Chapter 9

TCP over Ad Hoc Networks

9.1 Introduction

Over the past few years, MANETs have emerged as a promising approach for future mobile IP applications. This scheme can operate independently from existing underlying infrastructure and allows simple and fast implementation. Obviously, the more this technology evolves, the higher will be the probability of being an integral part of the global Internet [Perkins2002]. Therefore, considerable research efforts have been put on the investigation of the Transmission Control Protocol (TCP) [Comer1995] performance over MANETs.

As we know, TCP is the prevalent transport layer protocol in the IP world today and is employed by a vast majority of applications, especially those requiring reliability. More specifically, TCP is the most widely used transport protocol for data services like file transfer, email, web browsing, and so on. Therefore, it is essential to look at the TCP performance over MANETs. TCP is already fine-tuned to work well in wired environments; however in a MANET environment, its performance is highly degraded due to the high error rates and longer delays of wireless mediums, as well as mobility. In recent years, several improvements for TCP have been proposed for cellular [Balakrishnan1997, Zhang2001] wireless networks, but still much work has to be done for the MANETs paradigm.

Unlike cellular networks, where only the last hop is based on a wireless medium, MANETs are composed exclusively of wireless links, where multi-hop connections are often in place. Besides, in an ad hoc scenario, all nodes can move freely and unpredictably, which makes the clock based TCP congestion control quite hard. In particular, as the errors in wireless networks occur not only due to congestion but also due to medium constraints and mobility, TCP needs to distinguish nature of the error so that it can take the most appropriate action. Factors such as

path asymmetry and congestion window size also impact the performance. These and few other issues are addressed in this chapter.

Although there are a number of differences between cellular and MANETs, some of the ideas developed for the former can be used in the latter as well. As a matter of fact, many of the proposed TCP solutions for MANETs is a mixture of old concepts used for cellular networks. Nevertheless, we cover some solutions specifically tailored for MANETs, while many issues are still open.

9.2 TCP Protocol Overview

The TCP protocol is defined in the Request For Comment (RFC) standards document number 793 [Postel1981] by the Internet Engineering Task Force (IETF) [IETFwww]. The original specification written in 1981 was based on earlier research and experimentation in the original ARPANET. It is important to remember that most applications on the Internet make use of TCP, relying upon its mechanisms that ensure safe delivery of data across an unreliable IP layer below. In this section we explore the fundamental concepts behind TCP and how it is used to transport data between two endpoints. More specifically, we focus on those aspects of TCP that are of importance to understand its performance over MANETs. TCP adds a great deal of functionality to the IP service it is layered over:

- **Streams:** TCP data is organized as a stream of bytes, much like a file. The datagram nature of the network is concealed. A mechanism (the *Urgent Pointer*) exists to let out-of-band data be specially flagged.
- **Reliable delivery:** Sequence numbers are used to coordinate which data has been transmitted and received. TCP will arrange for retransmission if it determines that data has been lost;
- **Network adaptation:** TCP will dynamically learn the delay characteristics of a network and adjust its operation to maximize throughput without overloading the network.
- **Flow control:** TCP manages data buffers, and coordinates traffic so that its buffers will never overflow. Fast senders will be stopped periodically to keep up with slower receivers.

9.2.1 Designed and Fine-Tuned to Wired Networks

The design of TCP has been heavily influenced by what is commonly known as the "end-to-end argument" [Clark1988]. As it applies to the wired Internet, the system gets unnecessarily complicated by putting excessive intelligence in physical and link layers to handle error control, encryption or flow control. While these functions need to be done at the endpoints anyway, the result is the provision of minimal functionality on a hop-by-hop basis and maximal control between end-to-end communicating systems.

TCP performance is often dependent on the flow control and the congestion control. Flow control determines the rate at which data is transmitted between a sender and a receiver. Congestion control defines the methods for implicitly interpreting signals from the network in order for a sender to adjust its rate of transmission. Ultimately, intermediate devices, such as IP routers, would only be able to control congestion. A recent study on congestion control examines the current state of activity [Kristoff2000].

