Chapter 9

TRANSPORT LAYER AND SECURITY PROTOCOLS FOR AD HOC WIRELESS NETWORKS

9.1 INTRODUCTION

The objectives of a transport layer protocol include the setting up of an end-to-end 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 connection-oriented 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.

The previous chapters discussed various networking protocols for ad hoc wireless networks. However, almost all of them did not take into consideration one very important aspect of communication: security. Due to the unique characteristics of ad hoc wireless networks, which have been mentioned in the previous chapters, such networks are highly vulnerable to security attacks compared to wired networks or infrastructure-based wireless networks (such as cellular networks). Therefore, security protocols being used in the other networks (wired networks and infrastructure-based wireless networks) cannot be directly applied to ad hoc wireless networks. The second half of this chapter focuses on the security aspect of communication in ad hoc wireless networks. Some of the recently proposed protocols for achieving secure communication are discussed.

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

In this section, some of the 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 multi-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 MAC layers. 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.

9.3 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, scalable, and protocol-independent interaction with lower layers.
- The protocol should maintain end-to-end semantics.

9.4 CLASSIFICATION OF TRANSPORT LAYER SOLUTIONS

Figure 9.1 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.

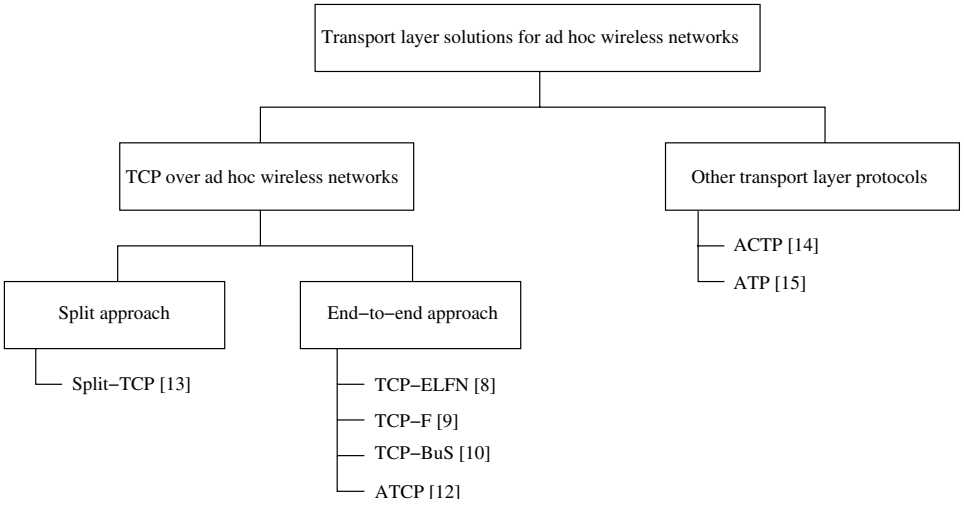

Figure 9.1. Classification of transport layer solutions.

9.5 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, byte-stream 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.

9.5.1 A Brief Revisit to Traditional TCP

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 include congestion 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 the 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 *slow-start threshold* (the slow-start threshold has an initial value of 64 KB). Figure 9.2 shows 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

[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 TCP packet is considered as a segment containing several bytes rather than a packet. However, segment and packet are used interchangeably in this chapter.

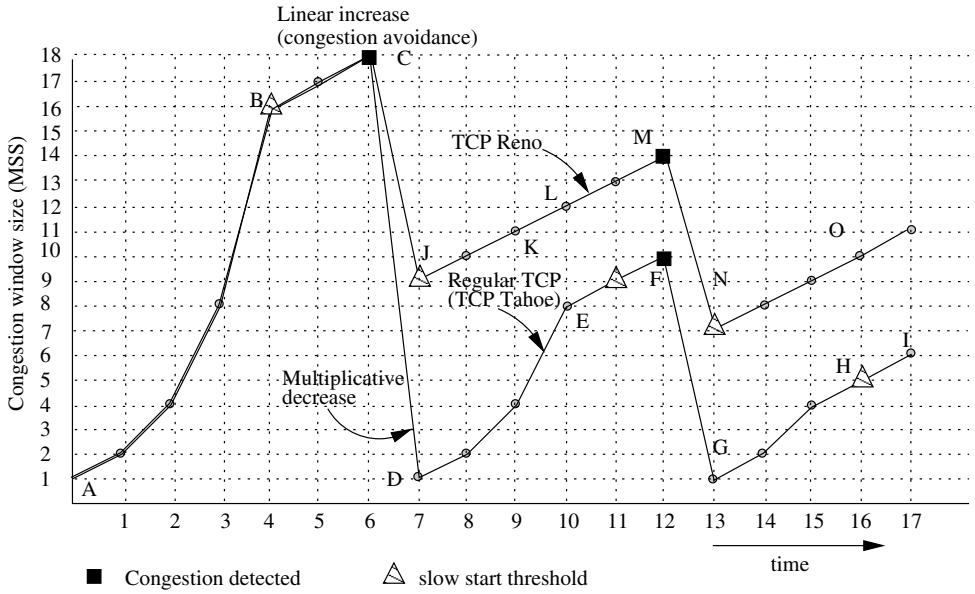

Figure 9.2. Illustration of TCP congestion window.

ACK packet. If the ACK packet does not arrive within the RTO period, 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.

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.

Among the several extensions of TCP, some of the important schemes are discussed below. The regular TCP which was discussed above is also called as TCP Tahoe [2] (in most of the existing literature). TCP Reno [3] is similar to TCP Tahoe with *fast recovery*. On timeout or arrival of three DUPACKs, the TCP Reno sender enters the fast recovery during which (refer to points C-J-K in Figure 9.2)

the TCP Reno sender retransmits the lost packet, reduces the slow-start threshold and congestion window size to half the size of the current congestion window, and increments the congestion window linearly (one MSS per DUPACK) with every subsequent DUPACK. On reception of a new ACK (not a DUPACK, *i.e.,* an ACK with a sequence number higher than the highest seen sequence number so far), the TCP Reno resets the congestion window with the slow-start threshold and enters the congestion avoidance phase similar to TCP Tahoe (points K-L-M in Figure 9.2).

J. C. Hoe proposed TCP-New Reno [4] extending the TCP Reno in which the TCP sender does not exit the fast-recovery state, when a new ACK is received. Instead it continues to remain in the fast-recovery state until all the packets originated are acknowledged. For every intermediate ACK packet, TCP-New Reno assumes the next packet after the last acknowledged one is lost and is retransmitted.

TCP with selective ACK (SACK) [5], [6] improves the performance of TCP by using the selective ACKs provided by the receiver. The receiver sends a SACK instead of an ACK, which contains a set of SACK blocks. These SACK blocks contain information about the recently received packets which is used by the TCP sender while retransmitting the lost packets.

9.5.2 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 (Chapter 7 discusses network layer routing protocols in detail). 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 in Figure 9.3. The possibility of a path break increases with path length. Given that the probability of a link break is p_l, the probability of a path break (p_b) for a path of length k can be obtained as $p_b = 1 - (1 - p_l)^k$. Figure 9.4 shows the variation of p_b with path length for $p_l = 0.1$. Hence as the path length increases, the probability of a path break increases, resulting in the degradation of the throughput in the network.

