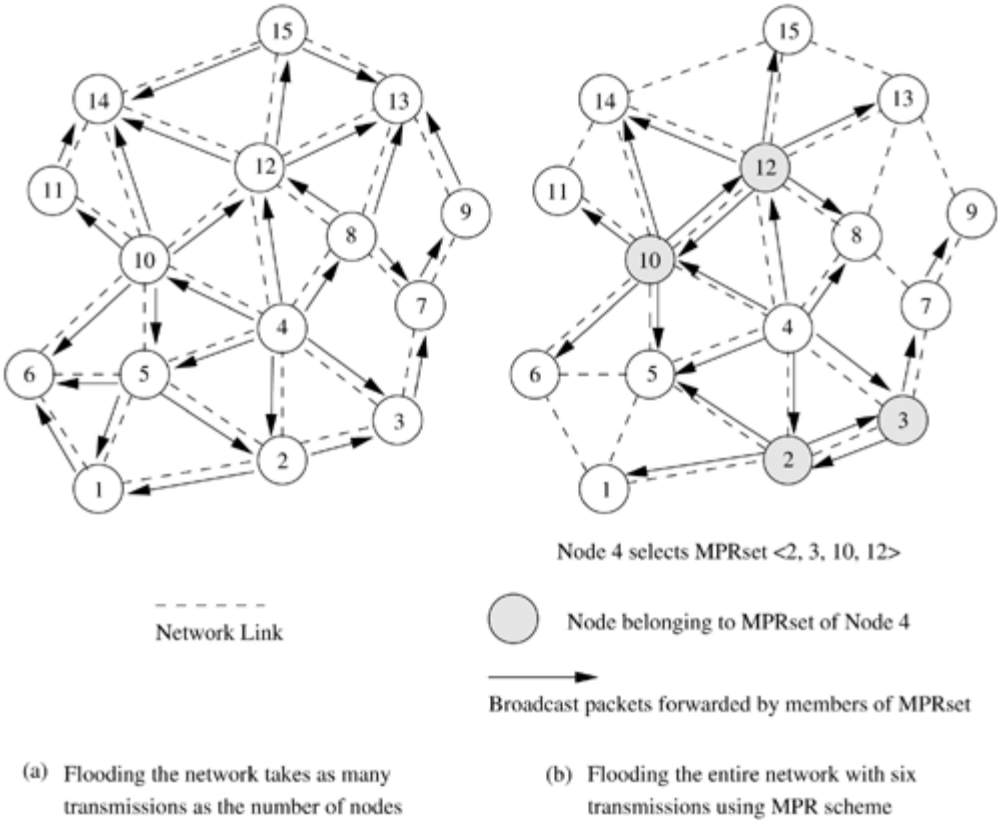
1. Let *BCnti* be the count of *beacon*s received from a neighbor *i* and *THbcon* is the number of beacons generated during a time period equal to that required to cover twice the transmission range ($TH_{bcon} = \frac{2 \times transmission\ range}{maximum\ velocity \times period\ of\ beacon}$)



Optimized Link State Routing

The optimized link state routing (OLSR) protocol [21] is a proactive routing protocol that employs an efficient link state packet forwarding mechanism called *multipoint relaying.* This protocol optimizes the pure link state routing protocol. Optimizations are done in two ways: by reducing the size of the control packets and by reducing the number of links that are used for forwarding the link state packets. The reduction in the size of link state packets is made by declaring only a subset of the links in the link state updates. This subset of links or neighbors that are designated for link state updates and are assigned the responsibility of packet forwarding are called *multipoint relays.* The optimization by the use of *multipoint relaying* facilitates periodic link state updates. The link state update mechanism does not generate any other control packet when a link breaks or when a link is newly added. The link state update optimization achieves higher efficiency when operating in highly dense networks. Figure 7.30 (a) shows the number of message transmissions required when the typical flooding-based approach is employed. In this case, the number of message transmissions is approximately equal to the number of nodes that constitute the network. The set consisting of nodes that are *multipoint relays* is referred to as *MPRset.* Each node (say, *P*) in the network selects an *MPRset* that processes and forwards every link state packet that node *P* originates. The neighbor nodes that do not belong to the *MPRset* process the link state packets originated by node *P* but do not forward them. Similarly, each node maintains a subset of neighbors called *MPR selectors,* which is nothing but the set of neighbors that have selected the node as a *multipoint relay.* A node forwards packets that are received from nodes belonging to its*MPRSelector* set. The members of both *MPRset* and *MPRSelectors* keep changing over time. The members of the *MPRset* of a node are selected in such a manner that every node in the node's two-hop neighborhood has a bidirectional link with the node. The

selection of nodes that constitute the *MPRset*significantly affects the performance of OLSR because a node calculates routes to all destinations only through the members of its *MPRset.* Every node periodically broadcasts its *MPRSelector* set to nodes in its immediate neighborhood. In order to decide on the membership of the nodes in the *MPRset,* a node periodically sends *Hello* messages that contain the list of neighbors with which the node has bidirectional links and the list of neighbors whose transmissions were received in the recent past but with whom bidirectional links have not yet been confirmed. The nodes that receive this *Hello* packet update their own two-hop topology tables. The selection of *multipoint relays* is also indicated in the*Hello* packet. A data structure called *neighbor table* is used to store the list of neighbors, the two-hop neighbors, and the status of neighbor nodes. The neighbor nodes can be in one of the three possible link status states, that is, unidirectional, bidirectional, and *multipoint relay.* In order to remove the stale entries from the *neighbor table,* every entry has an associated timeout value, which, when expired, removes the table entry. Similarly a sequence number is attached with the *MPRset* which gets incremented with every new *MPRset.*



(a) Flooding the network takes as many transmissions as the number of nodes

(b) Flooding the entire network with six transmissions using MPR scheme

HIERARCHICAL ROUTING PROTOCOLS

The use of routing hierarchy has several advantages, the most important ones being reduction in the size of routing tables and better scalability. This section discusses the existing hierarchical routing protocols for ad hoc wireless

networks.

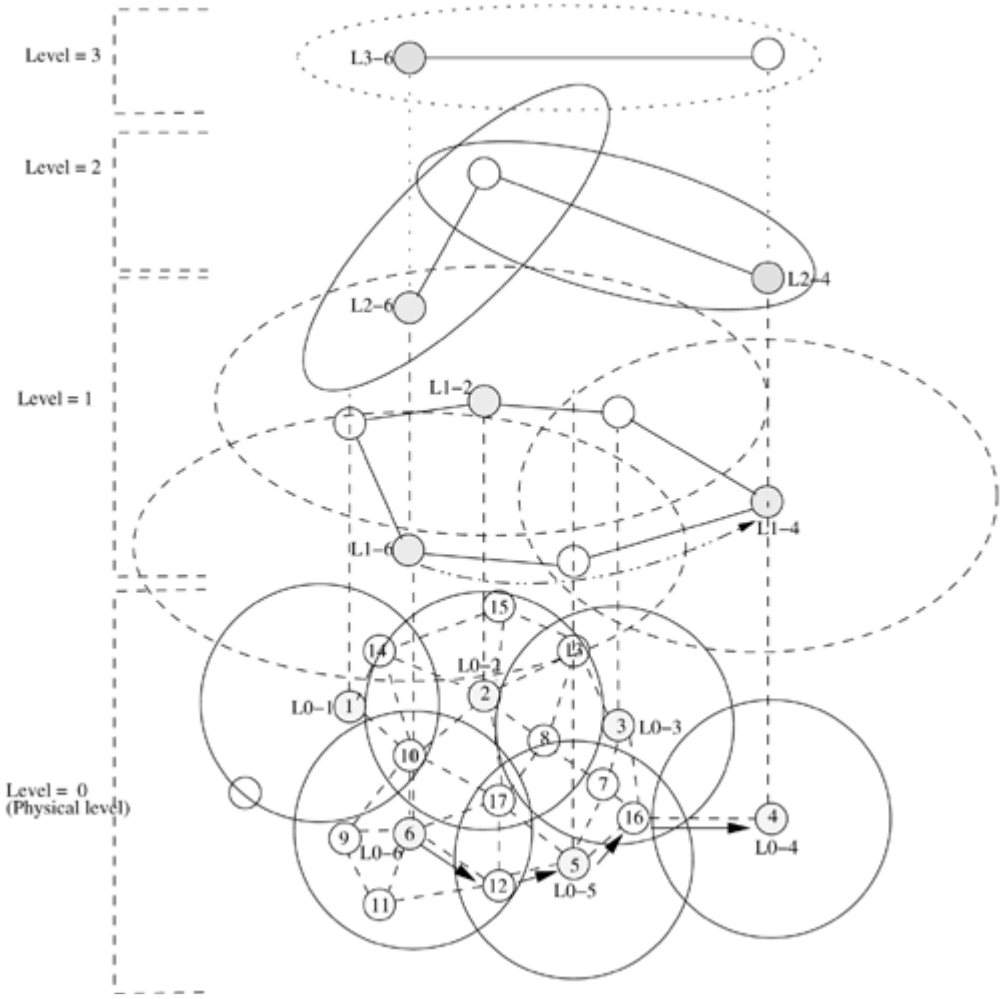
Hierarchical State Routing Protocol

The hierarchical state routing (HSR) protocol [23] is a distributed multi-level hierarchical routing protocol that employs clustering at different levels with efficient membership management at every level of clustering. The use of clustering enhances resource allocation and management. For example, the allocation of different frequencies or spreading codes to different clusters can improve the overall spectrum reuse. HSR operates by classifying different levels of clusters. Elected leaders at every level form the members at the immediate higher level. Different clustering algorithms, such as the one proposed in [8], are employed for electing leaders at every level. The first level of physical clustering is done among the nodes that are reachable in a single wireless hop. The next higher level of physical clustering is done among the nodes that are elected as leaders of each of these first-level clusters. In addition to the *physical clustering*, a *logical clustering* scheme has been proposed in HSR, which is based on certain relations among the nodes rather than on their geographical positions, as in the case of *physical clustering*.

Figure 7.31 illustrates the multilayer clustering defined by the HSR protocol. At the lowest level ($L = 0$), there are six cluster leaders (nodes 1, 2, 3, 4, 5, and 6). Nodes are classified as cluster leaders, or gateway nodes, or normal member nodes. A cluster leader is entrusted with responsibilities such as slot/frequency/code allocation, call admission control, scheduling of packet transmissions, exchange of routing information, and handling route breaks.

In Figure 7.31, node 5 is a clusterhead marked as $L0$-5, which refers to the level of clustering ($L = 0$) and node ID (5). Similarly, each of the higher-level cluster leaders is also marked (*e.g.*, $L1 - 6$, $L - 2 - 6$, and $L3 - 6$ refer to the same node 6, but acting as leader with the given leader IDs at levels 1, 2, and 3, respectively). The spectrum reuse schemes, including spreading code assignment, can be used among the cluster leaders of the $L = 0$ clusters. For the nodes under the leadership of node 6 at level 0, the cluster members are nodes 9, 10, 11, 12, and 17. Those nodes that belong to multiple clusters are referred to as cluster gateway nodes. For the level 0 cluster whose leader is node 6, the cluster gateways are nodes 10, 12, and 17. The second level of clustering is done among the leaders of the first level, that is, the leaders of 0*th* level clusters, $L0$ - 1, $L0$ - 2, $L0$ - 3,$L0$ - 4, $L0$ - 5, and $L0$ - 6, form the members of the first-level cluster.

Example of HSR multi-level clustering.



Every node maintains information about all its neighbors and the status of links to each of them. This information is broadcast within the cluster at regular intervals. The cluster leader exchanges the topology and link state routing information among its peers in the neighborhood clusters, using which the nexthigher-level clustering is performed. This exchange of link state information is done over multiple hops that consist of gateway nodes and cluster-heads. The path between two cluster-heads which is formed by multiple wireless links is called a *virtual link*. The link status for the *virtual link*(otherwise called *tunnel*) is obtained from the link status parameters of the wireless links that constitute the *virtual link*. In Figure 7.31, the path between first-level clusters *L1* - 6 and *L1* - 4 includes the wireless links 6 - 12 - 5 - 16 - 4. The clustering is done recursively to the higher levels. At any level, the cluster leader exchanges topology information with its peers. After obtaining information from its peers, it floods the information to the lower levels, making every node obtain the hierarchical topology information. This hierarchical topology necessitates a hierarchical addressing which helps in operating with less routing information

against the full topology exchange required in the link state routing. The hierarchical addressing defined in HSR includes the hierarchical ID (HID) and node ID. The HID is a sequence of IDs of cluster leaders of all levels starting from the highest level to the current node. This ID of a node in HSR is similar to the unique MAC layer address. The hierarchical addresses are stored in an HSR table at every node that indicates the node's own position in the hierarchy. The HSR table is updated whenever routing update packets are received by the node.

The hierarchical address of node 11 in Figure 7.31 is < 6, 6, 6, 6, 11 >, where the last entry (11) is the node ID and the rest consists of the node IDs of the cluster leaders that represent the location of node 11 in the hierarchy. Similarly, the HID of node 4 is < 6, 4, 4, 4 >. When node 11 needs to send a packet to node 4, the packet is forwarded to the highest node in the hierarchy (node 6). Node 6 delivers the packet to node 4, which is at the top-most level of the hierarchy.

Advantages and Disadvantages

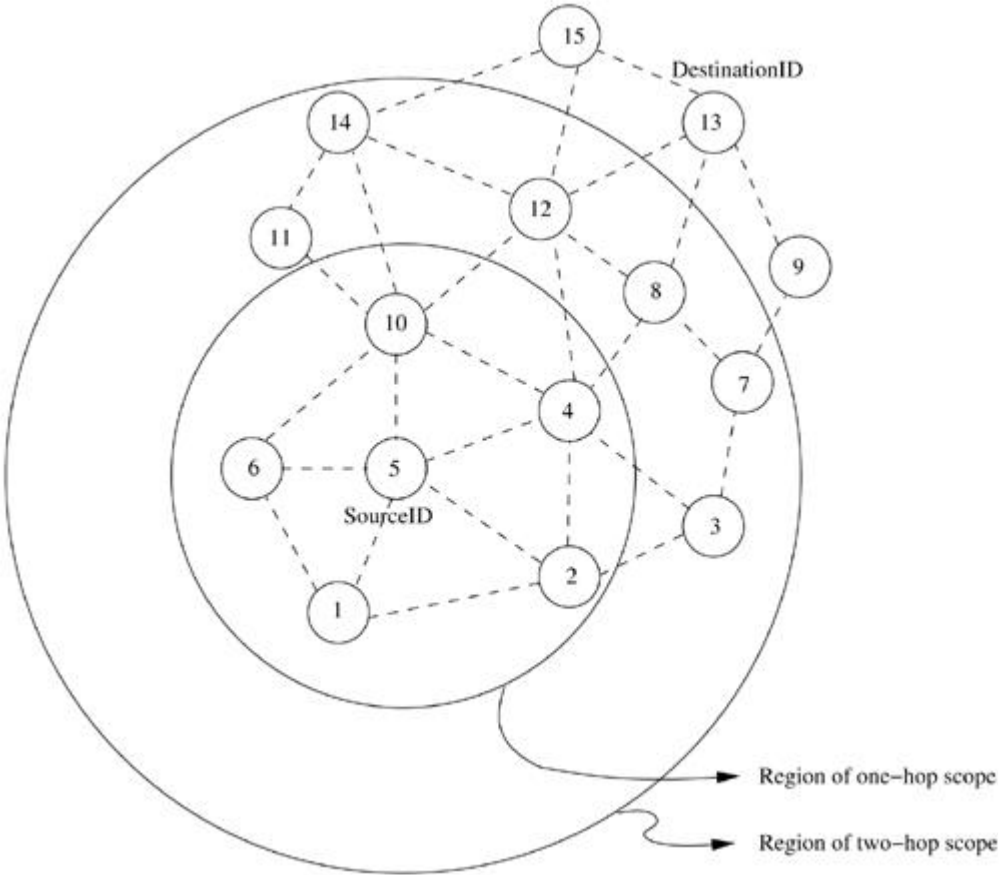
The HSR protocol reduces the routing table size by making use of hierarchy information. In HSR, the storage required is $O(n \times m)$ compared to $O(nm)$ that is required for a flat topology link state routing protocol (n is the average number of nodes in a cluster and m is the number of levels). Though the reduction in the amount of routing information stored at nodes is appreciable, the overhead involved in exchanging packets containing information about the multiple levels of hierarchy and the leader election process make the protocol unaffordable in the ad hoc wireless networks context. Besides, the number of nodes that participate in an ad hoc wireless network does not grow to the dimensions of the number of nodes in the Internet where the hierarchical routing is better suited. In the military applications of ad hoc wireless networks, the hierarchy of routing assumes significance where devices with higher capabilities of communication can act as the cluster leaders.