Timeouts and retransmissions handle error control in TCP. The nature of TCP and underlying packet switched network provide formidable challenges for managers, designers and researchers. It is important to incorporate link layer acknowledgements and error detection/correction functionality in TCP for wireless networks. Furthermore, when we consider MANETs, mobility comes into picture. Therefore, higher error rates, longer delays, and mobility makes MANET environments extremely challenging to the implementation of TCP as it tears down most the assumptions over which TCP was designed.

9.2.2 TCP Basics

TCP is often described as a byte stream, connection-oriented, full-duplex, reliable delivery transport layer protocol. In this subsection, we discuss the meaning for each of these descriptive terms.

9.2.2.1 Byte Stream Delivery

TCP interfaces between the application layer above and the network layer below. When an application sends data to TCP, it does so in 8-bit byte streams. It is then up to the sending TCP to segment or

delineate the byte stream in order to transmit data in manageable pieces to the receiver. It is this lack of "record boundaries" which gives it the name "byte stream delivery service".

9.2.2.2 Connection-Oriented

Before two communicating TCP entities (the sender and the receiver) can exchange data, they must first agree upon the willingness to communicate. Analogous to a telephone call, a connection must first be made before two parties exchange information.

9.2.2.3 Full-Duplex

No matter what a particular application may be, TCP almost always operates in full-duplex mode. It is sometimes useful to think of a TCP session as two independent byte streams, traveling in opposite directions. No TCP mechanism exists to associate data in the forward and reverse byte streams, and only during connection start and close sequences can TCP exhibit asymmetric behavior (i.e., data transfer in the forward direction but not in the reverse, or vice versa).

9.2.2.4 Reliability

A number of mechanisms help providing the reliability TCP guarantees. Each of these is described briefly below:

- **Checksums**: All TCP segments carry a checksum, which is used by the receiver to detect errors with either the TCP header or data.
- **Duplicate data detection**: It is possible for packets to be duplicated in packet switched network; therefore TCP keeps track of bytes received in order to discard duplicate copies of data that has already been received.
- **Retransmissions**: In order to guarantee delivery of data, TCP must implement retransmission schemes for data that may be lost or damaged. The use of positive acknowledgements by the receiver to the sender confirms successful reception of data. The lack of positive acknowledgements, coupled with a timeout period (see timers below) calls for a retransmission.
- **Sequencing**: In packet switched networks, it is possible for packets to be delivered out of order. It is TCP's job to properly sequence

segments it receives so that it can deliver the byte stream data to an application in order.

- **Timers**: TCP maintains various static and dynamic timers on data sent. The sending TCP waits for the receiver to reply with an acknowledgement within a bounded length of time. If the timer expires before receiving an acknowledgement, the sender can retransmit the segment.

9.2.3 TCP Header Format

As we know, the combination of TCP header and TCP in one packet is called a TCP segment. Figure 9.1 depicts the format of all valid TCP segments. The size of the header without options is 20 bytes. Below we briefly define each field of the TCP header.

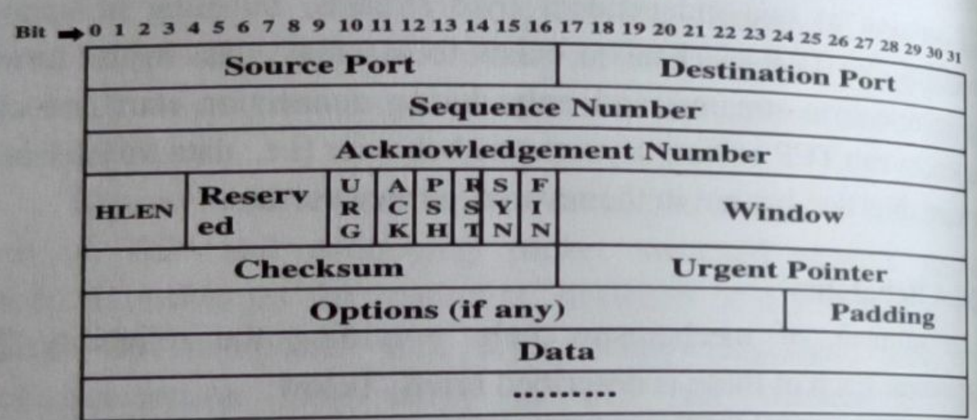

Figure 9.1 – TCP header format

Source Port: This is a 16-bit number identifying the application where the TCP segment originated from within the sending host. The port numbers are divided into three ranges: well-known ports (0 through 1023), registered ports (1024 through 49151) and private ports (49152 through 65535). Port assignments are used by TCP as an interface to the application layer. For example, the TELNET server is always assigned to the well-known port 23 by default on TCP hosts. A pair of IP addresses (source and destination) plus a complete pair of TCP ports (source and destination) defines a single TCP connection that is globally unique.