- **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

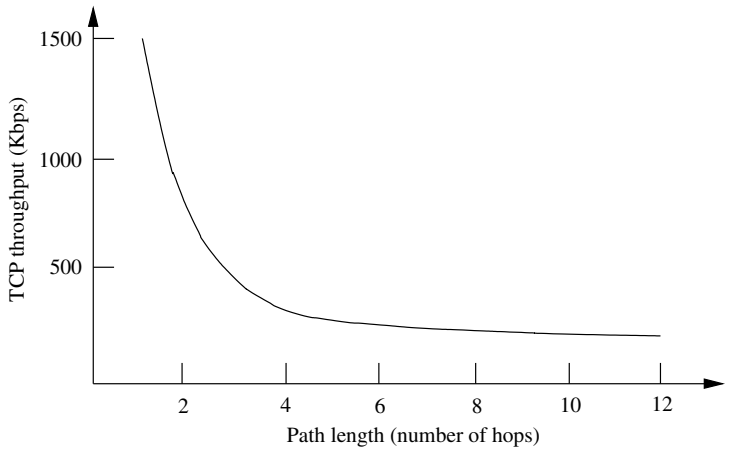

Figure 9.3. Variation of TCP throughput with path length.

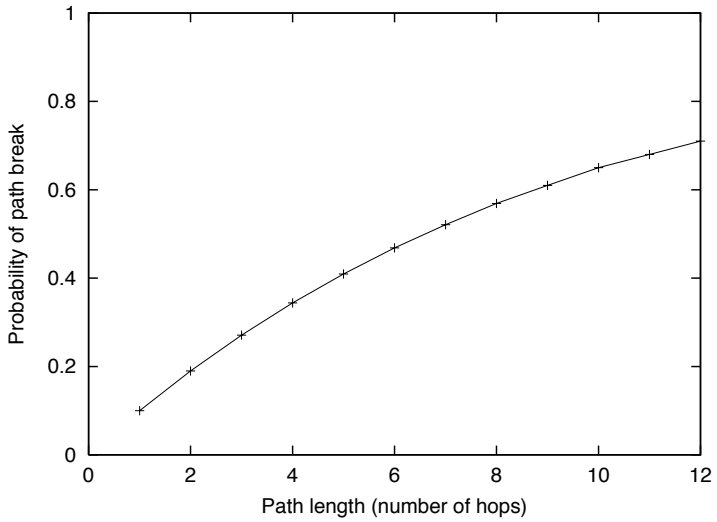

Figure 9.4. Variation of p_b with path length ($p_l = 0.1$).

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 TCP session 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 TCP session B experiences a path break. These partitions could merge back into a single network at time t3 (refer to Figure 9.5 (c)).

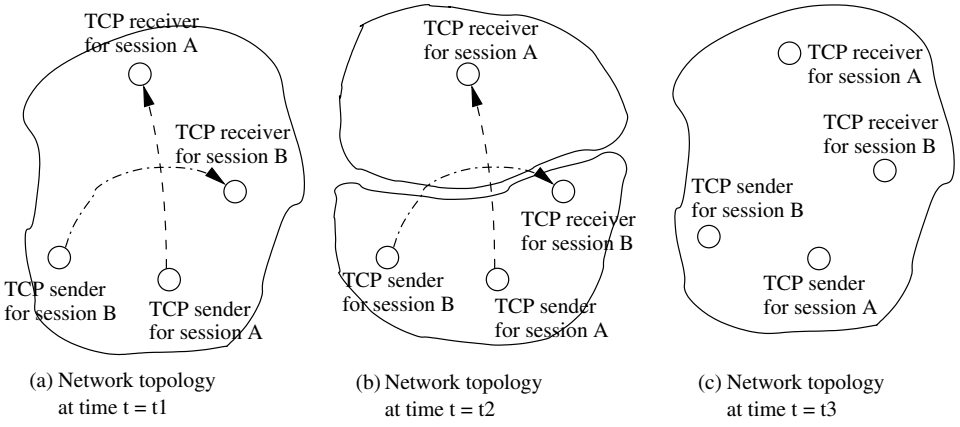

Figure 9.5. Effect of partitioning and merging of network.

- **The use of sliding-window-based transmission**: TCP uses a sliding window for flow control. The transmission of packets is decided by the size of the window, and when the ACKs arrive from a destination, further packets are transmitted. This avoids the use of individual fine-grained timers for transmission of each TCP flow. Such a design is preferred in order to improve scalability of the protocol in high-bandwidth networks such as the Internet where millions of TCP connections may be established with some heavily loaded servers. The use of a sliding window can also contribute to degraded performance in bandwidth-constrained ad hoc wireless networks where the MAC layer protocol may not exhibit short-term and long-term fairness. For example, the popular MAC protocols such as CSMA/CA protocol show short-term unfairness, where a node that has captured the channel has a higher probability of capturing the channel again. This unfairness can lead to a number of TCP ACK packets being delivered to the TCP sender in succession, leading to a burstiness in traffic due to the subsequent transmission of TCP segments.

The enhancements to TCP that improve the performance of TCP in ad hoc wireless networks are discussed in the following sections.

9.5.3 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. This was discussed earlier in this chapter.

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 RFN packet 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 TCP state is updated back to the connected state [Figure 9.6 (c)].

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.

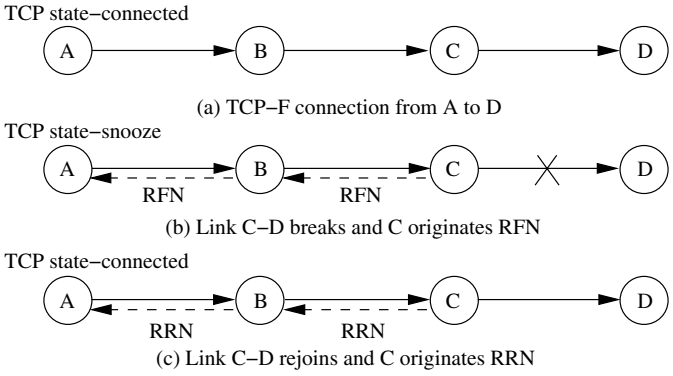

Figure 9.6. Operation of TCP-F.

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 RFN packet. 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.

9.5.4 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 ICMP[2] destination unreachable (DUR) message to the sender, or (ii) by piggy-backing this information on the *RouteError*[3] message that is sent to the sender.

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

[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.

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.

9.5.5 TCP-BuS

TCP with buffering capability and sequence information (TCP-BuS) [10] is similar to the TCP-F and TCP-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 of ABR 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 TCP-BuS 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 round-trip 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 the TCP-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 TCP-BuS 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 successful LQ-REPLY 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.

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

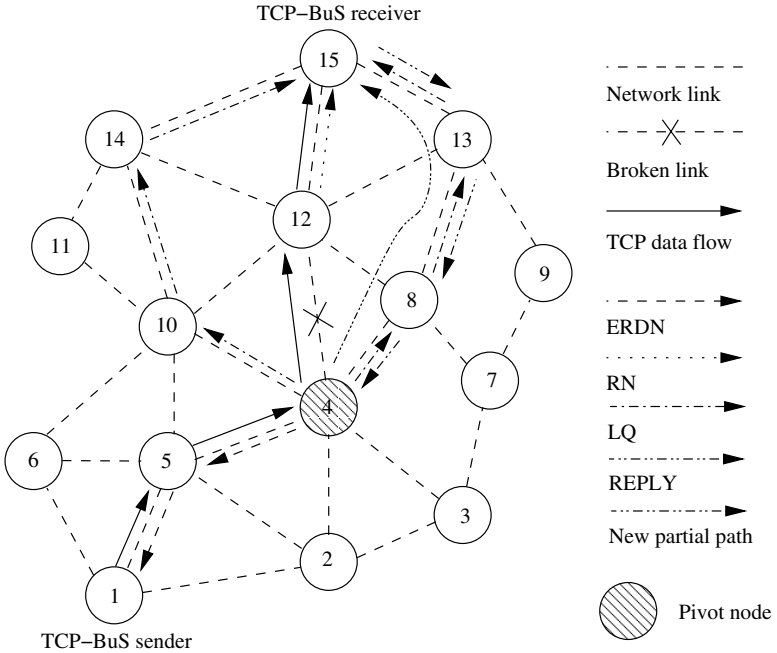

Figure 9.7. Operation of TCP-BuS.

acknowledgment. TCP-BuS also takes advantage of the underlying routing protocols, especially the on-demand routing protocols such as ABR. 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.