7.8.2 Fisheye State Routing Protocol

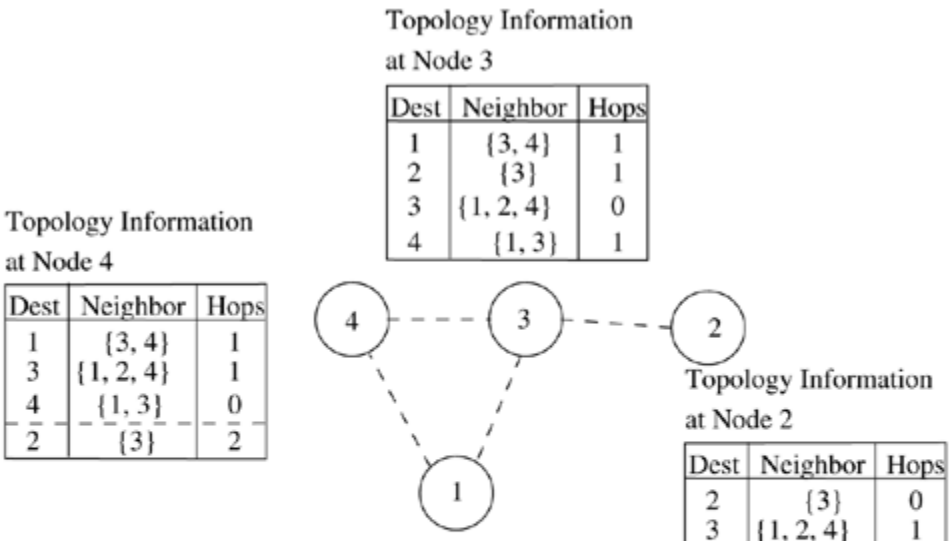
The table-driven routing protocols generate routing overhead that is dependent on the size of the network and mobility of the nodes, whereas the routing overhead generated by on-demand routing protocols are dependent on the number of connections present in the system in addition to the above two factors. Hence, as the number of senders in the network increases, the routing overhead also increases. ZRP uses an intra-zone proactive approach and an inter-zone reactive approach to reduce control overhead. The fisheye state routing (FSR) protocol [23] is a generalization of the GSR [24] protocol. FSR uses the *fisheye* technique to reduce information required to represent graphical data, to reduce routing overhead. The basic principle behind this technique is the property of a fish's eye that can capture pixel information with greater accuracy near its eye's focal point. This accuracy decreases with an increase in the distance from the center of the focal point. This property is translated to routing in ad hoc wireless networks by a node, keeping accurate information about nodes in its local topology, and not-so-accurate information about far-away nodes, the accuracy of the network information decreasing with

increasing distance. FSR maintains the topology of the network at every node, but does not flood the entire network with the information, as is done in link state routing protocols. Instead of flooding, a node exchanges topology information only with its neighbors. A sequence numbering scheme is used to identify the recent topology changes. This constitutes a hybrid approach comprising of the link-level information exchange of distance vector protocols and the complete topology information exchange of link state protocols. The complete topology information of the network is maintained at every node and the desired shortest paths are computed as required. The topology information exchange takes place periodically rather than being driven by an event. This is because instability of the wireless links may cause excessive control overhead when event-driven updates are employed. FSR defines routing scope, which is the set of nodes that are reachable in a specific number of hops. The scope of a node at two hops is the set of nodes that can be reached in two hops. Figure 7.32shows the scope of node 5 with one hop and two hops. The routing overhead is significantly reduced by adopting different frequencies of updates for nodes belonging to different scopes.

Fisheye state routing.



The link state information for the nodes belonging to the smallest scope is exchanged at the highest frequency. The frequency of exchanges decreases with an increase in scope. This keeps the immediate neighborhood topology information maintained at a node more precise compared to the information about nodes farther away from it. Thus the message size for a typical topology information update packet is significantly reduced due to the removal of topology information regarding the far-away nodes. The path information for a distant node may be inaccurate as there can be staleness in the information. But this is compensated by the fact that the route gets more and more accurate as the packet nears its destination. FSR scales well for large ad hoc wireless networks because of the reduction in routing overhead due to the use of the abovedescribed mechanism, where varying frequencies of updates are used.
Figure 7.33 illustrates an example depicting the network topology information maintained at nodes in a network. The routing information for the nodes that are one hop away from a node are exchanged more frequently than the routing information about nodes that are more than one hop away. Information regarding nodes that are more than one hop away from the current node are listed below the dotted line in the topology table.



Advantages and Disadvantages
The notion of multi-level scopes employed by FSR significantly reduces the

bandwidth consumed by link state update packets. Hence, FSR is suitable for large and highly mobile ad hoc wireless networks. The choice of the number of hops associated with each scope level has a significant influence on the performance of the protocol at different mobility values, and hence must be carefully chosen.

7.9 POWER-AWARE ROUTING PROTOCOLS

In a deviation from the traditional wired network routing and cellular wireless network routing, power consumption by the nodes is a serious factor to be taken into consideration by routing protocols for ad hoc wireless networks. This is because, in ad hoc wireless networks, the routers are also equally powerconstrained just as the nodes are. This section discusses some of the important routing metrics that take into consideration this energy factor.

7.9.1 Power-Aware Routing Metrics

The limitation on the availability of power for operation is a significant bottleneck, given the requirements of portability, weight, and size of commercial hand-held devices. Hence, the use of routing metrics that consider the capabilities of the power sources of the network nodes contributes to the efficient utilization of energy and increases the lifetime of the network. Singh *et al.* proposed a set of routing metrics in [25] that supports conservation of battery power. The routing protocols that select paths so as to conserve power must be aware of the states of the batteries at the given node as well as at the other intermediate nodes in the path.

Minimal Energy Consumption per Packet

This metric aims at minimizing the power consumed by a packet in traversing from source node to the destination node. The energy consumed by a packet when traversing through a path is the sum of the energies required at every intermediate hop in that path. The energy consumed at an intermediate hop is a function of the distance between the nodes that form the link and the load on that link. This metric does not balance the load so that uniform consumption of power is maintained throughout the network. The disadvantages of this metric include selection of paths with large hop length, inability to measure the power consumption at a link in advance when the load varies, and the inability to prevent the fast discharging of batteries at some nodes.

Maximize Network Connectivity

This metric attempts to balance the routing load among the *cut-set* (the subset of the nodes in the network, the removal of which results in network partitions). This assumes significance in environments where network connectivity is to be ensured by uniformly distributing the routing load among the *cut-set.* With a variable traffic origination rate and unbounded contention in the network, it is difficult to achieve a uniform battery draining rate for the *cut-set.*

Minimum Variance in Node Power Levels

This metric proposes to distribute the load among all nodes in the network so that the power consumption pattern remains uniform across them. This problem is very complex when the rate and size of data packets vary. A nearly optimal performance can be achieved by routing packets to the least-loaded next-hop

node.

Minimum Cost per Packet

In order to maximize the life of every node in the network, this routing metric is made as a function of the state of the node's battery. A node's cost decreases with an increase in its battery charge and vice versa. Translation of the remaining battery charge to a cost factor is used for routing. With the availability of a battery discharge pattern, the cost of a node can be computed. This metric has the advantage of ease in the calculation of the cost of a node and at the same time congestion handling is done.

Minimize Maximum Node Cost

This metric minimizes the maximum cost per node for a packet after routing a number of packets or after a specific period. This delays the failure of a node, occurring due to higher discharge because of packet forwarding.

Unit-5

TRANSPORT LAYER AND SECURITY PROTOCOLS FOR AD HOC WIRELESS NETWORKS

INTRODUCTION

The objectives of a transport layer protocol include the setting up of an end-toend connection, end-to-end delivery of data packets, flow control, and congestion control. There exist simple, unreliable, and connection-less transport layer protocols such as UDP, and reliable, byte-stream-based, and connectionoriented transport layer protocols such as TCP for wired networks. These traditional wired transport layer protocols are not suitable for ad hoc wireless networks due to the inherent problems associated with the latter. The first half of this chapter discusses the issues and challenges in designing a transport layer protocol for ad hoc wireless networks, the reasons for performance degradation when TCP is employed in ad hoc wireless networks, and it also discusses some of the existing TCP extensions and other transport layer protocols for ad hoc wireless networks.

ISSUES IN DESIGNING A TRANSPORT LAYER PROTOCOL FOR AD HOC WIRELESS NETWORKS

Issues to be considered while designing a transport layer protocol for ad hoc wireless networks are discussed

• **Induced traffic:**

Unlike wired networks, ad hoc wireless networks utilize ulti-hop radio relaying. A link-level transmission affects the neighbor nodes of both the sender and receiver of the link. In a path having multiple links, transmission at a particular link affects one upstream link and one downstream link. This traffic at any given link (or path) due to the traffic through neighboring links (or paths) is referred to as induced traffic. This is due to the broadcast nature of the channel and the location-dependent contention on the channel. This induced traffic affects the throughput achieved by the transport layer protocol.

• **Induced throughput unfairness:** This refers to the throughput unfairness at the transport layer due to the throughput/delay unfairness existing at the lower layers such as the network and MAClayers. For example, an ad hoc wireless network that uses IEEE 802.11 DCF as the MAC protocol may experience throughput unfairness at the transport layer as well. A transport layer protocol should consider these in order to provide a fair share of throughput across contending flows.

• **Separation of congestion control, reliability, and flow control:** A transport layer protocol can provide better performance if end-to-end reliability, flow control, and congestion control are handled separately. Reliability and flow control are end-to-end activities, whereas congestion can at times be a local activity. The transport layer flow can experience congestion with just one intermediate link under congestion. Hence, in networks such as ad hoc wireless networks, the performance of the transport layer may be improved if these are separately handled. While separating these, the most important objective to be considered is the minimization of the additional control overhead generated by them.

• **Power and bandwidth constraints:** Nodes in ad hoc wireless networks face resource constraints including the two most important resources: (i) power source and (ii) bandwidth. The performance of a transport layer protocol is significantly affected by these constraints.

• **Misinterpretation of congestion:** Traditional mechanisms of detecting congestion in networks, such as packet loss and retransmission timeout, are not suitable for detecting the network congestion in ad hoc wireless networks. This is because the high error rates of wireless channel, location-dependent contention, hidden terminal problem, packet collisions in the network, path breaks due to the mobility of nodes, and node failure due to a drained battery can also lead to packet loss in ad hoc wireless networks. Hence, interpretation of network congestion as used in traditional networks is not appropriate in ad hoc wireless networks.

• **Completely decoupled transport layer:** Another challenge faced by a transport layer protocol is the interaction with the lower layers. Wired network transport layer protocols are almost completely decoupled from the lower layers. In ad hoc wireless networks, the cross-layer interaction between the transport layer and lower layers such as the network layer and the MAC layer is important for the transport layer to adapt to the changing network environment.

• **Dynamic topology:** Some of the deployment scenarios of ad hoc wireless networks experience rapidly changing network topology due to the mobility of nodes. This can lead to frequent path

breaks, partitioning and remerging of networks, and high delay in reestablishment of paths. Hence, the performance of a transport layer protocol is significantly affected by the rapid changes in the network topology.

DESIGN GOALS OF A TRANSPORT LAYER PROTOCOL FOR AD HOC WIRELESS NETWORKS

The following are the important goals to be met while designing a transport layer protocol for ad hoc wireless networks:

• The protocol should maximize the throughput per connection.

• It should provide throughput fairness across contending flows.

• The protocol should incur minimum connection setup and connection maintenance overheads. It should minimize the resource requirements for setting up and maintaining the connection in order to make the protocol scalable in large networks.

• The transport layer protocol should have mechanisms for congestion control and flow control in the network.

• It should be able to provide both reliable and unreliable connections as per the requirements of the application layer.

• The protocol should be able to adapt to the dynamics of the network such as the rapid change in topology and changes in the nature of wireless links from uni-directional to bidirectional or vice versa.

• One of the important resources, the available bandwidth, must be used efficiently.

• The protocol should be aware of resource constraints such as battery power and buffer sizes and make efficient use of them.

• The transport layer protocol should make use of information from the lower layers in the protocol stack for improving the network throughput.