- **Destination Port:** A 16-bit number identifying the application TCP segment is destined for on a receiving host. Destination ports use the same port number assignments as those set aside for source ports.

- **Sequence Number:** Within the entire byte stream of the TCP connection, a 32-bit number, identifying the current position of the first data byte in the segment. After reaching 2^{32} -1, this number will wrap around to 0.
- **Acknowledgement Number:** This is a 32-bit number identifying the next data byte the receiver expects from the sender. Therefore, this number will be one greater than the most recently received data byte. This field is used only when the ACK control bit is turned on.
- **Header Length:** A 4-bit field that specifies the total TCP header length in 32-bit words (or in multiples of 4 bytes). Without options, a TCP header is always 20 bytes in length. On the other hand, the largest a TCP header is 60 bytes. Clearly, this field is required because the size of the options field(s) cannot be determined in advance. Note that this field is called "data offset" in the official TCP standard, but header length is more commonly used.
- **Reserved:** A 6-bit field currently unused and reserved for future use.
- **Control Bits:**
  - *Urgent Pointer (URG)* – If this bit field is set, the receiving TCP should interpret the urgent pointer field (see below);
  - *Acknowledgement (ACK)* – If this bit is set, the acknowledgment field is valid.
  - *Push Function (PSH)* – If this bit is set, the receiver should deliver this segment to the receiving application as soon as possible. An example of its use may be to send a Control-C request to an application, which can jump ahead of queued data.
  - *Reset Connection (RST)* – If this bit is present, it signals the receiver that the sender is aborting the connection and all the associated queued data and allocated buffers can be freely relinquished.
  - *Synchronize (SYN)* – When present, this bit field signifies that the sender is attempting to "synchronize" sequence numbers. This bit is used during the initial stages of connection establishment between a sender and a receiver.
  - *No More Data from Sender (FIN)* – If set, this bit field tells the receiver that the sender has reached the end of its byte stream for the current TCP connection.

- **Window:** This is a 16-bit integer used by TCP for flow control in the form of a data transmission window size. This number tells the sender how much data the receiver is willing to accept. The maximum value for this field would limit the window size to 65,535 bytes. However, a "window scale" option can be used to make use of even larger windows.
- **Checksum:** A TCP sender computes the checksum value based on the contents of the TCP header and data fields. This 16-bit value will be compared with the value the receiver generates using the same computation. If the values match, the receiver can be very confident that the segment arrived intact.
- **Urgent Pointer:** In certain circumstances, it may be necessary for a TCP sender to notify the receiver of urgent data that should be processed by the receiving application as soon as possible. This 16-bit field tells the receiver when the last byte of urgent data in the segment ends.
- **Options:** In order to provide additional functionality, several optional parameters may be used between a TCP sender and a receiver. Depending on the option(s) used, the length of this field varies in size, but it cannot be larger than 40 bytes due to the maximum size of the header length field (4 bits). The most common option is the maximum segment size (MSS) option where a TCP receiver tells the TCP sender the maximum segment size it is willing to accept. Other options are often used for various flow control and congestion control techniques.
- **Padding:** Because options may vary in size, it may be necessary to "pad" the TCP header with zeroes so that the segment ends on a 32-bit word boundary as defined by the standard.
- **Data:** Although not used in some circumstances (e.g., acknowledgement segments with no data in the reverse direction), this variable length field carries the application data from TCP sender to receiver. This field coupled with the TCP header fields constitutes a TCP segment.