9.5.6 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 TCP packets are propagated. Based on the feedback information received from the intermediate nodes, the TCP 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 the TCP 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 TCP protocols. The ATCP layer essentially makes use of the explicit congestion notification (ECN) for maintenance of the states.

Figure 9.8 (a) shows the thin layer implementation of ATCP between the traditional TCP layer and the IP layer. This does not require changes in the existing TCP protocol. This layer is active only at the TCP sender. The major function of the ATCP layer is to monitor the packets sent and received by the TCP sender, 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.

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 TCP buffer. When a new ACK comes from the TCP receiver,

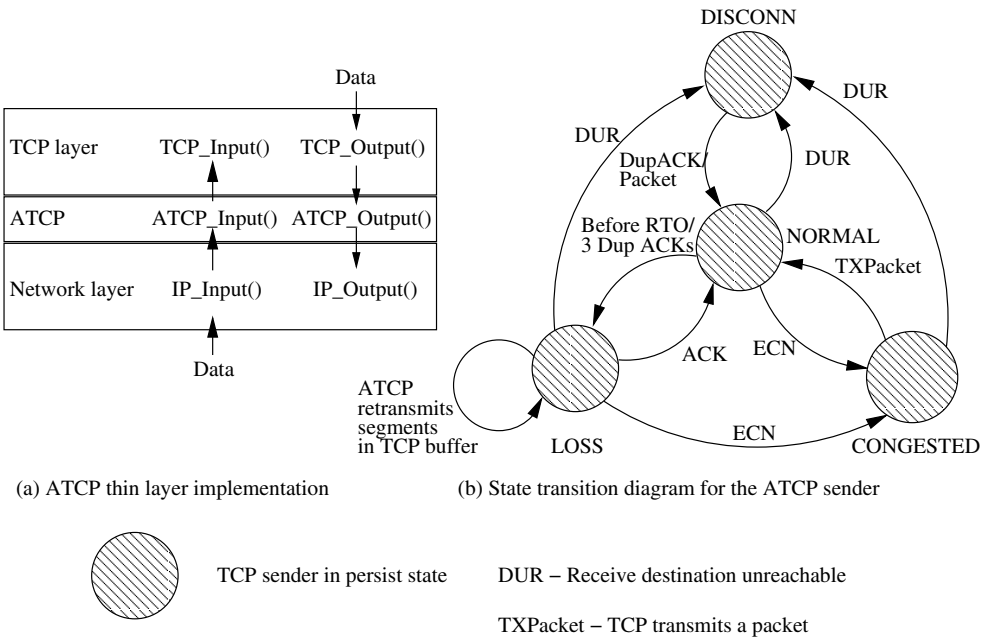

Figure 9.8. An illustration of ATCP thin layer and ATCP state diagram.

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 ECN[4] 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 ICMP destination unreachable (DUR) message. Upon reception of the DUR message, ATCP puts the TCP sender 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, ATCP changes 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 ICMP DUR 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

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

Table 9.1. The actions taken by ATCP

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

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, ATCP tries to perform the activities listed in Table 9.1.

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.

9.5.7 Split TCP

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 MAC protocols 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 6 for 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 TCP connection 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-TCP connection exists between source node 1 and destination node 15. A proxy node receives the TCP packets, 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 buffered packets only after receiving the end-to-end acknowledgment for those packets.

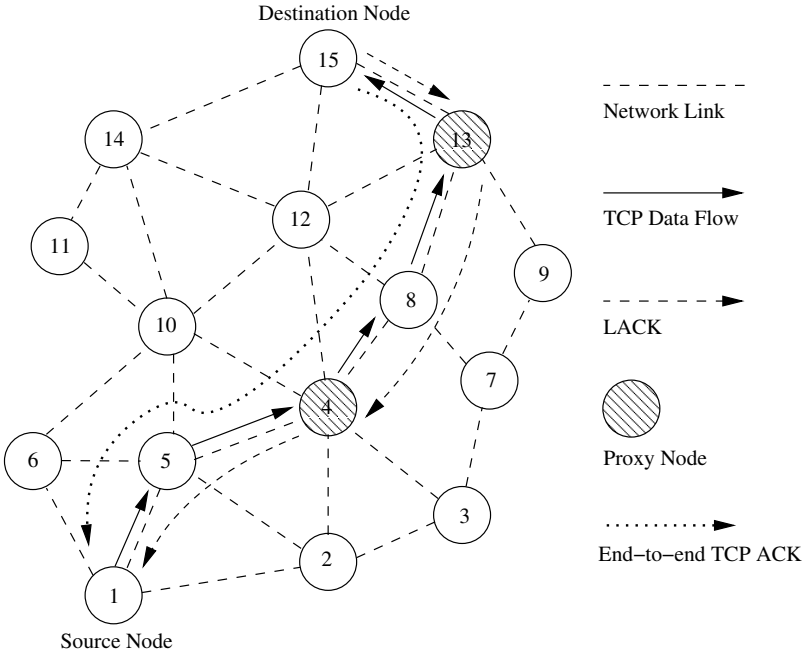

Figure 9.9. An illustration of Split-TCP.

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 TCP packets 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 LACK packet, 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.

9.5.8 A Comparison of TCP Solutions for Ad Hoc Wireless Networks

Table 9.2 compares how various issues are handled in the TCP extensions discussed so far in this chapter.

9.6 OTHER TRANSPORT LAYER PROTOCOLS FOR AD HOC WIRELESS NETWORKS

The performance of a transport layer protocol can be enhanced if it takes into account the nature of the network environment in which it is applied. Especially in wireless environments, it is important to consider the properties of the physical layer and the interaction of the transport layer with the lower layers. This section discusses some of the transport layer protocols that were designed specifically for ad hoc wireless networks. Even though interworking with TCP is very important, there exist several application scenarios such as military communication where a radically new transport layer protocol can be used.

9.6.1 Application Controlled Transport Protocol

Unlike the TCP solutions discussed earlier in this chapter, application controlled transport protocol (ACTP[5]) [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.

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 9.10 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

[5] Originally called ATP, for differentiating with ad hoc transport protocol it is referred to as ACTP in this chapter.

Table 9.2. 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

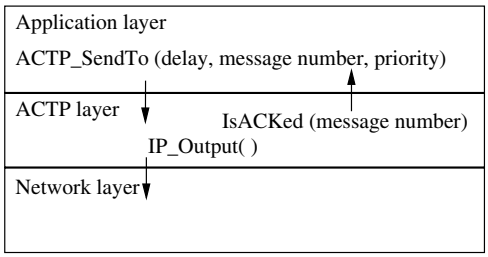

Figure 9.10. An illustration of the interface functions used in ACTP.

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.

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.

9.6.2 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 averaged[6] queuing delay (D_Q) and contention delay (D_C) 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 $\beta \times (D_Q + D_C)$, 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.