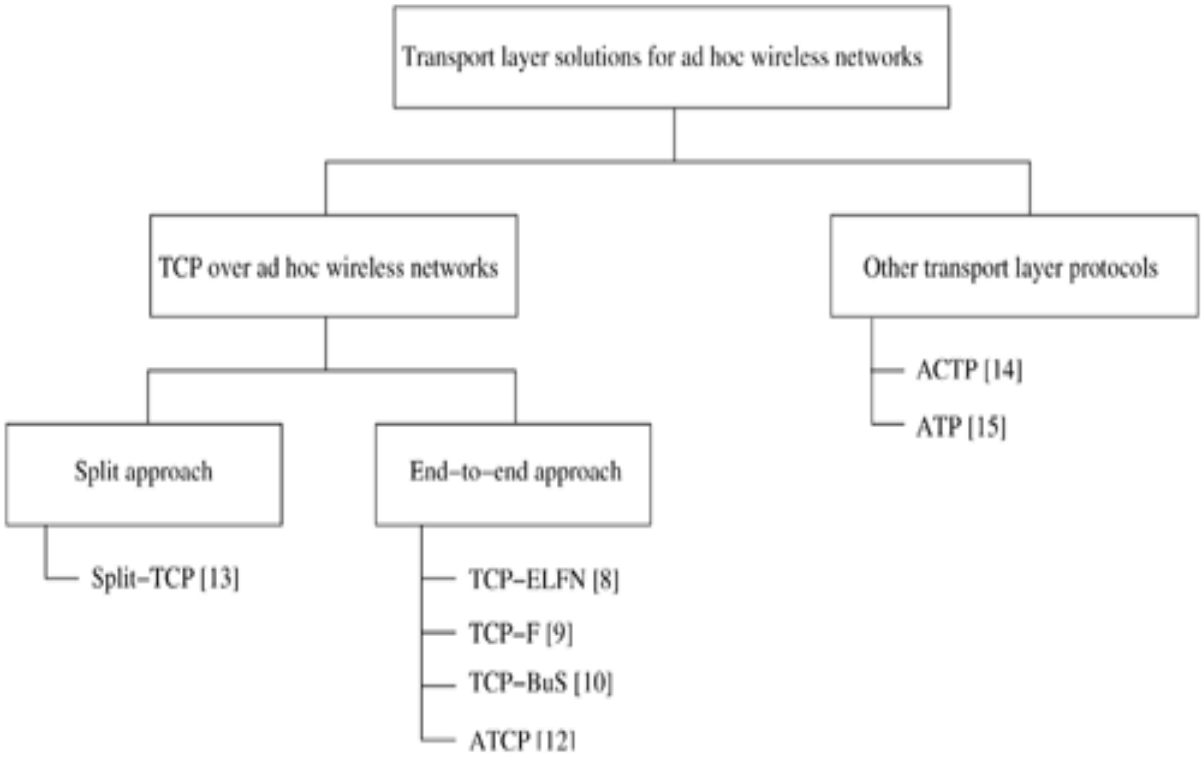
• It should have a well-defined cross-layer interaction framework for effective, calable, and protocol-independent interaction with lower layers.

• The protocol should maintain end-to-end semantics.

CLASSIFICATION OF TRANSPORT LAYER SOLUTIONS

Figure shows a classification tree for some of the transport layer protocols discussed in this chapter. The top-level classification divides the protocols as extensions of TCP for ad hoc wireless networks and other transport layer protocols which are not based on TCP. The solutions for TCP over ad hoc wireless networks can further be classified into split approaches and end-to-end approaches.

Figure: Classification of transport layer solutions.



TCP OVER AD HOC WIRELESS NETWORKS

The transmission control protocol (TCP) is the most predominant transport layer protocol in the Internet today. It transports more than 90% percent of the traffic on the Internet. Its reliability, end-to-end congestion control mechanism, bytestream transport mechanism, and, above all, its elegant and simple design have not only contributed to the success of the Internet, but also have made TCP an influencing protocol in the design of many of the other protocols and applications. Its adaptability to the congestion in the network has been an important feature leading to graceful degradation of the services offered by the network at times of extreme congestion. TCP in its traditional form was designed and optimized only for wired networks. Extensions of TCP that

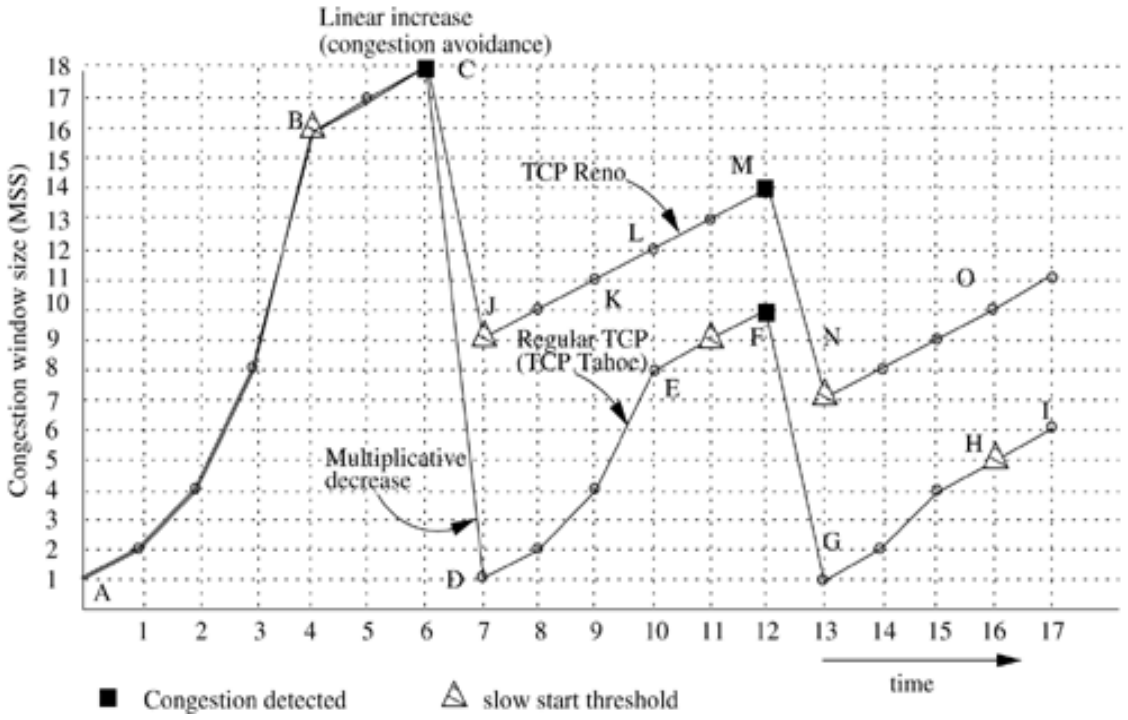
provide improved performance across wired and single-hop wireless networks were discussed in Chapter 4. Since TCP is widely used today and the efficient integration of an ad hoc wireless network with the Internet is paramount wherever possible, it is essential to have mechanisms that can improve TCP's performance in ad hoc wireless networks. This would enable the seamless operation of application-level protocols such as FTP, SMTP, and HTTP across the integrated ad hoc wireless networks and the Internet. This section discusses the issues and challenges that TCP experiences when used in ad hoc wireless networks as well as some of the existing solutions for overcoming them.

A Brief Revisit to TraditionalTCP

TCP [1] is a reliable, end-to-end, connection-oriented transport layer protocol that provides a byte-stream-based service [the stream of bytes from the application layer is split into TCP segments,1 the length of each segment limited by a maximum segment size (MSS)]. The major responsibilities of TCP includecongestion control, flow control, in-order delivery of packets, and reliable transportation of packets. Congestion control deals with excess traffic in the network which may lead to degradation in the performance of the network, whereas flow control controls the per-flow traffic such that the receiver capacity is not exceeded. TCP regulates the number of packets sent to the network by expanding and shrinking the congestion window. The TCP sender starts the session with a congestion window value of one MSS. It sends out one MSS and waits for he ACK. Once the ACK is received within the retransmission timeout (RTO) period, the congestion window is doubled and two MSSs are originated. This doubling of the congestion window with every successful acknowledgment of all the segments in the current congestion window, is called *slow-start* (a more appropriate name would be*exponential start*, as it actually grows exponentially) and it continues until the congestion window reaches the *slowstart threshold* (the slow-start threshold has an initial value of 64 KB). Figure 9.2shows the variation of the congestion window in TCP; the slow start phase is between points A-B. Once it reaches the slow-start threshold (in Figure 9.2, the slow-start threshold is initially taken as 16 for illustration), it grows linearly, adding one MSS to the congestion window on every ACK received. This linear growth, which continues until the congestion window reaches the receiver window (which is advertised by the TCP receiver and carries the information about the receiver's buffer size), is called*congestion avoidance*, as it tries to avoid increasing the congestion window

exponentially, which will surely worsen the congestion in the network. TCP updates the RTO period with the current round-trip delay calculated on the arrival of every ACK packet. If the ACK packet does not arrive within the RTOperiod, then it assumes that the packet is lost. TCP assumes that the packet loss is due to the congestion in the network and it invokes the congestion control mechanism. The TCP sender does the following during congestion control: (i) reduces the slow-start threshold to half the current congestion window or two MSSs whichever is larger, (ii) resets the congestion window size to one MSS, (iii) activates the slow-start algorithm, and (iv) resets the RTO with an exponential back-off value which doubles with every subsequent retransmission. The slow-start process further doubles the congestion window with every successfully acknowledged window and, upon reaching the slow-start threshold, it enters into the congestion avoidance phase. 1 TCP does not maintain packet boundaries, and hence multiple application layer packets belonging to the same TCP connection, containing stream of bytes, may be combined into a single packet, or a single packet may be split into multiple packets, but delivered as a stream of bytes. Hence, a TCPpacket is considered as a segment containing several bytes rather than a packet. However, segment and packet are used interchangeably in this chapter.

Illustration of TCP congestion window



The TCP sender also assumes a packet loss if it receives three consecutive duplicate ACKs (DUPACKs) [repeated acknowledgments for the same TCP segment that was successfully received in-order at the receiver]. Upon reception of three DUPACKs, the TCP sender retransmits the oldest unacknowledged segment. This is called the *fast retransmit* scheme. When the TCP receiver receives out-of-order packets, it generates DUPACKs to indicate to the TCP sender about the sequence number of the last in-order segment received successfully.

Why Does TCP Not Perform Well in Ad Hoc Wireless Networks?

The major reasons behind throughput degradation that TCP faces when used in ad hoc wireless networks are the following:

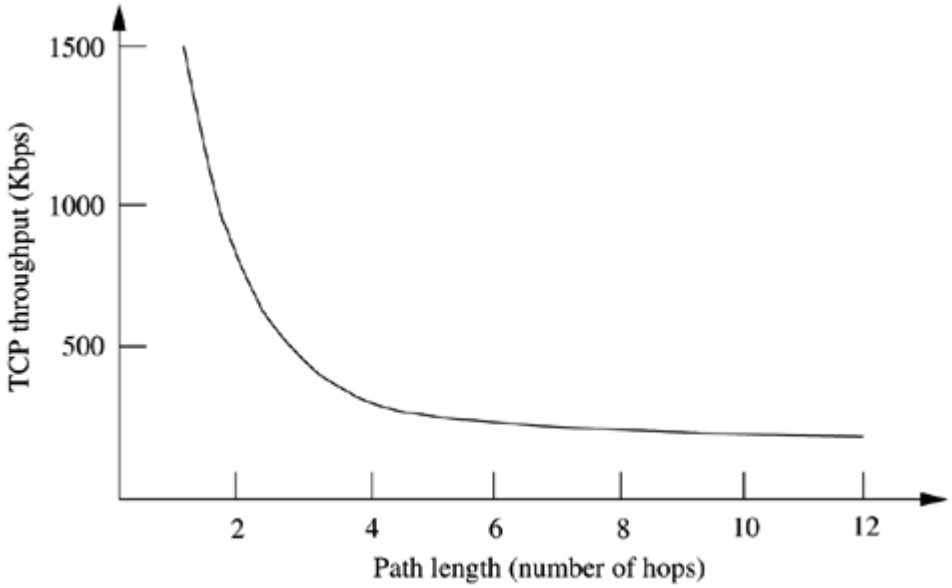
• **Misinterpretation of packet loss:** Traditional TCP was designed for wired networks where the packet loss is mainly attributed to network congestion. Network congestion is detected by the sender's packet RTO period. Once a packet loss is detected, the sender node assumes congestion in the network and invokes a congestion control algorithm. Ad hoc wireless networks experience a much higher packet loss due to factors such as high bit error rate (BER) in the wireless channel, increased collisions due to the presence of hidden terminals, presence of interference,

location-dependent contention, uni-directional links, frequent path breaks due to mobility of nodes, and the inherent fading properties of the wireless channel.

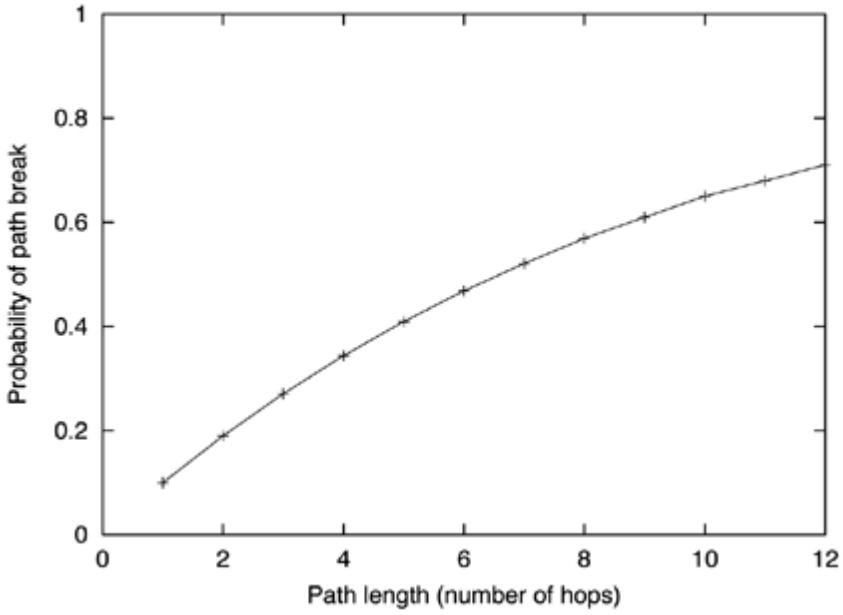
• **Frequent path breaks:** Ad hoc wireless networks experience dynamic changes in network topology because of the unrestricted mobility of the nodes in the network. The topology changes lead to frequent changes in the connectivity of wireless links and hence the route to a particular destination may need to be recomputed very often. The responsibility of finding a route and reestablishing it once it gets broken is attached to the network layer Once a path is broken, the routing protocol initiates a route reestablishment process. This route reestablishment process takes a significant amount of time to obtain a new route to the destination. The route reestablishment time is a function of the number of nodes in the network, transmission ranges of nodes, current topology of the network, bandwidth of the channel, traffic load in the network, and the nature of the routing protocol. If the route reestablishment time is greater than the RTO period of the TCP sender, then the TCP sender assumes congestion in the network, retransmits the lost packets, and initiates the congestion control algorithm. These retransmissions can lead to wastage of bandwidth and battery power. Eventually, when a new route is found, the TCP throughput continues to be low for some time, as it has to build up the congestion window since the traditional TCP undergoes a slow start.

• **Effect of path length:** It is found that the TCP throughput degrades rapidly with an increase in path length in string (linear chain) topology ad hoc wireless networks [7], [8]. This is shown inFigure 9.3. The possibility of a path break increases with path length. Given that the probability of a link break is pl , the probability of a path break (pb) for a path of length k can be obtained as $pb = 1 - (1 - pl)k$. Figure 9.4 shows the variation of pb with path length for $pl = 0.1$. Hence as the path length increases, the probability of a path break increases, resulting in the degradation of the throughput in the network.

Variation of TCP throughput with path length.



Variation of p_b with path length (p_l = 0.1).



• **Misinterpretation of congestion window:** TCP considers the congestion window as a measure of the rate of transmission that is acceptable to the network and the receiver. In ad hoc wireless networks, the congestion control mechanism is invoked when the network gets partitioned or when a path break occurs. This reduces the congestion window and increases the RTO period.

When the route is reconfigured, the congestion window may not reflect the transmission rate acceptable to the new route, as the new route may actually accept a much higher transmission rate. Hence, when there are frequent path breaks, the congestion window may not reflect the maximum transmission rate acceptable to the network and the receiver.