9.2.4 Congestion Control

TCP congestion control and Internet traffic management issues in general is an active area of research and experimentation. Although this section is a very brief summary of the standard congestion control algorithms widely used in TCP implementations today, it covers the main points necessary to understand behavior of the TCP over MANETs. These algorithms are defined in [Jacobson1988] and [Jacobson1990a], and the most update version of the TCP congestion control algorithm can be found in [Allman1999].

9.2.4.1 Slow Start

Slow Start, a requirement for TCP software implementation, is a mechanism used by the sender to control the transmission rate, otherwise known as sender-based flow control. The rate of acknowledgements returned by the receiver determines the rate at which the sender can transmit data. Whenever a TCP connection starts, the Slow Start algorithm at the sender initializes a *congestion window* (CWND) to one segment. As the connection is carried out and acknowledgements are returned by the receiver, the congestion window increases by one segment for each acknowledgement returned. Thus, the sender can transmit the minimum of the congestion window and the advertised window (contained in the header of the acknowledgment packet) of the receiver, which is simply called the transmission window and is increased exponentially.

9.2.4.2 Congestion Avoidance

During the initial data transfer phase of a TCP connection, the Slow Start algorithm is used. However, there may be a point during Slow Start that the network is forced to drop one or more packets due to overload or congestion. If this happens, Congestion Avoidance is used to reduce the transmission rate. However, Slow Start is used in conjunction with Congestion Avoidance in order to get the data transfer going again so it does not slow down and stay slow. In the Congestion Avoidance algorithm, the expiration of a timer called *retransmission timeout* (RTO) or the reception of duplicate ACKs implicitly signal the sender that a network congestion situation is occurring. The sender immediately sets

avoid standard TCP senders from being affected by reduced ACK frequency. This technique encompasses a soft-state agent (at the sender) called *ACK reconstructor* which receives the spaced ACKs from the receiver and paces its relaying to the sender in a regulated rate. This rate is based on the amount of data acknowledged by the received ACK and also on the actual rate of the backward path. As a result, the CWND growing at the sender is controlled by the actual rate in the backward path and no burst behavior arises.

As mentioned above, asymmetry can also induce inaccuracy on RTT estimation. As we have seen before, TCP estimates its RTT based on measurements of the delay suffered by a packet in both direction of the flow. Therefore, it can so happen that such estimation does not suit the actual forward path necessity, which might lead TCP to retransmit either prematurely or too late. In [Parsa1999], *TCP Santa Cruz* is proposed as a solution for decoupling CWND growth from the number of ACKs received. This scheme relies on measurements of relative delay that one packet experiences in relation to the previous packet, rather than on measurements of absolute delay for sampled packets as standard TCP does. This way, it pursues not only to provide enhanced RTT estimation but also to be resilient to ACK losses. Thus, under realistic assumption that MANETs will experience asymmetrical paths, the above mechanisms as well as other related ones [Oliveira2005] should be carefully investigated in the context of this challenging environment.

9.4 Solutions for TCP over Ad Hoc

We now present the most prominent schemes that have been specifically proposed to overcome TCP performance problems in MANETs. Here, we classify the proposed solutions into *Mobility-related* and *Fairness-related* based on the key TCP issue they aim to overcome. The mobility-related approaches address the TCP problems resulting from node mobility which may mistakenly trigger TCP congestion control mechanisms. On the other hand, fairness-related solutions tackle the serious unfairness conditions raised when TCP is run over MANETs.

9.4.1 Mobility-Related

Notably, most of the solutions in this category have limitations which can compromise their widespread deployment. Nevertheless, it is of paramount importance to understand them as they may serve as the basis for future research. In the following we discuss the details of each one of them.

9.4.1.1 TCP-Feedback

As the name suggests, TCP-Feedback (TCP-F) [CRVP97] is a feedback-based scheme in which the TCP sender can effectively distinguish between route failure and network congestion by receiving Route Failure Notification (RFN) messages from intermediate nodes. The idea is to push the TCP into a "snooze state" whenever such messages are received. In this state, TCP stops sending packets and freezes all its variables such as timers and CWND size, which makes sense once there is no available route to the destination. Upon receipt of a Route Re-establishment Notification (RRN) message from the routing protocol, indicating that there is again an available path to the destination, the sender leaves the frozen state and resumes transmission using the same variables values prior to the interruption. In addition, a *route failure timer* is employed to prevent infinite wait for RRN messages, and is started whenever a RFN is received. Upon expiration of this timer, the frozen timers of TCP are reset hence allowing the TCP congestion control to be invoked normally.