[6]Originally called "exponentially averaged," renamed here with a more appropriate term, "weighted average." An example for this is $\bar{Q}_{delay} = \alpha \times Q_{delay_{new}} + (1 - \alpha) \times Q_{delay_{old}}$, where α is an appropriate weight factor and the other terms are self-explanatory.

Unlike TCP, which employs either a decrease of the congestion window or an increase of the congestion window after a congestion, ATP has three phases, namely, increase, decrease, and maintain. If the new transmission rate (R) fed back from the network is beyond a threshold (γ) greater than the current transmission rate (S) [*i.e.,* $R > S(1+\gamma)$], then the current transmission rate is increased by a fraction (k) of the difference between the two transmission rates (*i.e.,* $S = S + \frac{R-S}{k}$). The fraction and threshold are taken to avoid rapid fluctuations in the transmission rate and induced load. The current transmission rate is updated to the new transmission rate if the new transmission rate is lower than the current transmission rate. In the maintain phase, if the new transmission rate is higher than the current transmission rate, but less than the above mentioned threshold, then the current transmission rate is maintained without any change.

If an ATP sender has not received any ACK packets for two consecutive feedback periods, it undergoes a multiplicative decrease of the transmission rate. After a third such period without any ACK, the connection is assumed to be lost and the ATP sender goes to the connection initiation phase during which it periodically generates probe packets. When a path break occurs, the network layer detects it and originates an ELFN packet toward the ATP sender. The ATP sender freezes the sender state and goes to the connection initiation phase. In this phase also, the ATP sender periodically originates probe packets to know the status of the path. With a successful probe, the sender begins data transmission again.

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 ATP sender may become a scalability bottleneck in resource-constrained mobile nodes.

9.7 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.

9.8 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.

9.9 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.

9.10 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

[7] A firewall is used to separate a local network from the outside world. It is a software which works closely with a router program and filters all packets entering the network to determine whether or not to forward those packets toward their intended destinations. A firewall protects the resources of a private network from malicious intruders on foreign networks such as the Internet. In an ad hoc wireless network, the firewall software could be installed on each node on the network.

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.

Figure 9.11 shows a classification of the different types of attacks possible in ad hoc wireless networks. The following sections describe the various attacks listed in the figure.

9.10.1 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

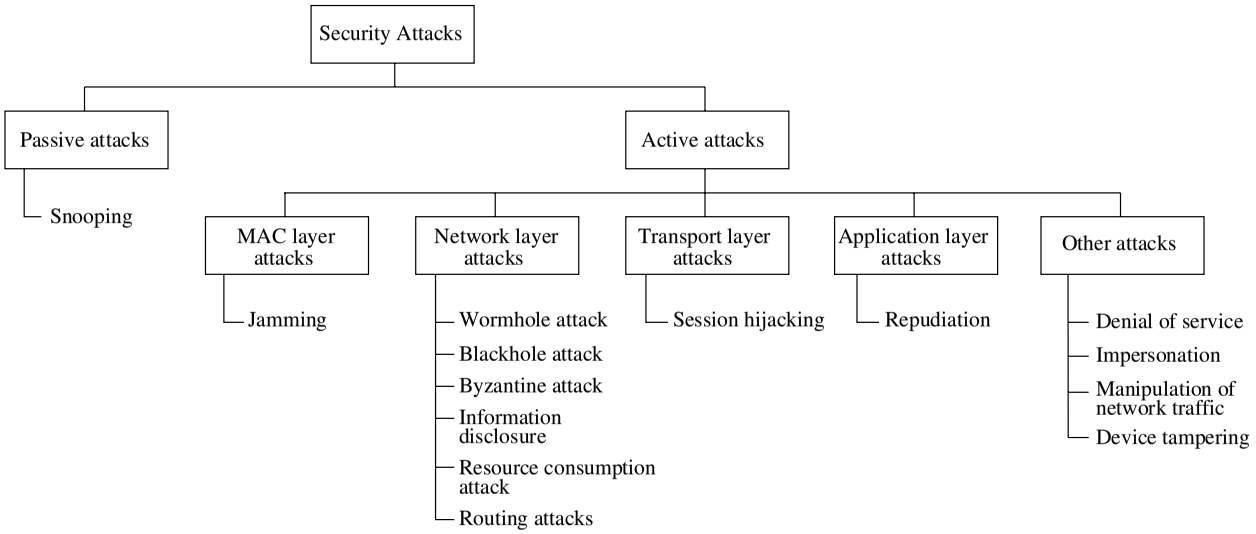

Figure 9.11. Classifications of attacks.

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.
  - **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.
  - **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.
  - **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.
  - **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.
  - **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.

9.10.2 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.

9.10.3 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.

9.10.4 Other Attacks

This section discusses security attacks that cannot strictly be associated with any specific layer in the network protocol stack.

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.
- **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

[8]SYN packets are used to establish an end-to-end session between two nodes at the transport layer.

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.

9.11 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. A detailed study of cryptography is presented in [21].

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 non-scalable. 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 the keys 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.

9.11.1 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 9.12 (a) illustrates the encryption using

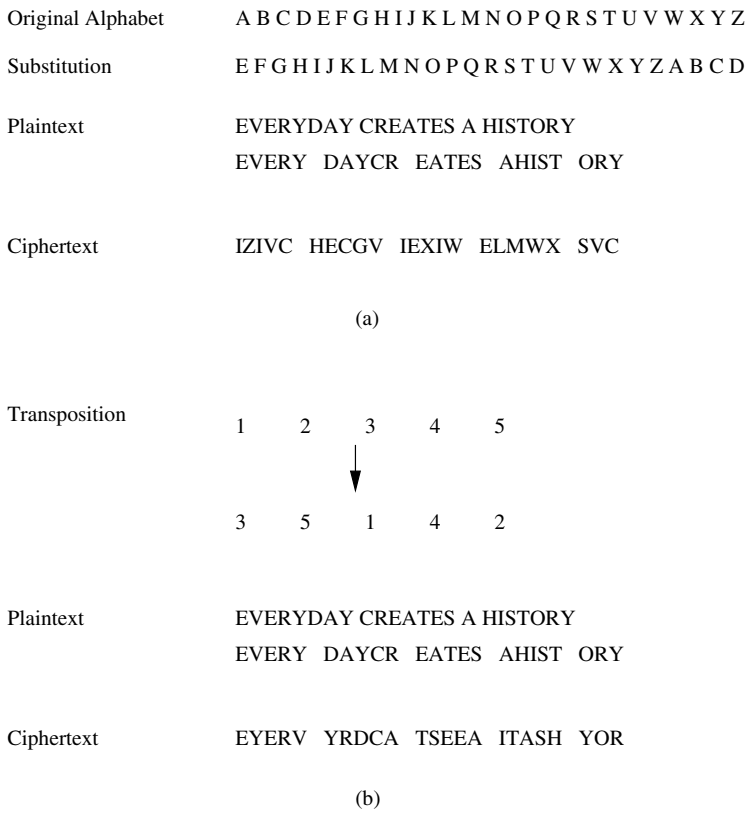

Figure 9.12. Substitution and transposition.

substitution, and Figure 9.12 (b) shows a transposition cipher. The block length used is five.

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.

9.11.2 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 [21] 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.

9.11.3 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 [22].

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 within 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 k_x, using which it encrypts K with f, and sends to node Y. Node Y encrypts this with a random key k_y using g, and sends it back to node X. Now, node X decrypts this message with its key k_x, and after applying the inverse function f^{-1}, sends it to node Y. Finally, node Y decrypts the message using k_y and g^{-1} to obtain the key K. The message exchanges of the protocol are illustrated in Figure 9.13.

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 the AP 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.