• **Asymmetric link behavior:** The radio channel used in ad hoc wireless networks has different properties such as location-dependent contention, environmental effects on propagation, and directional properties leading to asymmetric links. The directional links can result in delivery of a packet to a node, but failure in the delivery of the acknowledgment back to the sender. It is possible for a bidirectional link to become uni-directional for a while. This can also lead to TCP invoking the congestion control algorithm and several retransmissions.

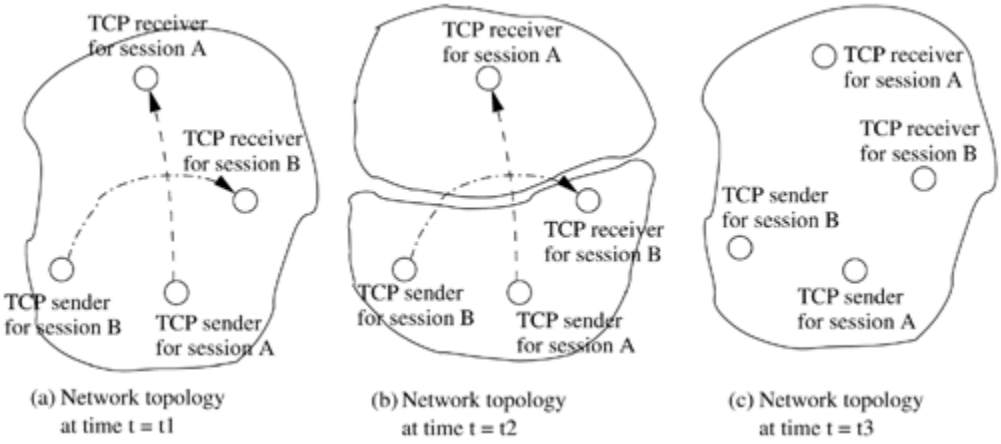
• **Uni-directional path:** Traditional TCP relies on end-to-end ACK for ensuring reliability. Since the ACK packet is very short compared to a data segment, ACKs consume much less bandwidth in wired networks. In ad hoc wireless networks, every TCP ACK packet requires RTS-CTS-Data-ACK exchange in case IEEE 802.11 is used as the underlying MAC protocol. This can lead to an additional overhead of more than 70 bytes if there are no retransmissions. This can lead to significant bandwidth consumption on the reverse path, which may or may not contend with the forward path. If the reverse path contends with the forward path, it can lead to the reduction in the throughput of the forward path. Some routing protocols select the forward path to be also used as the reverse path, whereas certain other routing protocols may use an entirely different or partially different path for the ACKs. A path break on an entirely different reverse path can affect the performance of the network as much as a path break in the forward path.

• **Multipath routing:** There exists a set of QoS routing and best-effort routing protocols that use multiple paths between a source-destination pair. There are several advantages in using multipath routing. Some of these advantages include the reduction in route computing time, the high resilience to path breaks, high call acceptance ratio, and better security. For TCP, these advantages may add to throughput degradation. These can lead to a significant amount of out-of-order packets, which in turn generates a set of duplicate acknowledgments (DUPACKs) which cause additional power consumption and invocation of congestion control.

• **Network partitioning and remerging:** The randomly moving nodes in an ad hoc wireless network can lead to network partitions. As long as the TCP sender, the TCP receiver, and all the

intermediate nodes in the path between the TCP sender and the TCP receiver remain in the same partition, the TCP connection will remain intact. It is likely that the sender and receiver of the TCPsession will remain in different partitions and, in certain cases, that only the intermediate nodes are affected by the network partitioning. Figure 9.5 illustrates the effect of network partitions in ad hoc wireless networks. A network with two TCP sessions A and B is shown in Figure 9.5 (a) at time instant t1. Due to dynamic topological changes, the network gets partitioned into two as in Figure 9.5 (b) at time t2. Now the TCP session A's sender and receiver belong to two different partitions and the TCPsession B experiences a path break.

Effect of partitioning and merging of network.



Feedback-Based TCP

Feedback-based TCP [also referred to as TCP feedback (TCP-F)] [9] proposes modifications to the traditional TCP for improving performance in ad hoc wireless networks. It uses a feedback-based approach. TCP-F requires the support of a reliable link layer and a routing protocol that can provide feedback to the TCP sender about the path breaks. The routing protocol is expected to repair the broken path within a reasonable time period. TCP-F aims to minimize the throughput degradation resulting from the frequent path breaks that occur in ad hoc wireless networks. During a TCP session, there could be several path breaks resulting in considerable packet loss and path reestablishment delay. Upon detection of packet loss, the sender in a TCP

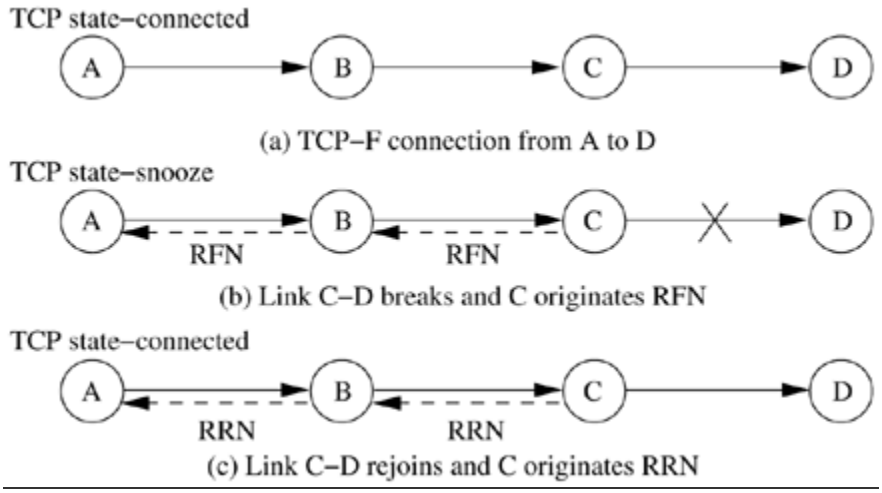
session invokes the congestion control algorithm leading to the exponential back-off of retransmission timers and a decrease in congestion window size.

In TCP-F, an intermediate node, upon detection of a path break, originates a route failure notification (RFN) packet. This RFN packet is routed toward the sender of the TCP session. The TCP sender's information is expected to be obtained from the TCP packets being forwarded by the node. The intermediate node that originates the RFN packet is called the failure point (FP).

The FP maintains information about all the RFNs it has originated so far. Every intermediate node that forwards the RFNpacket understands the route failure, updates its routing table accordingly, and avoids forwarding any more packets on that route. If any of the intermediate nodes that receive RFN has an alternate route to the same destination, then it discards the RFN packet and uses the alternate path for forwarding further data packets, thus reducing the control overhead involved in the route reconfiguration process. Otherwise, it forwards the RFN toward the source node. When a TCP sender receives an RFN packet, it goes into a state called *snooze*. In the snooze state, a sender stops sending any more packets to the destination, cancels all the timers, freezes its congestion window, freezes the retransmission timer, and sets up a route failure timer.

This route failure timer is dependent on the routing protocol, network size, and the network dynamics and is to be taken as the worst-case route reconfiguration time. When the route failure timer expires, the TCP sender changes from the snooze state to the *connected* state.Figure 9.6 shows the operation of the TCP-F protocol. In the figure, a TCP session is set up between node A and node D over the path A-B-C-D [refer to Figure 9.6 (a)]. When the intermediate link between node C and node D fails, node C originates an RFN packet and forwards it on the reverse path to the source node [see Figure 9.6 (b)]. The sender's TCP state is changed to the snooze state upon receipt of an RFN packet. If the link CD rejoins, or if any of the intermediate nodes obtains a path to destination node D,a route reestablishment notification (RRN) packet is sent to node A and the TCPstate is updated back to the connected state [Figure 9.6 (c)].

Operation of TCP-F.



As soon as a node receives an RRN packet, it transmits all the packets in its buffer, assuming that the network is back to its original state. This can also take care of all the packets that were not acknowledged or lost during transit due to the path break. In fact, such a step avoids going through the slow-start process that would otherwise have occurred immediately after a period of congestion. The route failure timer set after receiving the RFN packet ensures that the sender does not remain in the snooze state indefinitely. Once the route failure timer expires, the sender goes back to the connected state in which it reactivates the frozen timers and starts sending the buffered and unacknowledged packets. This can also take care of the loss of the RRN packet due to any possible subsequent congestion. TCP-F permits the TCP congestion control algorithm to be in effect when the sender is not in the snooze state, thus making it sensitive to congestion in the network.

Advantages and Disadvantages

TCP-F provides a simple feedback-based solution to minimize the problems arising out of frequent path breaks in ad hoc wireless networks. At the same time, it also permits the TCP congestion control mechanism to respond to congestion in the network. TCP-F depends on the intermediate nodes' ability to detect route failures and the routing protocols' capability to reestablish a broken path within a reasonably short duration. Also, the FP should be able to obtain the correct path (the path which the packet traversed) to the TCP-F sender for sending the RFN packet. This is simple with a routing protocol that uses source routing [*i.e.*, dynamic source routing (DSR)]. If a route to the sender is not available at the FP, then additional control packets may need to be generated for routing the RFNpacket. TCP-F has an additional state compared to

the traditional TCP state machine, and hence its implementation requires modifications to the existing TCP libraries. Another disadvantage of TCP-F is that the congestion window used after a new route is obtained may not reflect the achievable transmission rate acceptable to the network and the TCP-F receiver.

TCP with Explicit Link Failure Notification

Holland and Vaidya proposed the use of TCP with explicit link failure notification (TCP-ELFN) [8] for improving TCP performance in ad hoc wireless networks. This is similar to TCP-F, except for the handling of explicit link failure notification (ELFN) and the use of TCP probe packets for detecting the route reestablishment. The ELFN is originated by the node detecting a path break upon detection of a link failure to the TCP sender. This can be implemented in two ways: (i) by sending an ICMP2destination unreachable (DUR) message to the sender, or (ii) by piggy-backing this information on the*RouteError*3 message that is sent to the sender.

2 Internet control message protocol (IETF RFC 792) is used for defining control messages for aiding routing in the Internet.

3 Certain routing protocols for ad hoc wireless networks have explicit *RouteError* messages to inform the sender about path breaks so that the sender can recompute a fresh route to the destination. This is especially used in on-demand routing protocols such as DSR. Once the TCP sender receives the ELFN packet, it disables its retransmission timers and enters a*standby* state. In this state, it periodically originates probe packets to see if a new route is reestablished. Upon reception of an ACK by the TCP receiver for the probe packets, it leaves the standby state, restores the retransmission timers, and continues to function as normal.

Advantages and Disadvantages

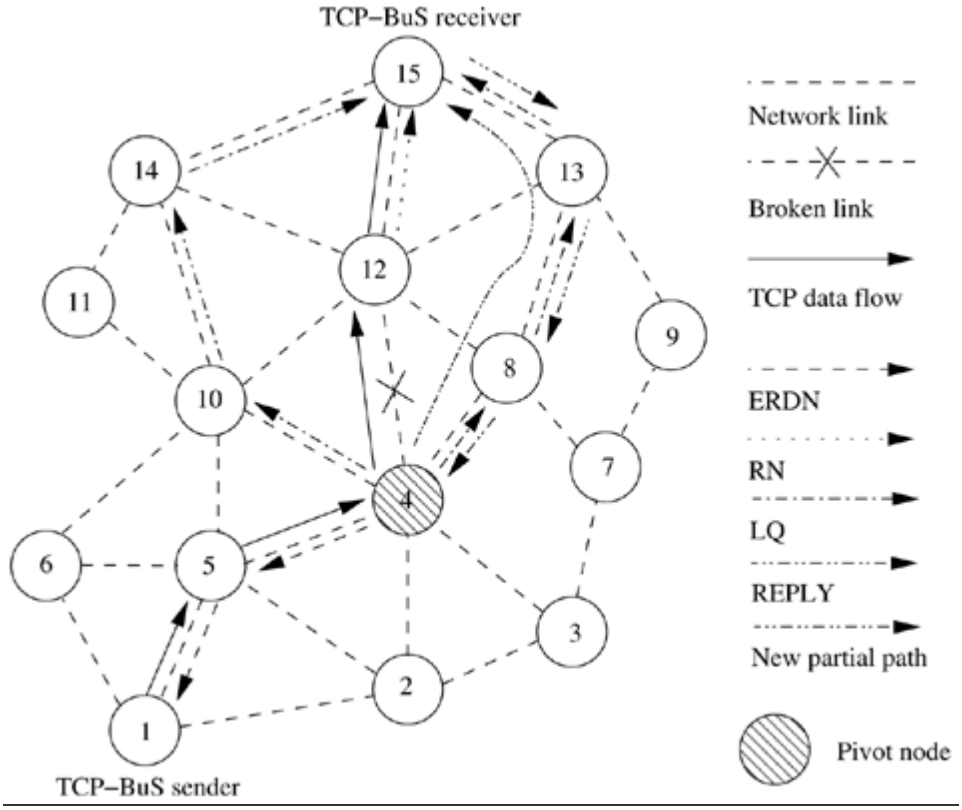
TCP-ELFN improves the TCP performance by decoupling the path break information from the congestion information by the use of ELFN. It is less dependent on the routing protocol and requires only link failure notification about the path break. The disadvantages of TCP-ELFN include the following: (i) when the network is temporarily partitioned, the path failure may last longer and this can lead to the origination of periodic probe packets consuming bandwidth and power and (ii) the congestion window used after a new route is obtained may not reflect the achievable transmission rate acceptable to the network and the TCP receiver.

TCP-BuS

TCP with buffering capability and sequence information (TCP-BuS) [10] is similar to the TCP-F andTCP-ELFN in its use of feedback information from an intermediate node on detection of a path break. But TCP-BuS is more dependent on the routing protocol compared to TCP-F and TCP-ELFN. TCP-BuS was proposed, with associativity-based routing (ABR) [11] protocol as the routing scheme. Hence, it makes use of some of the special messages such as localized query (LQ) and REPLY, defined as part ofABR for finding a partial path. These messages are modified to carry TCP connection and segment information. Upon detection of a path break, an upstream intermediate node [called pivot node (PN)] originates an explicit route disconnection notification (ERDN) message. This ERDN packet is propagated to the TCP-BuS sender and, upon reception of it, the TCP-BuS sender stops transmission and freezes all timers and windows as in TCP-F. The packets in transit at the intermediate nodes from the TCP-BuS sender to the PN are buffered until a new partial path from the PN to the TCPBuS receiver is obtained by the PN. In order to avoid unnecessary retransmissions, the timers for the buffered packets at the TCP-BuS sender and at the intermediate nodes up to PN use timeout values proportional to the roundtrip time (RTT). The intermediate nodes between the TCP-BuS sender and the PN can request the TCP-BuS sender to selectively retransmit any of the lost packets. Upon detection of a path break, the downstream node originates a route notification (RN) packet to the TCP-BuS receiver, which is forwarded by all the downstream nodes in the path. An intermediate node that receives an RN packet discards all packets belonging to that flow. The ERDN packet is propagated to the TCP-BuS sender in a reliable way by using an implicit acknowledgment and retransmission mechanism. The PN includes the sequence number of the TCP segment belonging to the flow that is currently at the head of its queue in the ERDN packet. The PN also attempts to find a new partial route to the TCP-BuS receiver, and the availability of such a partial path to destination is intimated to theTCP-BuS sender through an explicit route successful notification (ERSN) packet. TCP-BuS utilizes the route reconfiguration mechanism of ABR to obtain the partial route to the destination. Due to this, other routing protocols may require changes to support TCP-BuS. The LQ and REPLY messages are modified to carry TCP segment information, including the last successfully received segment at the destination. The LQ packet carries the sequence number of

the segment at the head of the queue buffered at the PN and the REPLY carries the sequence number of the last successful segment the TCPBuS receiver received. This enables the TCP-BuS receiver to understand the packets lost in transition and those buffered at the intermediate nodes. This is used to avoid fast retransmission requests usually generated by the TCP-BuS receiver when it notices an out-of-order packet delivery. Upon a successfulLQREPLY process to obtain a new route to the TCP-BuS receiver, PN informs the TCP-BuS sender of the new partial path using the ERSN packet. When the TCP-BuS sender receives an ERSN packet, it resumes the data transmission. Since there is a chance for ERSN packet loss due to congestion in the network, it needs to be sent reliably. The TCP-BuS sender also periodically originates probe packets to check the availability of a path to the destination. Figure 9.7 shows an illustration of the propagation of ERDN and RN messages when a link between nodes 4 and 12 fails.