Results from TCP-F shows gains over standard TCP in conditions where the route reestablishment delay are high, which are due to a fewer number of involved retransmission. Nevertheless, the simulation scenario employed to evaluate TCP-F has been quite simplified and so the results might not be a true representative. For example, the RFN and RRN messages employed in TCP-F are to be carried by the routing protocol, but no such protocol has been considered.

9.4.1.2 The ELFN Approach

The Explicit Link Failure Notification (ELFN) [Holland1999] is a cross-layer proposal in which TCP also interacts with the routing

protocol in order to detect route failure and take appropriate actions. Here, ELFN messages are sent back to the TCP sender from the node detecting the failure. Such messages are carried by the routing protocol that needs to be adapted for this purpose. In fact, the DSR's route error message has been modified to carry a payload similar to the "host unreachable" message of the Internet Control Message Protocol (ICMP) [Tanenbaum1996]. Basically, the ELFN messages contain sender and receiver addresses and ports, as well as the TCP sequence number. This way, the modified TCP is able to distinguish losses caused by congestion from the ones due to mobility. In ELFN, whenever the TCP sender receives an ELFN message it enters a "stand-by" mode in which all timers are disabled and probe packets are sent regularly towards the destination in order to detect route restoration. Upon receiving an ACK packet, the sender leaves the "stand-by" mode and resumes transmission using its previous timer values and state variables.

This scheme was only evaluated for the DSR routing protocol where the stale route problem was found to be crucial for the performance of ELFN. Additionally, the interval between transmission of probe packets and the choice of type of packet to be sent as a probe have also been evaluated. In essence, it has been suggested that a varying interval based on RTTs values could perform better than the fixed probe interval. In general, the ELFN approach provides meaningful enhancements over the standard TCP, but further evaluation may be needed. For instance, different routing protocols should be studied, and the performance of ELFN under congestion conditions should be considered. Last, but not the least, more appropriate values for the probe interval should be determined.

9.4.1.3 Fixed RTO

The fixed RTO scheme [Dyer2001] relies on the idea that routing error recovery should be accomplished in a fast fashion by the routing algorithm. As a result, any disconnection should be treated as a transitory period which does not justify the regular exponential backoff mechanism of TCP being invoked, as this can cause unnecessarily long recovery delays. Thus, it disables such a mechanism whenever two successive retransmissions due to timeout occur, assuming that it actually

indicates route failure. By doing so, it allows the TCP sender to retransmit at regular intervals instead of at increasingly exponential ones. In fact, the TCP sender doubles the RTO once and if the missing packet does not arrive before the second RTO expires, the packet is retransmitted again and again but the RTO is no longer increased. It remains fixed until the route is recovered and the retransmitted packet is acknowledged.

The fixed RTO approach has been evaluated in [Dyer2001] considering different routing protocols along with TCP selective and delayed acknowledgements options. Sizeable enhancements have been accomplished with on-demand routing protocols, but only marginal improvements have been noticed when using the TCP options. Nevertheless, this proposal is limited to wireless networks only, which makes it somewhat discouraging as interoperation with wired networks is a mandatory requirement in the vast majority of applications.

9.4.1.4 The ATCP Protocol

Different from previously discussed approaches, the Ad hoc TCP (ATCP) protocol [Liu2001] does not impose changes to the standard TCP itself. Rather, it implements an intermediate layer between the network and the transport layers in order to provide an enhanced performance to TCP and still maintain interoperation with non-ATCP nodes.