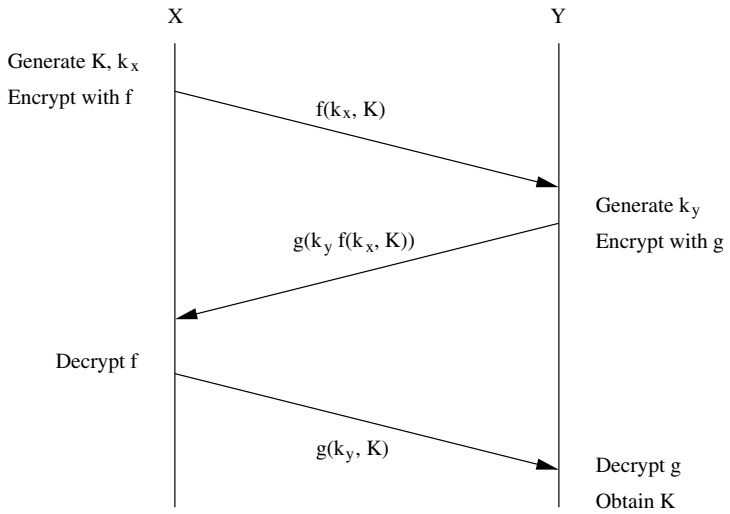

Figure 9.13. Shamir's three-pass protocol.

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.

9.11.4 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 public key. 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.

9.12 SECURE ROUTING IN AD HOC WIRELESS NETWORKS

Unlike the traditional wired Internet, where dedicated routers controlled by the Internet service providers (ISPs) exist, in ad hoc wireless networks, nodes act both as regular terminals (source or destination) and also as routers for other nodes. In the absence of dedicated routers, providing security becomes a challenging task in these networks. Various other factors which make the task of ensuring secure communication in ad hoc wireless networks difficult include the mobility of nodes, a promiscuous mode of operation, limited processing power, and limited availability of resources such as battery power, bandwidth, and memory. Section 9.10.1 has pointed out some of the possible security attacks at the network layer. In the following sections, we show how some of the well-known traditional routing protocols for ad hoc networks fail to provide security. Some of the mechanisms proposed for secure routing are also discussed.

9.12.1 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:** As explained in Section 9.10.1, 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.

In the following sections, some of the security-aware routing protocols proposed for ad hoc wireless networks are discussed.

9.12.2 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 $O1$ and $O2$ 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.

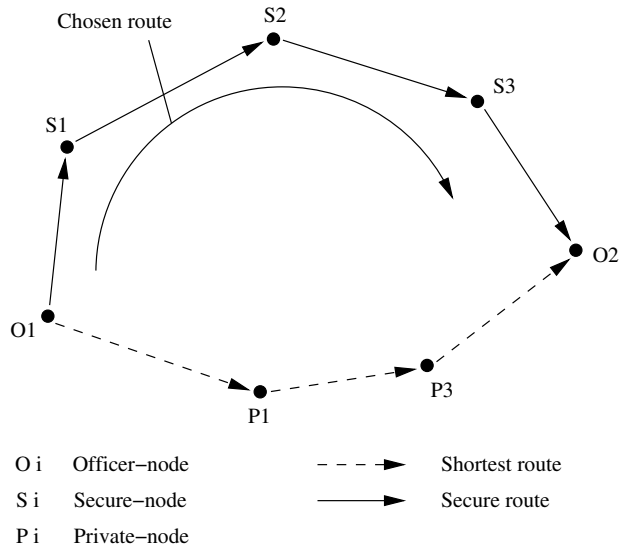

Figure 9.14. Illustration of the level of trust metric.

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.

Nodes of equal levels of trust distribute a common key among themselves and with those nodes having higher levels of trust. Hence, a hierarchical level of security could be maintained. This ensures that an encrypted packet can be decrypted (using the common key) only by nodes of the same or higher levels of security compared to the level of security of the packet. Different levels of trust can be defined using a number calculated based on the level of security required. It can be calculated using many methods. Since timeliness, in-order delivery of packets, authenticity, authorization, integrity, confidentiality, and non-repudiation are some of the desired characteristics of a routing protocol, a suitable number can be defined for the trust level for nodes and packets based on the number of such characteristics taken into account.

The SAR mechanism can be easily incorporated into the traditional routing protocols for ad hoc wireless networks. It could be incorporated into both on-demand and table-driven routing protocols. The SAR protocol allows the application to choose the level of security it requires. But the protocol requires different keys for different levels of security. This tends to increase the number of keys required when the number of security levels used increases.

9.12.3 Secure Efficient Ad Hoc Distance Vector Routing Protocol

Secure efficient ad hoc distance vector (SEAD) routing protocol [26], is a secure ad hoc routing protocol based on the destination-sequenced distance vector (DSDV) routing protocol [27] discussed in Chapter 7. This protocol is mainly designed to overcome security attacks such as DoS and resource consumption attacks. The operation of the routing protocol does not get affected even in the presence of multiple uncoordinated attackers corrupting the routing tables. The protocol uses a one-way hash function and does not involve any asymmetric cryptographic operation.

Distance Vector Routing

Distance vector routing protocols belong to the category of table-driven routing protocols. Each node maintains a routing table containing the list of all known routes to various destination nodes in the network. The metric used for routing is the distance measured in terms of hop-count. The routing table is updated periodically by exchanging routing information. An alternative to this approach is *triggered updates*, in which each node broadcasts routing updates only if its routing table gets altered. The DSDV protocol for ad hoc wireless networks uses *sequence number* tags to prevent the formation of loops, to counter the count-to-infinity problem, and for faster convergence. When a new route update packet is received for a destination, the node updates the corresponding entry in its routing table only if the sequence number on the received update is greater than that recorded with the corresponding entry in the routing table. If the received sequence number and the previously recorded sequence number are both equal, but if the routing update has a new value for the routing metric (distance in number of hops), then in this case also the update is effected. Otherwise, the received update packet is discarded. DSDV uses triggered updates (for important routing changes) in addition to the regular periodic updates. A slight variation of DSDV protocol known as DSDV-SQ (DSDV for sequence numbers) initiates triggered updates on receiving a new sequence number update.

One-Way Hash Function

SEAD uses authentication to differentiate between updates that are received from non-malicious nodes and malicious nodes. This minimizes resource consumption attacks caused by malicious nodes. SEAD uses a one-way hash function for authenticating the updates. A one-way hash function (H) generates a one-way hash chain $(h_1, h_2, \ldots)$. The function H maps an input bit-string of any length to a fixed length bit-string, that is, $H : (0,1)^* \rightarrow (0,1)^\rho$, where ρ is the length in bits of the output bit-string. To create a one-way hash chain, a node generates a random number with initial value $x \in (0,1)^\rho$. h_0, the first number in the hash chain is initialized to x. The remaining values in the chain are computed using the general formula, $h_i = H(h_{i-1})$ for $0 \leq i \leq n$, for some n. Now we shall see how the one-way hash function incorporates security into the existing DSDV-SQ routing protocol. The SEAD protocol assumes an upper bound on the metric used. For example, if the metric used is distance, then the upper bound value $m-1$ defines the maximum diameter (maximum of lengths of all the routes between a pair of nodes) of the ad hoc wireless network. Hence, the routing protocol ensures that no route of length greater than m hops exists between any two nodes.