Operation of TCP-BuS.



When a TCP-BuS sender receives the ERSN message, it understands, from the sequence number of the last successfully received packet at the destination and the sequence number of the packet

at the head of the queue at PN, the packets lost in transition. The TCP-BuS receiver understands that the lost packets will be delayed further and hence uses a selective acknowledgment strategy instead of fast retransmission. These lost packets are retransmitted by the TCP-BuS sender. During the retransmission of these lost packets, the network congestion between the TCP-BuS sender and PN is handled in a way similar to that in traditional TCP.

Advantages and Disadvantages

The advantages of TCP-BuS include performance improvement and avoidance of fast retransmission due to the use of buffering, sequence numbering, and selective acknowledgment. TCP-BuS also takes advantage of the underlying routing protocols, especially the on-demand routing protocols such asABR. The disadvantages of TCP-BuS include the increased dependency on the routing protocol and the buffering at the intermediate nodes. The failure of intermediate nodes that buffer the packets may lead to loss of packets and performance degradation. The dependency of TCP-BuS on the routing protocol may degrade its performance with other routing protocols that do not have similar control messages as in ABR.

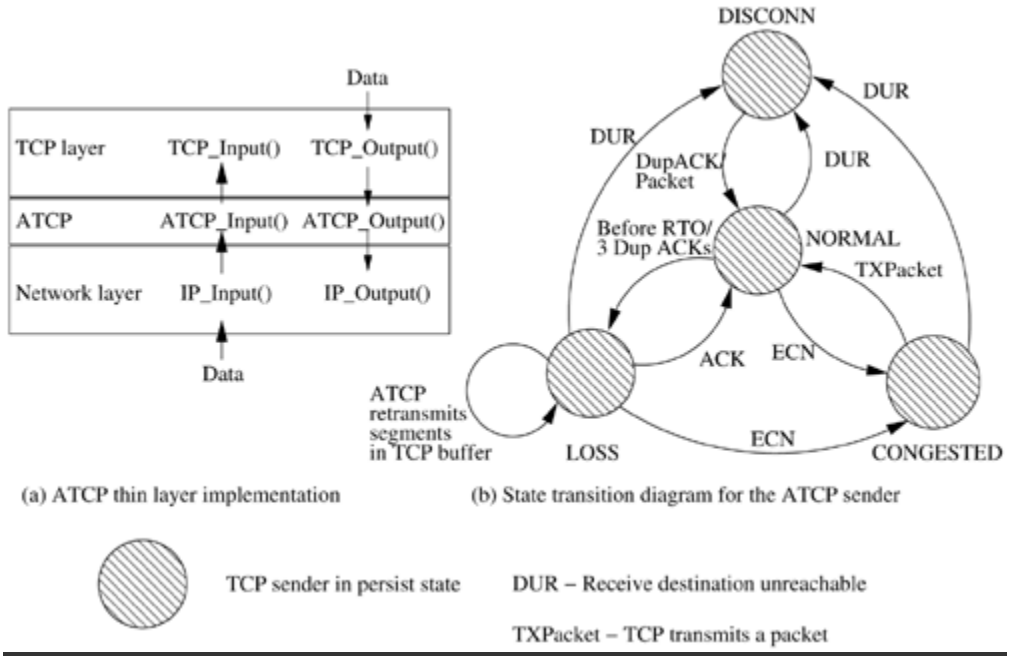
Ad Hoc TCP

Similar to TCP-F and TCP-ELFN, ad hoc TCP (ATCP) [12] also uses a network layer feedback mechanism to make the TCP sender aware of the status of the network path over which the TCPpackets are propagated. Based on the feedback information received from the intermediate nodes, theTCP sender changes its state to the *persist* state, *congestion control* state, or the *retransmit* state. When an intermediate node finds that the network is partitioned, then the TCP sender state is changed to the persist state where it avoids unnecessary retransmissions. When ATCP puts TCP in the persist state, it sets TCP's congestion window size to one in order to ensure that TCP does not continue using the old congestion window value. This forces TCP to probe the correct value of the congestion window to be used for the new route. If an intermediate node loses a packet due to error, then the ATCP at theTCP sender immediately retransmits it without invoking the congestion control algorithm. In order to be compatible with widely

deployed TCP-based networks, ATCP provides this feature without modifying the traditional TCP. ATCP is implemented as a thin layer residing between the IP and TCPprotocols. The ATCP layer essentially makes use of the explicit congestion notification (ECN) for maintenance of the states.

Figure (a) shows the thin layer implementation of ATCP between the traditional TCP layer and theIP layer. This does not require changes in the existing TCP protocol. This layer is active only at the TCPsender. The major function of the ATCP layer is to monitor the packets sent and received by the TCPsender, the state of the TCP sender, and the state of the network. Figure 9.8 (b) shows the state transition diagram for the ATCP at the TCP sender. The four states in the ATCP are (i) NORMAL, (ii) CONGESTED, (iii) LOSS, and (iv) DISCONN. When a TCP connection is established, the ATCP sender state is in NORMAL. In this state, ATCP does not interfere with the operation of TCP and it remains invisible.

An illustration of ATCP thin layer and ATCP state diagram.



When packets are lost or arrive out-of-order at the destination, it generates duplicate ACKs. In traditional TCP, upon reception of duplicate ACKs, the TCP sender retransmits the segment under consideration and shrinks the contention window. But the ATCP sender counts the number of duplicate ACKs received and if it reaches three, instead of forwarding the duplicate ACKs

to TCP, it puts TCP in the persist state and ATCP in the LOSS state. Hence, the TCP sender avoids invoking congestion control. In the LOSS state, ATCP retransmits the unacknowledged segments from the TCPbuffer. When a new ACK comes from the TCP receiver, it is forwarded to TCP and the TCP sender is removed from the persist state and then the ATCP sender changes to the NORMAL state. When the ATCP sender is in the LOSS state, the receipt of an ECN message or an ICMP *source quench*message changes it to the CONGESTED state. Along with this state transition, the ATCP sender removes the TCP sender from the persist state. When the network gets congested, the ECN4 flag is set in the data and the ACK packets. When the ATCP sender receives this ECN message in the normal state, it changes to the CONGESTED state and just remains invisible, permitting TCP to invoke normal congestion control mechanisms. When a route failure or a transient network partition occurs in the network, ATCP expects the network layer to detect these and inform the ATCP sender through an ICMPdestination unreachable (DUR) message. Upon reception of the DUR message, ATCP puts the TCPsender into the persist state and enters into the DISCONN state. It remains in the DISCONN state until it is connected and receives any data or duplicate ACKs. On the occurrence of any of these events, ATCPchanges to the NORMAL state. The connected status of the path can be detected by the acknowledgments for the periodic probe packets generated by the TCP sender. The receipt of an ICMPDUR message in the LOSS state or the CONGESTED state causes a transition to the DISCONN state. When ATCP puts TCP into the persist state, it sets the congestion window to one segment in order to make TCP probe for the new congestion window when the new route is available. In summary, ATCPtries to perform the activities listed in Table.

ECN is currently under consideration by IETF and is now a standard (IETF RFC 3168).

The actions taken by ATCP in table

Event	Action
Packet loss due to high BER	Retransmits the lost packets without reducing congestion window
Route recomputation delay	Makes the TCP sender go to persist state and stop transmission until new route has been found
Transient partitions	Makes the TCP sender go to persist state and stop transmission until new route has been found
Out-of-order packet delivery due to multipath routing	Maintains TCP sender unaware of this and retransmits the packets from TCP buffer
Change in route	Recomputes the congestion window

Advantages and Disadvantages

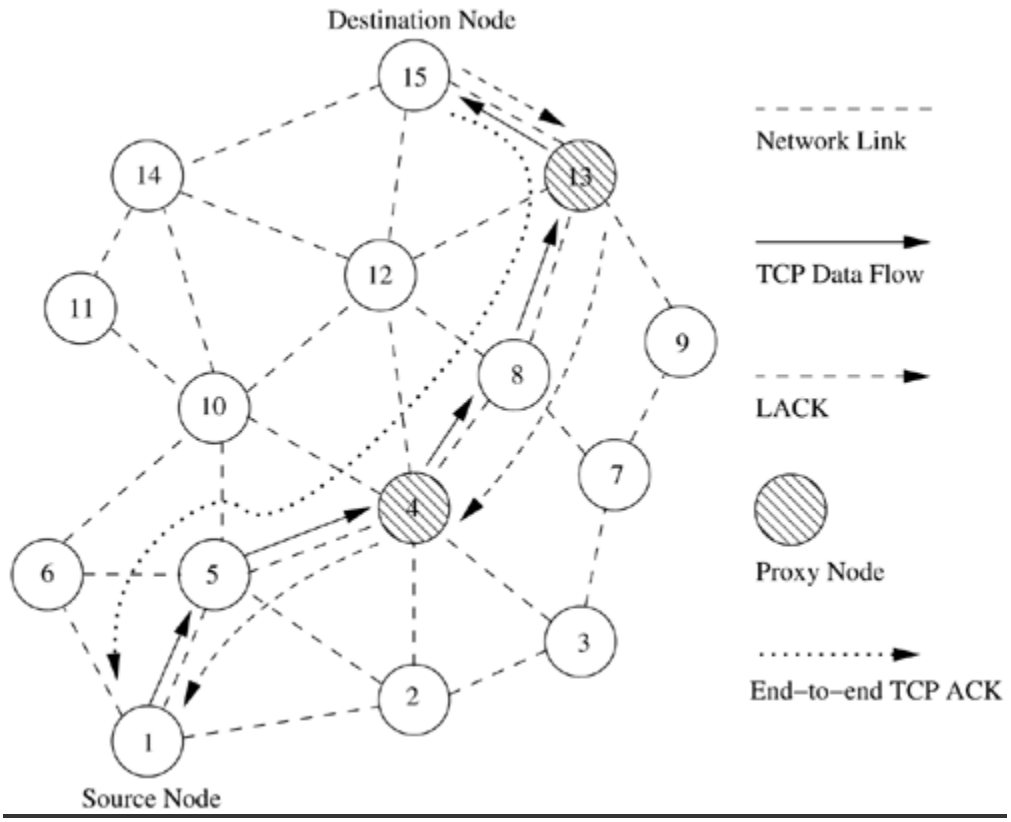
Two major advantages of ATCP are (i) it maintains the end-to-end semantics of TCP and (ii) it is compatible with traditional TCP. These advantages permit ATCP to work seamlessly with the Internet. In addition, ATCP provides a feasible and efficient solution to improve throughput of TCP in ad hoc wireless networks. The disadvantages of ATCP include (i) the dependency on the network layer protocol to detect the route changes and partitions, which not all routing protocols may implement and (ii) the addition of a thin ATCP layer to the TCP/IP protocol stack that requires changes in the interface functions currently being used.

SplitTCP

One of the major issues that affects the performance of TCP over ad hoc wireless networks is the degradation of throughput with increasing path length, as discussed early in this chapter. The short (*i.e.,* in terms of path length) connections generally obtain much higher throughput than long connections. This can also lead to unfairness among TCP sessions, where one session may obtain much higher throughput than other sessions. This unfairness problem is further worsened by the use of MACprotocols such as IEEE 802.11, which are found to give a higher throughput for certain link-level sessions, leading to an effect known as *channel capture* effect. This effect leads to certain flows capturing the channel for longer time durations, thereby reducing throughput for other flows. The channel capture effect can also lead to low overall system

throughput. The reader can refer to Chapter 6for more details on MAC protocols and throughput fairness. Split-TCP [13] provides a unique solution to this problem by splitting the transport layer objectives into congestion control and end-to-end reliability. The congestion control is mostly a local phenomenon due to the result of high contention and high traffic load in a local region. In the ad hoc wireless network environment, this demands local solutions. At the same time, reliability is an end-to-end requirement and needs end-to-end acknowledgments. In addition to splitting the congestion control and reliability objectives, split- TCP splits a long TCPconnection into a set of short concatenated TCP connections (called *segments* or *zones*) with a number of selected intermediate nodes (known as *proxy* nodes) as terminating points of these short connections. Figure 9.9 illustrates the operation of split-TCP where a three segment split-TCPconnection exists between source node 1 and destination node 15. A proxy node receives the TCPpackets, reads its contents, stores it in its local buffer, and sends an acknowledgment to the source (or the previous proxy). This acknowledgment called local acknowledgment (LACK) does not guarantee end-to-end delivery. The responsibility of further delivery of packets is assigned to the proxy node. A proxy node clears a buffered packet once it receives LACK from the immediate successor proxy node for that packet. Split-TCP maintains the end-to-end acknowledgment mechanism intact, irrespective of the addition of zone-wise LACKs. The source node clears the uffered packets only after receiving the end-to-end acknowledgment for those packets.

An illustration of Split-TCP.



In Figure 9.9, node 1 initiates a TCP session to node 15. Node 4 and node 13 are chosen as proxy nodes. The number of proxy nodes in a TCP session is determined by the length of the path between source and destination nodes. Based on a distributed algorithm, the intermediate nodes that receive TCPpackets determine whether to act as a proxy node or just as a simple forwarding node. The most simple algorithm makes the decision for acting as proxy node if the packet has already traversed more than a predetermined number of hops from the last proxy node or the sender of the TCP session. In Figure 9.9, the path between node 1 and node 4 is the first zone (segment), the path between nodes 4 and 13 is the second zone (segment), and the last zone is between node 13 and 15.

The proxy node 4, upon receipt of each TCP packet from source node 1, acknowledges it with a LACKpacket, and buffers the received packets. This buffered packet is forwarded to the next proxy node (in this case, node 13) at a transmission rate proportional to the arrival of LACKs from the next proxy node or destination. The transmission control window at the TCP sender is also split into two windows, that is, the congestion window and the end-to-end window.