More specifically, ATCP relies on the ICMP protocol and on the Explicit Congestion Notification (ECN) [ECNwww] scheme to detect/distinguish network partition and congestion, respectively. This way, the intermediate layer keeps track of the packets to and from the transport layer so that the TCP congestion control is not invoked when it is not really needed, which is done as follows. Whenever three duplicate ACKs are detected, indicating a lossy channel, ATCP puts TCP in "persistent mode" and quickly retransmits the lost packet from the TCP buffer; after receiving the next ACK, the normal state is resumed. In case an ICMP "Destination Unreachable" message arrives, pointing out a network partition, ATCP also puts the TCP in "persistent mode" which only ends when the connection is reestablished. Finally, when network congestion is detected by the receipt of an ECN message, the ATCP does

regions such as triangle tile, square tile, hexagon tile, ring, star, and linear are discussed. The sensor placement strategy evaluation is based on three goals: resilience to single point of failure, the area of interest has to be covered by at lease one sensor, and minimum number of nodes. Finally, it is found that the choice of placement depends on distance between SNs and the sensing radius.

10.6.3 Environmental Monitoring

The use of sensors in monitoring the landfill and the air quality have been suggested [Agrawal2004]. Household solid waste and non-hazardous industrial waste such as construction debris and sewer sludge are being disposed off by using over 6000 landfills in USA and associated organic components undergo biological and chemical reaction such as fermentation, biodegradation and oxidation-reduction. This causes harmful gases like methane, carbon monoxide (CO), carbon dioxide, nitrogen, sulfide compounds and ammonia to be produced and migration of gases in the landfill causes physical reactions which eventually lead to ozone gas that is known to be a primary air pollutant and an irritant to our respiratory systems.

The current method of monitoring landfill employs periodic drilling of collection well, collecting gas samples in airtight bags and analyze off-site, making the process very time consuming. So, the idea is to interface gas sensors with custom-made devices and wireless radio and transmit sensed data for further analysis and appropriate collective action. Deployment of a large number of sensors allows real-time monitoring of gases being emitted by the waste material or from industrial spills and allows cost-effective remote control. Among the hazardous air pollutants, CO is monitored continuously at very few selected places with high traffic volumes due to installation and maintenance cost. The idea is to place a large number of sensors throughout the area of interest and appropriate type of sensors can be placed according to the type of pollutant anticipated in a given area [Agrawal2004]. A large volume of raw data from sensors can be collected and processed. Efficient retrieval of information using appropriate queries is also discussed to have distributed decision making [Biswas2005]. Effective power aware ways of gathering information

from sensors are also covered using clustering approach and use of sub-optimal paths for long periodic queries and shortest paths for time-critical queries are also discussed. Selection and placement of sensors, their calibration and interfacing with the wireless network and data integration with geographical information systems, are also considered. The generic scheme can be easily used and adopted for other applications like smoke and gas detector, coastal observing system, etc.

10.6.4 Environment Observation and Forecasting System

The Environment Observation and Forecasting System (EOFS) is a distributed system that spans large geographic areas and monitors, models and forecasts physical processes such as environmental pollution, flooding, among others. Usually, it consists of three components: sensor stations, a distribution network, and a centralized processing.

CORIE [CORIEwww] is a prototype of EOFS for the Columbia River (Oregon, USA) which integrates a real-time 13-sensors placed on piers with one mobile sensor drifting off-shore for data management. The stationary stations are powered by a power grid, while the mobile station uses solar panel to harness solar energy. Sensor data are transmitted via wireless links toward on-shore master stations which, in turn, forward the data to a centralized server where a computationally intensive physical environment model is used to guide vessel transportation and forecasting.

Practical difficulties arise from issues such as power supply and antenna affixation for the off-shore sensor nodes and frequently obscured line-of-sight due to height of surface waves. The Automated Local Evaluation in Real-Time (ALERT) [Alertwww] is probably the first well-known wireless sensor network being deployed across most of the western United States and is heavily used for flood alarming in California and Arizona. It was developed by the National Weather Service in the 1970's providing important real-time rainfall and water level information to evaluate the possibility of potential flooding. ALERT sensor sites are usually equipped with meteorological/hydrological sensors, such as water level sensors, temperature sensors, and wind sensors. Data are transmitted via line-of-sight radio communication from the sensor site to the base station, where a Flood Forecast Model is adopted to process the data and issue automatic warnings. Web-based queries are also available.