If the sequence of values calculated by a node using the hash function H is given by $(h_1, h_2, \ldots, h_n)$, where n is divisible by m, then for a routing table entry with sequence number i, let $k = \frac{k}{m} - i$. If the metric j (distance) used for that routing table entry is $0 \leq j \leq m-1$, then the value h_{km+j} is used to authenticate the routing update entry for that sequence number i and that metric j. Whenever a route update message is sent, the node appends the value used for authentication

along with it. If the authentication value used is h_{km+j}, then the attacker who tries to modify this value can do so only if he/she knows h_{km+j-1}. Since it is a one-way hash chain, calculating h_{km+j-1} becomes impossible. An intermediate node, on receiving this authenticated update, calculates the new hash value based on the earlier updates (h_{km+j-1}), the value of the metric, and the sequence number. If the calculated value matches with the one present in the route update message, then the update is effected; otherwise, the received update is just discarded.

SEAD avoids routing loops unless the loop contains more than one attacker. This protocol could be implemented easily with slight modifications to the existing distance vector routing protocols. The protocol is robust against multiple uncoordinated attacks. The SEAD protocol, however, would not be able to overcome attacks where the attacker uses the same metric and sequence number which were used by the recent update message, and sends a new routing update.

9.12.4 Authenticated Routing for Ad Hoc Networks

Authenticated routing for ad hoc networks (ARAN) routing protocol [28], based on cryptographic certificates, is a secure routing protocol which successfully defeats all identified attacks in the network layer. It takes care of authentication, message integrity, and non-repudiation, but expects a small amount of prior security coordination among nodes. In [28], vulnerabilities and attacks specific to AODV and DSR protocols are discussed and the two protocols are compared with the ARAN protocol.

During the route discovery process of ARAN, the source node broadcasts *RouteRequest* packets. The destination node, on receiving the *RouteRequest* packets, responds by unicasting back a reply packet on the selected path. The ARAN protocol uses a preliminary cryptographic certification process, followed by an end-to-end route authentication process, which ensures secure route establishment.

Issue of Certificates

This section discusses the certification process in which the certificates are issued to the nodes in the ad hoc wireless network. There exists an authenticated trusted server whose public key is known to all legal nodes in the network. The ARAN protocol assumes that keys are generated *a priori* by the server and distributed to all nodes in the network. The protocol does not specify any specific key distribution algorithm. On joining the network, each node receives a certificate from the trusted server. The certificate received by a node A from the trusted server T looks like the following:

$$T \rightarrow A: \ \ cert_A \ = \ [IP_A, K_{A+}, t, e]K_{T-} \tag{9.12.1}$$

Here, IP_A, K_{A+}, t, e, and K_{T-} represent the IP address of node A, the public key of node A, the time of creation of the certificate, the time of expiry of the certificate, and the private key of the server, respectively.

End-to-End Route Authentication

The main goal of this end-to-end route authentication process is to ensure that the correct intended destination is reached by the packets sent from the source node. The source node S broadcasts a *RouteRequest*/*RouteDiscovery* packet destined to the destination node D. The *RouteRequest* packet contains the packet identifier [route discovery process (RDP)], the IP address of the destination (IP_D), the certificate of the source node S ($Cert_S$), the current time (t), and nonce N_S. The process can be denoted as below. Here, K_{S-} is the private key of the source node S.

$$S \rightarrow broadcasts :=\ [RDP, IP_D, Cert_S, N_S, t]K_{S-} \tag{9.12.2}$$

Whenever the source sends a route discovery message, it increments the value of nonce. Nonce is a counter used in conjunction with the time-stamp in order to make the nonce recycling easier. When a node receives an RDP packet from the source with a higher value of the source's nonce than that in the previously received RDP packets from the same source node, it makes a record of the neighbor from which it received the packet, encrypts the packet further with its own certificate, and broadcasts it further. The process can be denoted as follows:

$$A \rightarrow broadcasts :=\ [[RDP, IP_D, Cert_S, N_S, t]K_{S-}]K_{A-}, Cert_A \tag{9.12.3}$$

An intermediate node B, on receiving an RDP packet from a node A, removes its neighbor's certificate, inserts its own certificate, and broadcasts the packet further. The destination node, on receiving an RDP packet, verifies node S's certificate and the tuple (N_S, t) and then replies with the *RouteReply* packet (REP). The destination unicasts the REP packet to the source node along the reverse path as follows:

$$D \rightarrow X :=\ [REP, IP_S, Cert_D, N_S, t]K_{D-} \tag{9.12.4}$$

where node X is the neighbor of the destination node D, which had originally forwarded the RDP packet to node D. The REP packet follows the same procedure on the reverse path as that followed by the route discovery packet. An error message is generated if the time-stamp or nonce do not match the requirements or if the certificate fails. The error message looks similar to the other packets except that the packet identifier is replaced by the ERR message.

Table 9.3 shows a comparison between the AODV, DSR, and ARAN protocols with respect to their security-related features. ARAN remains robust in the presence of attacks such as unauthorized participation, spoofed route signaling, fabricated routing messages, alteration of routing messages, securing shortest paths, and replay attacks.

9.12.5 Security-Aware AODV Protocol

This section discusses security solutions that address a particular security flaw in the AODV routing protocol [18]. AODV is an on-demand routing protocol where

Table 9.3. Comparison of vulnerabilities of ARAN with DSR and AODV protocols

Attacks	Protocols		
	AODV	**DSR**	**ARAN**
Modifications required during remote redirection	Sequence number and hop-counts	Source routes	None
Tunneling during remote redirection	Yes	Yes	Yes
Spoofing	Yes	Yes	No
Cache poisoning	No	Yes	No

the route discovery process is initiated by sending *RouteRequest* packets only when data packets arrive at a node for transmission. A malicious intermediate node could advertise that it has the shortest path to the destination, thereby redirecting all the packets through itself. This is known as a blackhole attack, as explained in Section 9.10.1. The blackhole attack is illustrated in Figure 9.15. Let node M be the malicious node that enters the network. It advertises that it has the shortest path to the destination node D when it receives the *RouteRequest* packet sent by node S. The attacker may not be able to succeed if node A, which also receives the *RouteRequest* packet from node S, replies earlier than node M. But a major advantage for the malicious node is that it does not have to search its routing table

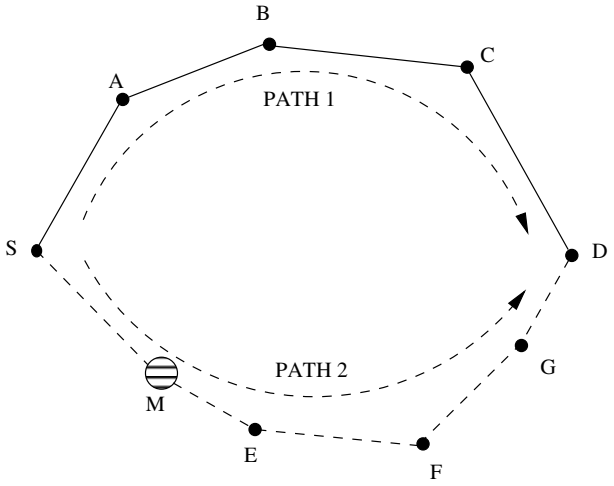

Figure 9.15. Illustration of blackhole problem.

for a route to the destination. Also, the *RouteReply* packets originate directly from the malicious node and not from the destination node. Hence, the malicious node would be able to reply faster than node A, which would have to search its routing table for a route to the destination node. Thus, node S may tend to establish a route to destination D through the malicious node M, allowing node M to listen to all packets meant for the destination node.