The congestion window changes according to the rate of arrival of LACKs from the next proxy node and the end-to-end window is updated based on the arrival of end-to-end ACKs. Both these windows are updated as per traditional TCP except that the congestion window should stay within the end to- end window. In addition to these transmission windows at the TCP sender, every proxy node maintains a congestion window that governs the segment level transmission rate.

Advantages and Disadvantages

Split-TCP has the following advantages: (i) improved throughput, (ii) improved throughput fairness, and (iii) lessened impact of mobility. Throughput improvement is due to the reduction in the effective transmission path length (number of hops in a zone or a path segment). TCP throughput degrades with increasing path length. Split-TCP has shorter concatenated path segments, each operating at its own transmission rate, and hence the throughput is increased.

This also leads to improved throughput fairness in the system. Since in split- TCP, the path segment length can be shorter than the end-to-end path length, the effect of mobility on throughput is lessened. The disadvantages of split-TCP can be listed as follows: (i) It requires modifications to TCP protocol, (ii) the end-to-end connection handling of traditional TCP is violated, and (iii) the failure of proxy nodes can lead to throughput degradation. The traditional TCP has end-to-end semantics, where the intermediate nodes do not process TCP packets, whereas in split-TCP, the intermediate nodes need to process the TCP packets and hence, in addition to the loss of end-to-end semantics, certain security schemes that require IP payload encryption cannot be used. During frequent path breaks or during frequent node failures, the performance of split-TCP may be affected.

A Comparison of TCP Solutions for Ad Hoc Wireless Networks

Issue	TCP-F	TCP-ELFN	TCP-BuS	ATCP	Split-TCP
Packet loss due to BER or collision	Same as TCP	Same as TCP	Same as TCP	Retransmits the lost packets without invoking congestion control	Same as TCP
Path breaks	RFN is sent to the TCP sender and state changes to snooze	ELFN is sent to the TCP sender and state changes to standby	ERDN is sent to the TCP sender, state changes to snooze, ICMP DUR is sent to the TCP sender, and ATCP puts TCP into persist state	Same as TCP	Same as TCP
Out-of-order packets	Same as TCP	Same as TCP	Out-of-order packets reached after a path recovery are handled	ATCP reorders packets and hence TCP avoids sending duplicates	Same as TCP
Congestion	Same as TCP	Same as TCP	Explicit messages such as ICMP source quench are used	ECN is used to notify TCP sender. Congestion control is same as TCP	Since connection is split, the congestion control is handled within a zone by proxy nodes
Congestion window after path reestablishment	Same as before the path break	Same as before the path break	Same as before the path break	Recomputed for new route	Proxy nodes maintain congestion window and handle congestion
Explicit path break notification	Yes	Yes	Yes	Yes	No
Explicit path reestablishment notification	Yes	No	Yes	No	No
Dependency on routing protocol	Yes	Yes	Yes	Yes	No
End-to-end semantics	Yes	Yes	Yes	Yes	No
Packets buffered at intermediate nodes	No	No	Yes	No	Yes

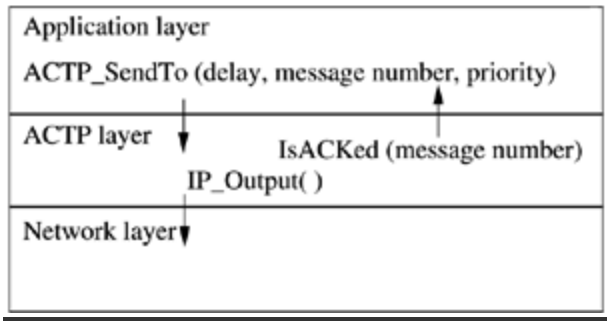
Application Controlled Transport Protocol(ACTP)

Unlike the TCP solutions discussed earlier in this chapter, application controlled transport protocol (ACTP5) [14] is a light-weight transport layer protocol. It is not an extension to TCP. ACTP assigns the responsibility of ensuring reliability to the application layer. It is more like UDP with feedback of delivery and state maintenance. ACTP stands in between TCP and UDP where TCP experiences low performance with high reliability and UDP provides better performance with high packet loss in ad hoc wireless networks. 5 Originally called ATP, for differentiating with ad hoc transport protocol it is referred to as ACTP in this chapter. The key design philosophy of ACTP is to leave the provisioning of reliability to the application layer and provide a simple feedback information about the delivery status of packets to the application layer. ACTP supports the priority of packets to be delivered, but it is the responsibility of the lower layers to actually provide a differentiated service based on this priority.

Figure shows the ACTP layer and the API functions used by the application layer to interact with the ACTP layer. Each API function call to send a packet [*SendTo()*] contains the additional

information required for ACTP such as the maximum delay the packet can tolerate (delay), the message number of the packet, and the priority of the packet. The message number is assigned by the application layer, and it need not to be in sequence. The priority level is assigned for every packet by the application. It can be varied across packets in the same flow with increasing numbers referring to higher priority packets. The non-zero value in the message number field implicitly conveys that the application layer expects a delivery status information about the packet to be sent. This delivery status is maintained at the ACTP layer, and is available to the application layer for verification through another API function*IsACKed<message number>*. The delivery status returned by *IsACKed<message number>* function call can reflect (i) a successful delivery of the packet (ACK received), (b) a possible loss of the packet (no ACK received and the deadline has expired), (iii) remaining time for the packet (no ACK received but the deadline has not expired), and (iv) no state information exists at the ACTP layer regarding the message under consideration. A zero in the delay field refers to the highest priority packet, which requires immediate transmission with minimum possible delay. Any other value in the delay field refers to the delay that the message can experience. On getting the information about the delivery status, the application layer can decide on retransmission of a packet with the same old priority or with an updated priority. Well after the packet's lifetime expires, ACTP clears the packet's state information and delivery status. The packet's lifetime is calculated as 4 × retransmit timeout (RTO) and is set as the lifetime when the packet is sent to the network layer. A node estimates the RTO interval by using the round-trip time between the transmission time of a message and the time of reception of the corresponding ACK. Hence, the RTO value may not be available if there are no existing reliable connections to a destination. A packet without any message number (*i.e.*, no delivery status required) is handled exactly the same way as in UDP without maintaining any state information.

An illustration of the interface functions used in ACTP.



Advantages and Disadvantages

One of the most important advantages of ACTP is that it provides the freedom of choosing the required reliability level to the application layer. Since ACTP is a light-weight transport layer protocol, it is scalable for large networks. Throughput is not affected by path breaks as much as in TCP as there is no congestion window for manipulation as part of the path break recovery. One disadvantage of ACTP is that it is not compatible with TCP. Use of ACTP in a very large ad hoc wireless network can lead to heavy congestion in the network as it does not have any congestion control mechanism.

Ad Hoc Transport Protocol

Ad hoc transport protocol (ATP) [15] is specifically designed for ad hoc wireless networks and is not a variant of TCP. The major aspects by which ATP defers from TCP are (i) coordination among multiple layers, (ii) rate based transmissions, (iii) decoupling congestion control and reliability, and (iv) assisted congestion control. Similar to other TCP variants proposed for ad hoc wireless networks, ATP uses services from network and MAC layers for improving its performance. ATP uses information from lower layers for (i) estimation of the initial transmission rate, (ii) detection, avoidance, and control of congestion, and (iii) detection of path breaks.

Unlike TCP, ATP utilizes a timer-based transmission, where the transmission rate is decided by the granularity of the timer which is dependent on the congestion in the network. The congestion control mechanism is decoupled from the reliability and flow control mechanisms. The network congestion information is obtained from the intermediate nodes, whereas the flow control and reliability information are obtained from the ATP receiver. The intermediate nodes attach the congestion information to every ATP packet and the ATP receiver collates it before including it

in the next ACK packet. The congestion information is expressed in terms of the weighted averaged6 queuing delay (*DQ*) and contention delay (*DC*) experienced by the packets at every intermediate node. The field in which this delay information is included is referred to as the *rate feedback field* and the transmission rate is the inverse of the delay information contained in the rate feedback field. Intermediate nodes attach the current delay information to every ATP data packet if the already existing value is smaller than the current delay. The ATP receiver collects this delay information and the weighted average value is attached in the periodic ACK (ATP uses SACK mechanism, hence ACK refers to SACK) packet sent back to the ATP sender. During a connection startup process or when ATP recovers from a path break, the transmission rate to be used is determined by a process called *quick start.* During the quick start process, the ATP sender propagates a probe packet to which the intermediate nodes attach the transmission rate (in the form of current delay), which is received by the ATP receiver, and an ACK is sent back to the ATP sender. The ATP sender starts using the newly obtained transmission rate by setting the data transmission timers. During a connection startup, the connection request and the ACK packets are used as probe packets in order to reduce control overhead. When there is no traffic around an intermediate node, the transmission delay is approximated as $\beta \times (DQ + DC)$, where β is the factor that considers the induced traffic load. This is to consider the induced load (load on a particular link due to potential contention introduced by the upstream and downstream nodes in the path) when the actual transmission begins. A default value of 3 is used for β. ATP uses SACK packets periodically to ensure the selective retransmission of lost packets, which ensures the reliability of packet delivery. The SACK period is chosen such that it is more than the round-trip time and can track the network dynamics. The receiver performs a weighted average of the delay/transmission rate information for every incoming packet to obtain the transmission rate for an ATP flow and this value is included in the subsequent SACK packet it sends. In addition to the rate feedback, the ATP receiver includes flow control information in the SACK packets.

Advantages and Disadvantages

The major advantages of ATP include improved performance, decoupling of the congestion control and reliability mechanisms, and avoidance of congestion window fluctuations. ATP does not maintain any per flow state at the intermediate nodes. The congestion information is gathered

directly from the nodes that experience it. The major disadvantage of ATP is the lack of interoperability with TCP. As TCP is a widely used transport layer protocol, interoperability with TCP servers and clients in the Internet is important in many applications. For large ad hoc wireless networks, the fine-grained per-flow timer used at the ATPsender may become a scalability bottleneck in resource-constrained mobile nodes.

SECURITY IN AD HOC WIRELESS NETWORKS

As mentioned earlier, due to the unique characteristics of ad hoc wireless networks, such networks are highly vulnerable to security attacks compared to wired networks or infrastructure-based wireless networks. The following sections discuss the various security requirements in ad hoc wireless networks, the different types of attacks possible in such networks, and some of the solutions proposed for ensuring network security.

NETWORK SECURITY REQUIREMENTS

A security protocol for ad hoc wireless networks should satisfy the following requirements. The requirements listed below should in fact be met by security protocols for other types of networks also.

• **Confidentiality:** The data sent by the sender (source node) must be comprehensible only to the intended receiver (destination node). Though an intruder might get hold of the data being sent, he/she must not be able to derive any useful information out of the data. One of the popular techniques used for ensuring confidentiality is data encryption.

• **Integrity:** The data sent by the source node should reach the destination node as it was sent: unaltered. In other words, it should not be possible for any malicious node in the network to tamper with the data during transmission.

• **Availability:** The network should remain operational all the time. It must be robust enough to tolerate link failures and also be capable of surviving various attacks mounted on it. It should be able to provide the guaranteed services whenever an authorized user requires them.

• **Non-repudiation:** Non-repudiation is a mechanism to guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message. Digital signatures, which function as unique identifiers for each user, much like a written signature, are used commonly for this purpose.

ISSUES AND CHALLENGES IN SECURITY PROVISIONING

Designing a foolproof security protocol for ad hoc wireless is a very challenging task. This is mainly because of certain unique characteristics of ad hoc wireless networks, namely, shared broadcast radio channel, insecure operating environment, lack of central authority, lack of association among nodes, limited availability of resources, and physical vulnerability. A detailed discussion on how each of the above mentioned characteristics causes difficulty in providing security in ad hoc wireless networks is given below.

• **Shared broadcast radio channel:** Unlike in wired networks where a separate dedicated transmission line can be provided between a pair of end users, the radio channel used for communication in ad hoc wireless networks is broadcast in nature and is shared by all nodes in the network. Data transmitted by a node is received by all nodes within its direct transmission range. So a malicious node could easily obtain data being transmitted in the network. This problem can be minimized to a certain extent by using directional antennas.

• **Insecure operational environment:** The operating environments where ad hoc wireless networks are used may not always be secure. One important application of such networks is in battlefields. In such applications, nodes may move in and out of hostile and insecure enemy territory, where they would be highly vulnerable to security attacks.

• **Lack of central authority:** In wired networks and infrastructure-based wireless networks, it would be possible to monitor the traffic on the network through certain important central points (such as routers, base stations, and access points) and implement security mechanisms at such points. Since ad hoc wireless networks do not have any such central points, these mechanisms cannot be applied in ad hoc wireless networks.

• **Lack of association:** Since these networks are dynamic in nature, a node can join or leave the network at any point of the time. If no proper authentication mechanism is used for associating nodes with a network, an intruder would be able to join into the network quite easily and carry out his/her attacks.

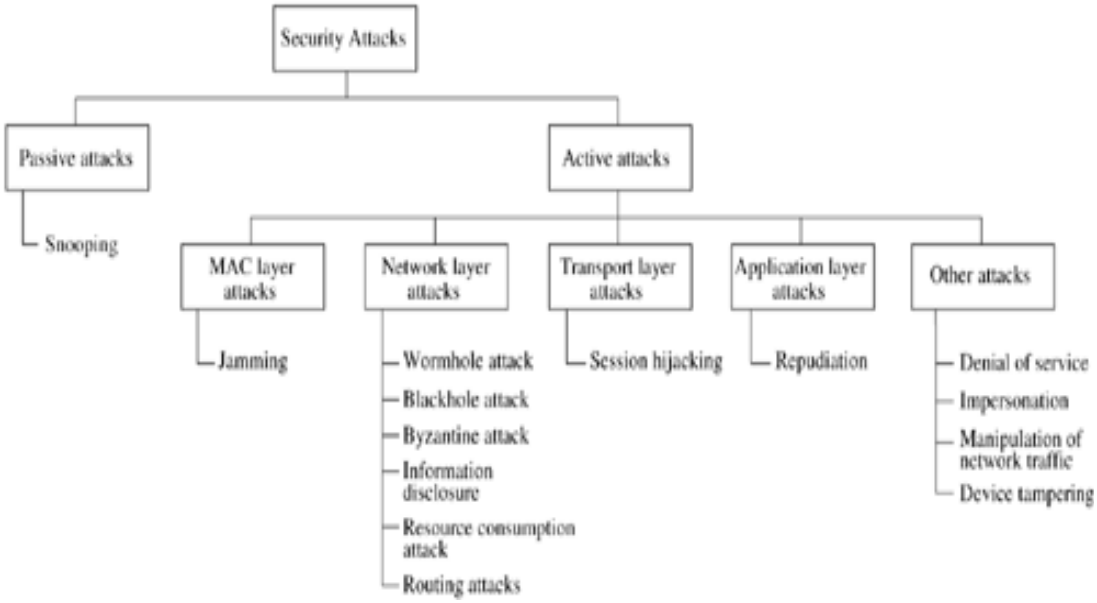
•**Limited resource availability:** Resources such as bandwidth, battery power, and computational power (to a certain extent) are scarce in ad hoc wireless networks. Hence, it is difficult to implement complex cryptography-based security mechanisms in such networks.

• **Physical vulnerability:** Nodes in these networks are usually compact and hand-held in nature. They could get damaged easily and are also vulnerable to theft.

NETWORK SECURITY ATTACKS

Attacks on ad hoc wireless networks can be classified into two broad categories, namely, *passive* and*active* attacks. A passive attack does not disrupt the operation of the network; the adversary snoops the data exchanged in the network without altering it. Here, the requirement of confidentiality can be violated if an adversary is also able to interpret the data gathered through snooping. Detection of passive attacks is very difficult since the operation of the network itself does not get affected. One way of overcoming such problems is to use powerful encryption mechanisms to encrypt the data being transmitted, thereby making it impossible for eavesdroppers to obtain any useful information from the data overheard. An active attack attempts to alter or destroy the data being exchanged in the network, thereby disrupting the normal functioning of the network. Active attacks can be classified further into two categories, namely, *external* and *internal* attacks. External attacks are carried out by nodes that do not belong to the network. These attacks can be prevented by using standard security mechanisms such as encryption techniques and firewalls.7 Internal attacks are from compromised nodes that are actually part of the network. Since the adversaries are already part of the network as authorized nodes, internal attacks are more severe and difficult to detect when compared to external attacks.

Classifications of attacks



Network Layer Attacks

This section lists and gives brief descriptions of the attacks pertaining to the network layer in the network protocol stack.

• **Wormhole attack:** In this attack, an attacker receives packets at one location in the network and tunnels them (possibly selectively) to another location in the network, where the packets are resent into the network [16]. This tunnel between two colluding attackers is referred to as a wormhole. It could be established through a single long-range wireless link or even through a wired link between the two colluding attackers. Due to the broadcast nature of the radio channel, the attacker can create a wormhole even for packets not addressed to itself. Though no harm is done if the wormhole is used properly for efficient relaying of packets, it puts the attacker in a powerful position compared to other nodes in the network, which the attacker could use in a manner that could compromise the security of the network. If proper mechanisms are not employed to defend the network against wormhole attacks, most of the existing routing protocols for ad hoc wireless networks may fail to find valid routes.

• **Blackhole attack:** In this attack, a malicious node falsely advertises good paths (*e.g.*, shortest path or most stable path) to the destination node during the path-finding process (in on-demand routing protocols) or in the route update messages (in table-driven routing protocols). The intention of the malicious node could be to hinder the path-finding process or to intercept all data packets being sent to the destination node concerned.

• **Byzantine attack:** Here, a compromised intermediate node or a set of compromised intermediate nodes works in collusion and carries out attacks such as creating routing loops, routing packets on non-optimal paths, and selectively dropping packets [17]. Byzantine failures are hard to detect. The network would seem to be operating normally in the viewpoint of the nodes, though it may actually be exhibiting Byzantine behavior.

• **Information disclosure:** A compromised node may leak confidential or important information to unauthorized nodes in the network. Such information may include information regarding the network topology, geographic location of nodes, or optimal routes to authorized nodes in the network.

• **Resource consumption attack:** In this attack, a malicious node tries to consume/waste away resources of other nodes present in the network. The resources that are targeted are battery power, bandwidth, and computational power, which are only limitedly available in ad hoc

wireless networks. The attacks could be in the form of unnecessary requests for routes, very frequent generation of beacon packets, or forwarding of stale packets to nodes. Using up the battery power of another node by keeping that node always busy by continuously pumping packets to that node is known as a sleep deprivation attack.

• **Routing attacks:** There are several types attacks mounted on the routing protocol which are aimed at disrupting the operation of the network. In what follows, the various attacks on the routing protocol are described briefly.

1.Routing table overflow: In this type of attack, an adversary node advertises routes to non-existent nodes, to the authorized nodes present in the network. The main objective of such an attack is to cause an overflow of the routing tables, which would in turn prevent the creation of entries corresponding to new routes to authorized nodes. Proactive routing protocols are more vulnerable to this attack compared to reactive routing protocols.

2 Routing table poisoning: Here, the compromised nodes in the networks send fictitious routing updates or modify genuine route update packets sent to other uncompromised nodes. Routing table poisoning may result in sub-optimal routing, congestion in portions of the network, or even make some parts of the network inaccessible.

3 Packet replication: In this attack, an adversary node replicates stale packets. This consumes additional bandwidth and battery power resources available to the nodes and also causes unnecessary confusion in the routing process.

4 Route cache poisoning: In the case of on-demand routing protocols (such as the AODV protocol [18]), each node maintains a route cache which holds information regarding routes that have become known to the node in the recent past. Similar to routing table poisoning, an adversary can also poison the route cache to achieve similar objectives.

5 Rushing attack: On-demand routing protocols that use duplicate suppression during the route discovery process are vulnerable to this attack [19]. An adversary node which receives a *RouteRequest*packet from the source node floods the packet quickly throughout the network before other nodes which also receive the same *RouteRequest* packet can react. Nodes that receive the legitimate*RouteRequest* packets assume those packets to be duplicates of the packet already received through the adversary node and hence discard those packets. Any route discovered by the source node would contain the adversary node as one of the intermediate nodes. Hence, the source node would not be able to find secure routes, that is, routes that do not

include the adversary node. It is extremely difficult to detect such attacks in ad hoc wireless networks.

Transport Layer Attacks

This section discusses an attack which is specific to the transport layer in the network protocol stack.

• **Session hijacking:** Here, an adversary takes control over a session between two nodes. Since most authentication processes are carried out only at the start of a session, once the session between two nodes gets established, the adversary node masquerades as one of the end nodes of the session and hijacks the session.

Application Layer Attacks

This section briefly describes a security flaw associated with the application layer in the network protocol stack.

• **Repudiation:** In simple terms, repudiation refers to the denial or attempted denial by a node involved in a communication of having participated in all or part of the communication. As mentioned in Section 9.8, non-repudiation is one of the important requirements for a security protocol in any communication network.

Other Attacks

Multi-layer Attacks

Multi-layer attacks are those that could occur in any layer of the network protocol stack. Denial of service and impersonation are some of the common multi-layer attacks. This section discusses some of the multi-layer attacks in ad hoc wireless networks.

• **Denial of Service:** In this type of attack, an adversary attempts to prevent legitimate and authorized users of services offered by the network from accessing those services. A denial of service (DoS) attack can be carried out in many ways. The classic way is to flood packets to any centralized resource (*e.g.*, an access point) used in the network so that the resource is no longer available to nodes in the network, resulting in the network no longer operating in the manner it was designed to operate. This may lead to a failure in the delivery of guaranteed services to the end users. Due to the unique characteristics of ad hoc wireless networks, there exist many more ways to launch a DoS attack in such a network, which would not be possible in wired networks.

DoS attacks can be launched against any layer in the network protocol stack [20]. On the physical and MAC layers, an adversary could employ jamming signals which disrupt the on-going transmissions on the wireless channel. On the network layer, an adversary could take part in the routing process and exploit the routing protocol to disrupt the normal functioning of the network. For example, an adversary node could participate in a session but simply drop a certain number of packets, which may lead to degradation in the QoS being offered by the network. On the higher layers, an adversary could bring down critical services such as the key management service (key management will be described in detail in the next section). Some of the DoS attacks are described below.

– **Jamming:** In this form of attack, the adversary initially keeps monitoring the wireless medium in order to determine the frequency at which the receiver node is receiving signals from the sender. It then transmits signals on that frequency so that error-free reception at the receiver is hindered. Frequency hopping spread spectrum (FHSS) and direct sequence spread spectrum (DSSS) (described in detail in the first chapter of this book) are two commonly used techniques that overcome jamming attacks.

– **SYN flooding:** Here, an adversary sends a large number of SYN packets[8] to a victim node, spoofing the return addresses of the SYN packets. On receiving the SYN packets, the victim node sends back acknowledgment (SYN-ACK) packets to nodes whose addresses have been specified in the received SYN packets. However, the victim node would not receive any ACK packet in return. In effect, a half-open connection gets created. The victim node builds up a table/data structure for holding information regarding all pending connections. Since the maximum possible size of the table is limited, the increasing number of half-open connections results in an overflow in the table. Hence, even if a connection request comes from a legitimate node at a later point of time, because of the table overflow, the victim node would be forced to reject the call request. [8] SYN packets are used to establish an end-to-end session between two nodes at the transport layer.

– **Distributed DoS attack:** A more severe form of the DoS attack is the distributed DoS (DDoS) attack. In this attack, several adversaries that are distributed throughout the network collude and prevent legitimate users from accessing the services offered by the network.

• **Impersonation:** In impersonation attacks, an adversary assumes the identity and privileges of an authorized node, either to make use of network resources that may not be available to it under

normal circumstances, or to disrupt the normal functioning of the network by injecting false routing information into the network. An adversary node could masquerade as an authorized node using several methods. It could by chance guess the identity and authentication details of the authorized node (target node), or it could snoop for information regarding the identity and authentication of the target node from a previous communication, or it could circumvent or disable the authentication mechanism at the target node. A *man-in-the-middle* attack is another type of impersonation attack. Here, the adversary reads and possibly modifies, messages between two end nodes without letting either of them know that they have been attacked. Suppose two nodes *X* and *Y* are communicating with each other; the adversary impersonates node *Y* with respect to node *X* and impersonates node *X* with respect to node *Y*, exploiting the lack of third-party authentication of the communication between nodes *X* and *Y*.

Device Tampering

Unlike nodes in a wired network, nodes in ad hoc wireless networks are usually compact, soft, and hand-held in nature. They could get damaged or stolen easily.

KEY MANAGEMENT

Having seen the various kinds of attacks possible on ad hoc wireless networks, we now look at various techniques employed to overcome the attacks. Cryptography is one of the most common and reliable means to ensure security. Cryptography is not specific to ad hoc wireless networks. It can be applied to any communication network. It is the study of the principles, techniques, and algorithms by which information is transformed into a disguised version which no unauthorized person can read, but which can be recovered in its original form by an intended recipient. In the parlance of cryptography, the original information to be sent from one person to another is called *plaintext*. This plaintext is converted into *ciphertext* by the process of encryption, that is, the application of certain algorithms or functions. An authentic receiver can decrypt/decode the ciphertext back into plaintext by the process of decryption. The processes of encryption and decryption are governed by *keys*, which are small amounts of information used by the cryptographic algorithms. When the key is to be kept secret to ensure the security of the system, it is called a secret key. The secure administration of cryptographic keys is called key management.

The four main goals of cryptography are confidentiality, integrity, authentication (the receiver should be able to identify the sender and verify that the message actually came from that sender), and non-repudiation. There are two major kinds of cryptographic algorithms: symmetric key algorithms, which use the same key for encryption and decryption, and asymmetric key algorithms, which use two different keys for encryption and decryption. Symmetric key algorithms are usually faster to execute electronically, but require a secret key to be shared between the sender and receiver. When communication needs to be established among a group of nodes, each sender-receiver pair should share a key, which makes the system nonscalable. If the same key is used among more than two parties, a breach of security at any one point makes the whole system vulnerable. The asymmetric key algorithms are based on some mathematical principles which make it infeasible or impossible to obtain one key from another; therefore, one of thekeys can be made public while the other is kept secret (private). This is called Public key cryptography. Such systems are used extensively in practice, but are not provably secure. They rely upon the difficulty of solving certain mathematical problems, and the network would be open to attacks once the underlying mathematical problem is solved.

Symmetric Key Algorithms

Symmetric key algorithms rely on the presence of the shared key at both the sender and receiver, which has been exchanged by some previous arrangement. There are two kinds of symmetric key algorithms, one involving block ciphers and the other stream ciphers. A block cipher is an encryption scheme in which the plaintext is broken into fixed-length segments called blocks, and the blocks are encrypted one at a time. The simplest examples include substitution and transposition. In substitution, each alphabet of the plaintext is substituted by another in the ciphertext, and this table mapping the original and the substituted alphabet is available at both the sender and receiver. A transposition cipher permutes the alphabet in the plaintext to produce the ciphertext. Figure (a) illustrates the encryption using substitution, and Figure 9.12 (b) shows a transposition cipher. The block length used is five.

Substitution and transposition.

Original Alphabet	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Substitution	E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
Plaintext	EVERYDAY CREATES A HISTORY EVERY DAYCR EATES AHIST ORY
Ciphertext	IZIVC HECGV IEXIW ELMWX SVC

(a)

Transposition	1 2 3 4 5 ↓ 3 5 1 4 2
Plaintext	EVERYDAY CREATES A HISTORY EVERY DAYCR EATES AHIST ORY
Ciphertext	EYERV YRDCA TSEEA ITASH YOR

(b)

A stream cipher is, in effect, a block cipher of block length one. One of the simplest stream ciphers is the Vernam cipher, which uses a key of the same length as the plaintext for encryption. For example, if the plaintext is the binary string 10010100, and the key is 01011001, then the encrypted string is given by the XOR of the plaintext and key, to be 11001101. The plaintext is again recovered by XORing the ciphertext with the same key. If the key is randomly chosen, transported securely to the receiver, and used for only one communication, this forms the one-time pad which has proven to be the most secure of all cryptographic systems. The only bottleneck here is to be able to securely send the key to the receiver.

Asymmetric Key Algorithms

Asymmetric key (or public key) algorithms use different keys at the sender and receiver ends for encryption and decryption, respectively. Let the encryption process be represented by a function E, and decryption by D. Then the plaintext m is transformed into the ciphertext c as $c = E(m)$. The receiver then decodes c by applying D. Hence, D is such that $m = D(c) = D(E(m))$. When

this asymmetric key concept is used in public key algorithms, the key *E* is made public, while *D* is private, known only to the intended receiver. Anyone who wishes to send a message to this receiver encrypts it using *E*. Though *c*can be overheard by adversaries, the function *E* is based on a computationally difficult mathematical problem, such as the factorization of large prime numbers. Hence, it is not possible for adversaries to derive *D* given *E*. Only the receiver can decrypt *c* using the private key *D*. A very popular example of public key cryptography is the RSA system developed by Rivest, Shamir, and Adleman, which is based on the integer factorization problem.

Digital signatures schemes are also based on public key encryption. In these schemes, the functions *E*and *D* are chosen such that $D(E(m)) = E(D(m)) = m$ for any message *m*. These are called reversible public key systems. In this case, the person who wishes to sign a document encrypts it using his/her private key *D*, which is known only to him/her. Anybody who has his/her public key *E* can decrypt it and obtain the original document, if it has been signed by the corresponding sender. In practice, a trusted third party (TTP) is agreed upon in advance, who is responsible for issuing these digital signatures (*D* and *E* pairs) and for resolving any disputes regarding the signatures. This is usually a governmental or business organization.

Key Management Approaches

The primary goal of key management is to share a secret (some information) among a specified set of participants. There are several methods that can be employed to perform this operation, all of them requiring varying amounts of initial configuration, communication, and computation. The main approaches to key management are key predistribution, key transport, key arbitration, and key agreement .