Solutions for the Blackhole Problem

One of the solutions for the blackhole problem is to restrict the intermediate nodes from originating *RouteReply* packets. Only the destination node would be permitted to initiate *RouteReply* packets. Security is still not completely assured, since the malicious node may lie in the path chosen by the destination node. Also, the delay involved in the route discovery process increases as the size of the network increases. In another solution to this problem, suggested in [29], as soon as the *RouteReply* packet is received from one of the intermediate nodes, another *RouteRequest* packet is sent from the source node to the neighbor node of the intermediate node in the path. This is to ensure that such a path exists from the intermediate node to the destination node. For example, let the source node send *RouteRequest* packets and receive *RouteReply* through the intermediate malicious node M. The *RouteReply* packet of node M contains information regarding its next-hop neighbor nodes. Let it contain information about the neighbor node E. Then, as shown in Figure 9.16, the source node S sends *FurtherRouteRequest* packets to this neighbor node E. Node E responds by sending a *FurtherRouteReply* packet to source node S. Since

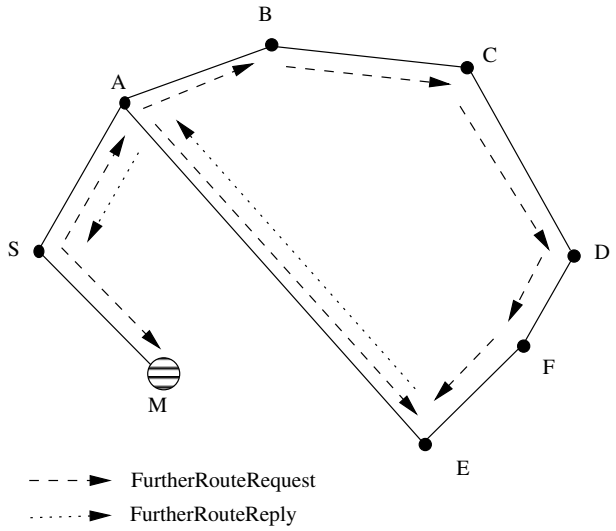

Figure 9.16. Propagation of *FurtherRouteRequest* and *FurtherRouteReply.*

node M is a malicious node which is not present in the routing list of node E, the *FurtherRouteReply* packet sent by node E will not contain a route to the malicious node M. But if it contains a route to the destination node D, then the new route to the destination through node E is selected, and the earlier selected route through node M is rejected. This protocol completely eliminates the blackhole attack caused by a single attacker. The major disadvantage of this scheme is that the control overhead of the routing protocol increases considerably. Also, if the malicious nodes work in a group, this protocol fails miserably.

9.13 SUMMARY

This chapter discussed the major challenges that a transport layer protocol faces in ad hoc wireless networks. The major design goals of a transport layer protocol were listed and a classification of existing transport layer solutions was provided. TCP is the most widely used transport layer protocol and is considered to be the backbone of today's Internet. It provides end-to-end, reliable, byte-streamed, in-order delivery of packets to nodes. Since TCP was designed to handle problems present in traditional wired networks, many of the issues that are present in dynamic topology networks such as ad hoc wireless networks are not addressed. This causes reduction of throughput when TCP is used in ad hoc wireless networks. It is very important to employ TCP in ad hoc wireless networks as it is important to seamlessly communicate with the Internet whenever and wherever it is available. This chapter provided a discussion on the major reasons for the degraded performance of traditional TCP in ad hoc wireless networks and explained a number of recently proposed solutions to improve TCP's performance. Other non-TCP solutions were also discussed in detail.

The second half of this chapter dealt with the security aspect of communication in ad hoc wireless networks. The issues and challenges involved in provisioning security in ad hoc wireless networks were identified. This was followed by a layer-wise classification of the various types of attacks. Detailed discussions on key management techniques and secure routing techniques for ad hoc wireless networks were provided. Table 9.4 lists out the various attacks possible in ad hoc wireless networks along with the solutions proposed for countering those attacks.

Table 9.4. Defense against attacks

Attack	Targeted Layer in the Protocol Stack	Proposed Solutions
Jamming	Physical and MAC layers	FHSS, DSSS
Wormhole attack	Network layer	Packet Leashes [16]
Blackhole attack	Network layer	[25], [29]
Byzantine attack	Network layer	[17]
Resource consumption attack	Network layer	SEAD [26]
Information disclosure	Network layer	SMT [30]
Location disclosure	Network layer	SRP [30], NDM [31]
Routing attacks	Network layer	[19], SEAD [26], ARAN [28], ARIADNE [32]
Repudiation	Application layer	ARAN [28]
Denial of Service	Multi-layer	SEAD [26], ARIADNE [32]
Impersonation	Multi-layer	ARAN [28]

9.14 PROBLEMS

1. Assume that when the current size of the congestion window is 48 KB, the TCP sender experiences a timeout. What will be the congestion window size if the next three transmission bursts are successful? Assume that MSS is 1 KB. Consider (a) TCP Tahoe and (b) TCP Reno.

2. Find out the probability of a path break for an eight-hop path, given that the probability of a link break is 0.2.

3. Discuss the effects of multiple breaks on a single path at the TCP-F sender.

4. What additional state information is to be maintained at the FP in TCP-F?

5. Mention one advantage and one disadvantage of using probe packets for detection of a new path.

6. Mention one advantage and one disadvantage of using LQ and REPLY for finding partial paths in TCP-BuS.

7. What is the impact of the failure of proxy nodes in split-TCP?

8. During a research discussion, one of your colleagues suggested an extension of split-TCP where every intermediate node acts as a proxy node. What do you think would be the implications of such a protocol?

9. What are the pros and cons of assigning the responsibility of end-to-end reliability to the application layer?

10. What is the default value of β used for handling induced traffic in ATP and why is such a value chosen?

11. Explain how network security requirements vary in the following application scenarios of ad hoc wireless networks:

    (a) Home networks

    (b) Classroom networks

    (c) Emergency search-and-rescue networks

    (d) Military networks

12. Explain how security provisioning in ad hoc wireless networks differs from that in infrastructure-based networks?

13. Explain the key encrypting key (KEK) method.

14. Nodes A and B want to establish a secure communication, and node A generates a random key 11001001. Suppose the function used by both nodes A and B for encryption is XOR, and let node A generate a random transport key 10010101, and let node B generate 00101011. Explain the three-pass Shamir protocol exchanges.

15. Why is it not advisable to use natural-language passwords directly for cryptographic algorithms?

16. Consider the certificate graph shown in Figure 9.17, with the local certificate repositories of nodes A and B as indicated. Find the possible paths of trust from node A to node B which can be obtained using a chain of keys.

17. List a few inherent security flaws present in the following types of routing protocols: (a) table-driven and (b) on-demand routing.

18. List and explain how some of the inherent properties of the wireless ad hoc networks introduce difficulties while implementing security in routing protocols.

19. Mark the paths chosen by the following secure-routing protocols for the network topology shown in Figure 9.18: (a) Shortest path routing and (b) SAR protocol. Assume that node 2 is a secure node. (c) If node 2 (which lies in the path chosen by SAR protocol) is suddenly attacked and becomes a malicious node, then mark an alternative path chosen by SAODV protocol.