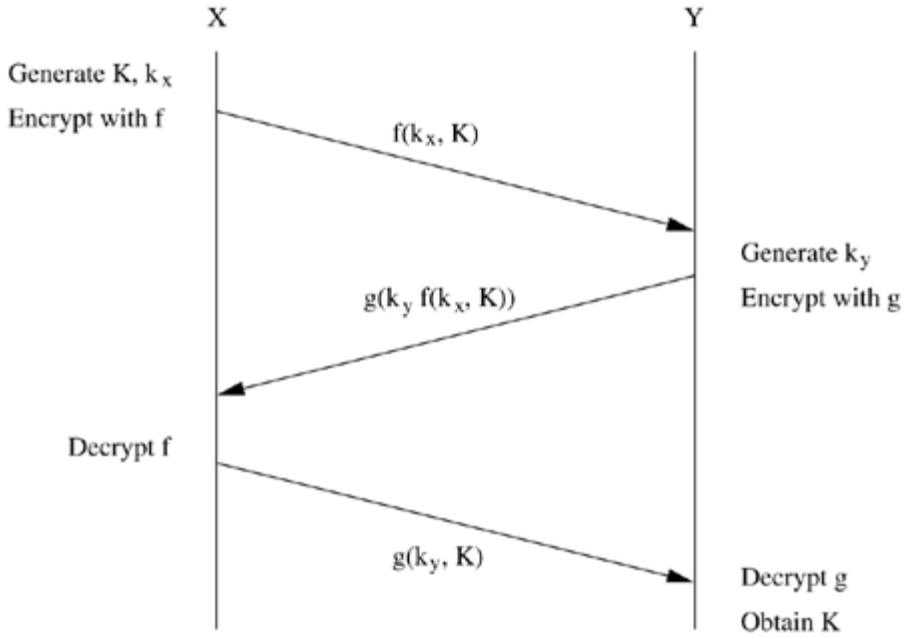
Key Predistribution

Key predistribution, as the name suggests, involves distributing keys to all interested parties before the start of communication. This method involves much less communication and computation, but all participants must be known *a priori*, during the initial configuration. Once deployed, there is no mechanism to include new members in the group or to change the key. As an improvement over the basic predistribution scheme, sub-groups may be formed with in the group, and some communication can be restricted to a subgroup. However, the formation of sub-groups is also an *a priori* decision with no flexibility during the operation.

Key Transport

In key transport systems, one of the communicating entities generates keys and transports them to the other members. The simplest scheme assumes that a shared key already exists among the participating members. This prior shared key is used to encrypt a new key and is transmitted to all corresponding nodes. Only those nodes which have the prior shared key can decrypt it. This is called the key encrypting key (KEK) method. However, the existence of a prior key cannot always be assumed. If the public key infrastructure (PKI) is present, the key can be encrypted with each participant's public key and transported to it. This assumes the existence of a TTP, which may not be available for ad hoc wireless networks. An interesting method for key transport without prior shared keys is the Shamir's three-pass protocol [22]. The scheme is based on a special type of encryption called commutative encryption schemes [which are reversible and composable (composition of two functions *f* and *g* is defined as $f(g(x))$)].

Consider two nodes *X* and *Y* which wish to communicate. Node *X* selects a key *K* which it wants to use in its communication with node *Y*. It then generates another random key *kx*, using which it encrypts *K*with *f*, and sends to node *Y*. Node *Y* encrypts this with a random key *ky* using *g*, and sends it back to node *X*. Now, node *X* decrypts this message with its key *kx*, and after applying the inverse function *f*-1, sends it to node *Y*. Finally, node *Y* decrypts the message using *ky* and *g*-1 to obtain the key *K*. The message exchanges of the protocol are illustrated in Figure .



Key Arbitration

Key arbitration schemes use a central arbitrator to create and distribute keys among all participants. Hence, they are a class of key transport schemes. Networks which have a fixed infrastructure use theAP as an arbitrator, since it does not have stringent power or computation constraints. In ad hoc wireless networks, the problem with implementation of arbitrated protocols is that the arbitrator has to be powered on at all times to be accessible to all nodes. This leads to a power drain on that particular node. An alternative would be to make the keying service distributed, but simple replication of the arbitration at different nodes would be expensive for resource-constrained devices and would offer many points of vulnerability to attacks. If any one of the replicated arbitrators is attacked, the security of the whole system breaks down.

Key Agreement

Most key agreement schemes are based on asymmetric key algorithms. They are used when two or more people want to agree upon a secret key, which will then be used for further communication. Key agreement protocols are used to establish a secure context over which a session can be run, starting with many parties who wish to communicate and an insecure channel. In group key agreement schemes, each participant contributes a part to the secret key. These need the least amount of preconfiguration, but such schemes have high computational complexity. The most popular key agreement schemes use the Diffie-Hellman exchange [21], an asymmetric key algorithm based on discrete logarithms

Key Management in Ad Hoc Wireless Networks

Ad hoc wireless networks pose certain specific challenges in key management due to the lack of infrastructure in such networks. Three types of infrastructure have been identified in [23], which are absent in ad hoc wireless networks. The first is the network infrastructure, such as dedicated routers and stable links, which ensure communication with all nodes. The second missing infrastructure is services such as name resolution, directory, and TTPs. The third missing infrastructure in ad hoc wireless networks is the administrative support of certifying authorities.

Password-Based Group Systems Several solutions for group keying in ad hoc wireless networks have been suggested in [23]. The example scenario for implementation is a meeting room, where different mobile devices want to start a secure session. Here, the parties involved in the session are to be identified based on their location, that is, all devices in the room can be part of the

session. Hence, relative location is used as the criterion for access control. If a TTP which knows the location of the participants exists, then it can implement location-based access control. A prior shared secret can be obtained by a physically more secure medium such as a wired network. This secret can be obtained by plugging onto a wired network first, before switching to the wireless mode. A password-based system has been explored where, in the simplest case, a long string is given as the password for users for one session. However, human beings tend to favor natural language phrases as passwords, over randomly generated strings. Such passwords, if used as keys directly during a session, are very weak and open to attack because of high redundancy, and the possibility of reuse over different sessions. Hence, protocols have been proposed to derive a strong key (not vulnerable to attacks) from the weak passwords given by the participants. This password-based system could be two-party, with a separate exchange between any two participants, or it could be for the whole group, with a leader being elected to preside over the session. Leader election is a special case of establishing an order among all participants. The protocol used is as follows. Each participant generates a random number, and sends it to all others. When every node has received the random number of every other node, a common predecided function is applied on all the numbers to calculate a *reference value*. The nodes are ordered based on the difference between their random number and the reference value.

Threshold Cryptography

Public key infrastructure (PKI) enables the easy distribution of keys and is a scalable method. Each node has a public/private key pair, and a certifying authority (CA) can bind the keys to the particular node. But the CA has to be present at all times, which may not be feasible in ad hoc wireless networks. It is also not advisable to simply replicate the CA at different nodes. In [20], a scheme based on threshold cryptography has been proposed by which n servers exist in the ad hoc wireless network, out of which any $(t+1)$ servers can jointly perform any arbitration or authorization successfully, but $t$servers cannot perform the same. Hence, up to t compromised servers can be tolerated. This is called an $(n, t + 1)$ configuration, where $n \geq 3t + 1$.

To sign a certificate, each server generates a partial signature using its private key and submits it to a combiner. The combiner can be any one of the servers. In order to ensure that the key is combined correctly, $t + 1$ combiners can be used to account for at most t malicious servers. Using $t + 1$ partial signatures (obtained from itself and t other servers), the combiner computes a signature and verifies its validity using a public key. If the verification fails, it means that at least

one of the $t + 1$ keys is not valid, so another subset of $t + 1$ partial signatures is tried. If the combiner itself is malicious, it cannot get a valid key, because the partial signature of itself is always invalid. The scheme can be applied to asynchronous networks, with no bound on message delivery or processing times. This is one of the strengths of the scheme, as the requirement of synchronization makes the system vulnerable to DoS attacks. An adversary can delay a node long enough to violate the synchrony assumption, thereby disrupting the system. Sharing a secret in a secure manner alone does not completely fortify a system. Mobile adversaries can move from one server to another, attack them, and get hold of their private keys. Over a period of time, an adversary can have more than t private keys. To counter this, *share refreshing* has been proposed, by which servers create a new independent set of shares (the partial signatures which are used by the servers) periodically. Hence, to break the system, an adversary has to attack and capture more than $t$servers within the period between two successive refreshes; otherwise, the earlier share information will no longer be valid. This improves protection against mobile adversaries.

Self-Organized Public Key Management for Mobile Ad Hoc Networks

The authors of [24] have proposed a completely self-organized public key system for ad hoc wireless networks. This makes use of absolutely no infrastructure – TTP, CA, or server – even during initial configuration. The users in the ad hoc wireless network issue certificates to each other based on personal acquaintance. A certificate is a binding between a node and its publickey. These certificates are also stored and distributed by the users themselves. Certificates are issued only for a specified period of time and contain their time of expiry along with them. Before it expires, the certificate is updated by the user who had issued the certificate. Initially, each user has a local repository consisting of the certificates issued by him and the certificates issued by other users to him. Hence, each certificate is initially stored twice, by the issuer and by the person for whom it is issued. Periodically, certificates from neighbors are requested and the repository is updated by adding any new certificates. If any of the certificates are conflicting (*e.g.*, the same public key to different users, or the same user having different public keys), it is possible that a malicious node has issued a false certificate. A node then labels such certificates as *conflicting* and tries to resolve the conflict. Various methods exist to compare the confidence in one certificate over another. For instance, another set of certificates obtained from another neighbor can be used to take a majority decision. This can be used to evaluate the trust in other

users and detect malicious nodes. If the certificates issued by some node are found to be wrong, then that node may be assumed to be malicious. The authors of [24] define a certificate graph as a graph whose vertices are public keys of some nodes and whose edges are public-key certificates issued by users. When a user *X* wants to obtain the public key of another user *Y*, he/she finds a chain of valid public key certificates leading to *Y*. The chain is such that the first hop uses an edge from *X*, that is, a certificate issued by *X*, the last hop leads into *Y*(this is a certificate issued to *Y*), and all intermediate nodes are trusted through the previous certificate in the path. The protocol assumes that trust is transitive, which may not always be valid. Having seen the various key management techniques employed in ad hoc wireless networks, we now move on to discuss some of the security-aware routing schemes for ad hoc wireless networks.

SECURE ROUTING IN AD HOC WIRELESS NETWORKS

Requirements of a Secure Routing Protocol for Ad Hoc Wireless Networks

The fundamental requisites of a secure routing protocol for ad hoc wireless networks are listed as follows:

• **Detection of malicious nodes:** A secure routing protocol should be able to detect the presence of malicious nodes in the network and should avoid the participation of such nodes in the routing process. Even if such malicious nodes participate in the route discovery process, the routing protocol should choose paths that do not include such nodes.

• **Guarantee of correct route discovery:** If a route between the source and the destination nodes exists, the routing protocol should be able to find the route, and should also ensure the correctness of the selected route.

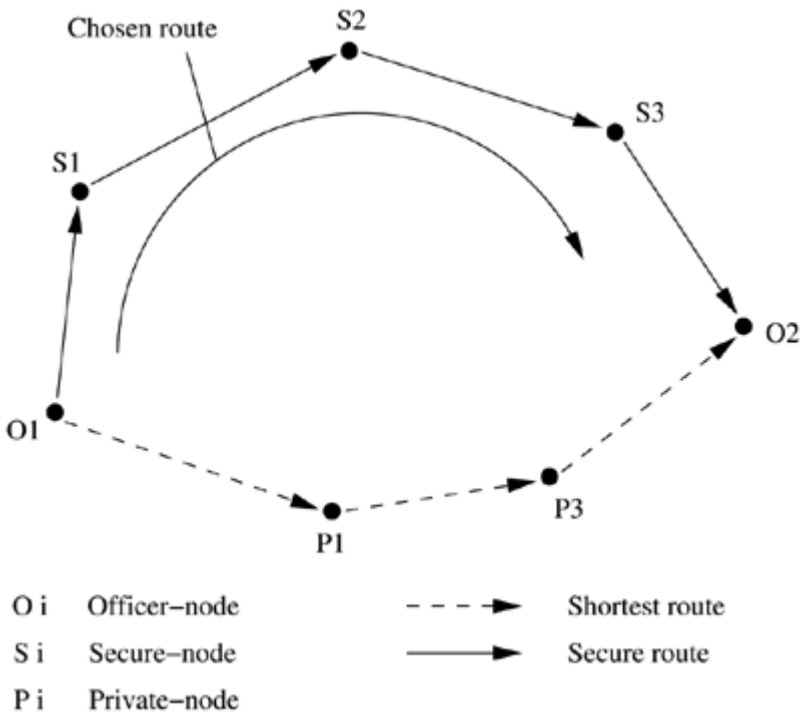
• **Confidentiality of network topology: An** information disclosure attack may lead to the discovery of the network topology by the malicious nodes. Once the network topology is known, the attacker may try to study the traffic pattern in the network. If some of the nodes are found to be more active compared to others, the attacker may try to mount (*e.g.,* DoS) attacks on such bottleneck nodes. This may ultimately affect the on-going routing process. Hence, the confidentiality of the network topology is an important requirement to be met by the secure routing protocols.

• **Stability against attacks:** The routing protocol must be self-stable in the sense that it must be able to revert to its normal operating state within a finite amount of time after a passive or an active attack. The routing protocol should take care that these attacks do not permanently disrupt the routing process. The protocol must also ensure Byzantine robustness, that is, the protocol should work properly even if some of the nodes, which were earlier participating in the routing process, turn out to become malicious at a later point of time or are intentionally damaged.

Security-Aware Ad Hoc Routing Protocol

The security-aware ad hoc routing (SAR) protocol [25] uses security as one of the key metrics in path finding. A framework for enforcing and measuring the attributes of the security metric has been provided in [25]. This framework also enables the use of different levels of security for different applications that use SAR for routing. In ad hoc wireless networks, communication between end nodes through possibly multiple intermediate nodes is based on the fact that the two end nodes trust the intermediate nodes. SAR defines *level of trust* as a metric for routing and as one of the attributes for security to be taken into consideration while routing. The routing protocol based on the level of trust is explained using Figure 9.14. As shown in Figure 9.14, two paths exist between the two officers *O*1 and*O*2 who want to communicate with each other. One of these paths is a shorter path which runs through private nodes whose trust levels are very low. Hence, the protocol chooses a longer but secure path which passes through other secure (officer) nodes.

Illustration of the level of trust metric



The SAR protocol can be explained using any one of the traditional routing protocols. This section explains SAR using the AODV protocol [18] discussed in detail in Chapter 7. In the AODV protocol, the source node broadcasts a *RouteRequest* packet to its neighbors. An intermediate node, on receiving a*RouteRequest* packet, forwards it further if it does not have a route to the destination. Otherwise, it initiates a *RouteReply* packet back to the source node using the reverse path traversed by the*RouteRequest* packet. In SAR, a certain level of security is incorporated into the packet-forwarding mechanism. Here, each packet is associated with a security level which is determined by a number calculation method (explained later in this section). Each intermediate node is also associated with a certain level of security. On receiving a packet, the intermediate node compares its level of security with that defined for the packet.

If the node's security level is less than that of the packet, the*RouteRequest* is simply discarded. If it is greater, the node is considered to be a secure node and is permitted to forward the packet in addition to being able to view the packet. If the security levels of the intermediate node and the received packet are found to be equal, then the intermediate node will not be able to view the packet (which can be ensured using a proper authentication mechanism); it just forwards the packet further.