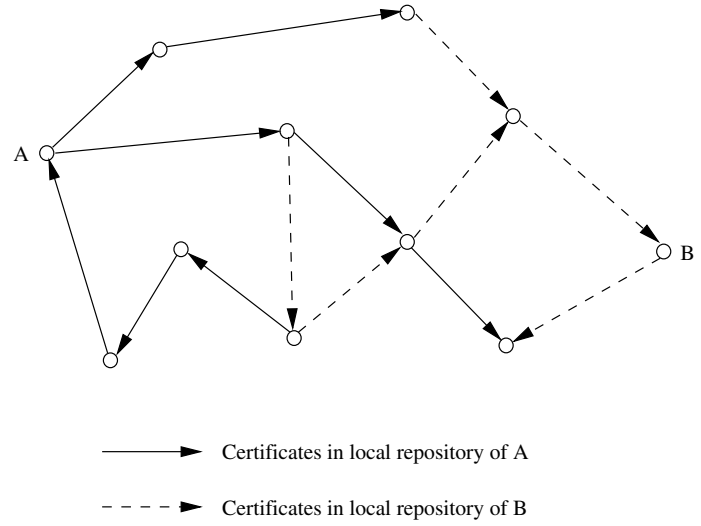

Figure 9.17. Certificate graph.

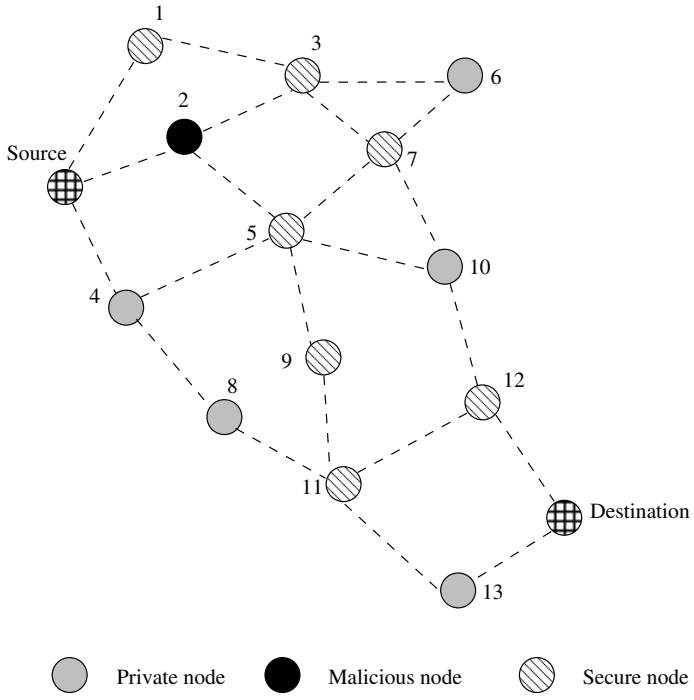

Figure 9.18. Example network topology.

BIBLIOGRAPHY

[1] J. Postel, "Transmission Control Protocol," *IETF RFC 793*, September 1981.

[2] V. Jacobson, "Congestion Avoidance and Control," *Proceedings of ACM SIGCOMM 1988*, pp. 314–329, August 1988.

[3] W. R. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmission, and Fast Recovery Algorithms," *IETF RFC 2001*, January 1997.

[4] J. C. Hoe, "Improving the Start-Up Behavior of a Congestion Control Scheme for TCP," *Proceedings of the ACM SIGCOMM 1996*, pp. 270–280, August 1996.

[5] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options," *IETF RFC 2018*, October 1996.

[6] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An Extension to the Selective Acknowledgment (SACK) Option for TCP," *IETF RFC 2883*, July 2000.

[7] M. Gerla, K. Tang, and R. Bagrodia, "TCP Performance in Wireless Multi-Hop Networks," *Proceedings of IEEE WMCSA 1999*, pp. 41-50, February 1999.

[8] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," *Proceedings of ACM MOBICOM 1999*, pp. 219-230, August 1999.

[9] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A Feedback-Based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks," *IEEE Personal Communications Magazine*, vol. 8, no. 1 , pp. 34-39, February 2001.

[10] D. Kim, C. K. Toh, and Y. Choi, "TCP-BuS: Improving TCP Performance in Wireless Ad Hoc Networks," *Journal of Communications and Networks*, vol. 3, no. 2, pp. 1-12, June 2001.

[11] C. K. Toh, "Associativity-Based Routing for Ad Hoc Mobile Networks," *Wireless Personal Communications*, vol. 4, no. 2, pp. 1-36, March 1997.

[12] J. Liu and S. Singh,"ATCP: TCP for Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1300-1315, July 2001.

[13] S. Kopparty, S. V. Krishnamurthy, M. Faloutsos, and S. K. Tripathi, "Split TCP for Mobile Ad Hoc Networks," *Proceedings of IEEE GLOBECOM 2002*, vol. 1, pp. 138-142, November 2002.

[14] J. Liu and S. Singh,"ATP: Application Controlled Transport Protocol for Mobile Ad Hoc Networks," *Proceedings of IEEE WCMC 1999*, vol. 3, pp. 1318-1322, September 1999.

[15] K. Sundaresan, V. Anantharaman, H. Y. Hsieh, and R. Sivakumar, "ATP: A Reliable Transport Protocol for Ad Hoc Networks," *Proceedings of ACM MOBIHOC 2003*, pp. 64-75, June 2003.

[16] Y. Hu, A. Perrig, and D. B. Johnson, "Packet Leashes: A Defense Against Wormhole Attacks in Wireless Ad Hoc Networks," *Proceedings of IEEE INFOCOM 2003*, vol. 3, pp. 1976-1986, April 2003.

[17] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An On-Demand Secure Routing Protocol Resilient to Byzantine Failures," *Proceedings of the ACM Workshop on Wireless Security 2002*, pp. 21-30, September 2002.

[18] C. E. Perkins and E. M. Royer, "Ad Hoc On-Demand Distance Vector Routing," *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90-100, February 1999.

[19] Y. Hu, A. Perrig, and D. B. Johnson, "Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols," *Proceedings of the ACM Workshop on Wireless Security 2003*, pp. 30-40, September 2003.

[20] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network Magazine*, vol. 13, no. 6, pp. 24-30, December 1999.

[21] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[22] A. Khalili, W. A. Arbaugh, "Security of Wireless Ad Hoc Networks," http://www.cs.umd.edu/~aram/wireless/survey.pdf.

[23] N. Asokan and P. Ginzboorg, "Key-Agreement in Ad Hoc Networks," *Computer Communications*, vol. 23, no. 17, pp. 1627-1637, 2000.

[24] S. Capkun, L. Buttyan, and J. P. Hubaux, "Self-Organized Public-Key Management for Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 52-64, January-March 2003.

[25] S. Yi, P. Naldurg, and R. Kravets, "Security-Aware Ad Hoc Routing for Wireless Networks," *Proceedings of ACM MOBIHOC 2001*, pp. 299-302, October 2001.

[26] Y. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks," *Proceedings of IEEE WMCSA 2002*, pp. 3-13, June 2002.

[27] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proceedings of ACM SIGCOMM 1994*, pp. 234-244, August 1994.

[28] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. B. Royer, "A Secure Routing Protocol for Ad Hoc Networks," *Proceedings of IEEE ICNP 2002*, pp. 78-87, November 2002.

[29] H. Deng, W. Li, and D. P. Agrawal, "Routing Security in Wireless Ad Hoc Networks," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 70-75, October 2002.

[30] P. Papadimitratos and Z. J. Haas, "Secure Routing: Secure Data Transmission in Mobile Ad Hoc Networks," *Proceedings of ACM Workshop on Wireless Security 2003*, pp. 41-50, September 2003.

[31] A. Fasbender, D. Kesdogan, and O. Kubitz, "Variable and Scalable Security: Protection of Location Information in Mobile IP," *Proceedings of IEEE VTC 1996*, vol. 2, pp. 963-967, May 1996.

[32] Y. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A Secure On-Demand Routing for Ad Hoc Networks," *Proceedings of ACM MOBICOM 2002*, pp. 12-23, September 2002.