10.6.5 Drinking Water Quality

A sensor based monitoring system has recently been proposed [Ailamaki2003], with the emphasis on placement and utilization of in situ sensing technologies and doing spatial-temporal data mining for water-quality monitoring and modeling. The main objective is to develop data-mining techniques to water-quality databases and use them for interpreting and using environmental data. This also helps in controlling addition of chlorine to the treated water before releasing to the distribution system. Detailed implementation of a bio-sensor for incoming wastewater treatment has been discussed in [Melidis2005]. A pilot-scale and full scale system has also been described.

10.6.6 Disaster Relief Management

Novel sensor network architecture has been proposed in [Cayirci2004] that could be useful for major disasters including earthquakes, storms, floods, fires and terrorist attacks. The SNs are deployed randomly at homes, offices and other places prior to the disaster and data collecting nodes communicate with database server for a given sub area which are in-turn linked to a central database for continuous update. Under normal operating conditions, the database servers from different sub areas are connected by a backbone and any disruption due to disaster could force them to be connected either via a satellite or a low-flying aerial vehicle. Steps are also discussed for establishing a route, disseminating a task and getting sensed data using a selected route. Hello messages are used to determine if a SN is alive or dead and signal strength indicates relative distance and direction of reporting SN.

Based on the statistical data from 1999 Izmit earthquake, various performance curves are obtained to indicate required average number of active SNs to detect a disaster, probability of the disaster to be within the sensing range of at least one SN for percentage of SNs failed, total number of transmitted packets, and the number of SNs failed due to energy depletion. This could also be very helpful in seismic monitoring, glacier movements, and underwater sensing.

10.6.7 Soil Moisture Monitoring

A soil moisture monitoring scheme using sensors, have been developed over a one hectare outdoor area [Oliver2005] and various performance parameters have been measured from an actual system. A custom made moisture sensor is interfaced with Mica 2 Mote wireless board. In place of monitoring of the moisture level, a rain gauge is used to wake up the SNs from the sleep mode (2-hour after 1 mm of rains) and such reactive triggering eliminates the need for clock synchronization, clock-drift, time stamping and time setting, thereby drastically saving the energy consumed by WSN. This also helps in achieving robustness and longevity.

A base node linked to GSM gateway, collects information for SNs and SMAC with 4-way handshake of (RTS-CTS-DATA-ACK) the MAC protocol. 100% message delivery success has been observed in the laboratory tests while the field trails show dependence of delivery rate whether it is rainy or dry. End-to-end message delivery in field trails is also given for 10 minutes of rain followed by 2 hours of dry period. The longest interval with no data is observed to be 12 hours (6-readings) for dry rate and 3-5 hours (21 readings) for rain rate. For improving robustness, the researchers also suggest the use of multi-path routing between critical pairs or simply doubling the numbers of SNs and plan to undertake in their future work. The sensors are observed to be $\pm 1\%$ accurate when calibrated for specific soil and $\pm 3\%$ without calibration. The Berkley Mote hardware is seen to draw 5-20 milliamps during active period and 5 microamps during sleep. With 100% duty cycle of motes, the alkaline batteries falls below 2.7 V after 18 hours while 100 hours for NiMH batteries. The researchers are experimenting to generalize the event conditions.

10.7 Health Care Monitoring

Applications in this category include telemonitoring of human physiological data, tracking and monitoring of doctors and patients inside a hospital, drug administrator in hospitals, and so on [Akyildiz2002]. The use of fixed sensors has also been explored and preliminary results provided in monitoring health of cattle [Mayer2004] by checking food and water availability. This is done by measuring intra-ruminal

movement of cows by accelerometer and characterizing the feeding cycle. Micomotes have been used to set up an experimental system. A considerable difference is also observed on the feed cycle and the amount of water at different places. The health of a cow is to be predicted by observing when drinking occurs, how much water is digested, what kind of mixing occurs in the rumen and the amount of heat generated when fermentation of digested feed occurs, and how cold the optimum distribution of water within a paddock.

10.8 Building, Bridge, and Structural Monitoring

Several recent projects have explored the use of sensors in monitoring the health of buildings, bridges and highways. A Bluetooth based scatternet has been proposed [Mehta2004] to monitor stress, vibration, temperature, humidity etc. in civil infrastructures and rational for using. Simulation results are given to justify effectiveness of their solution by having a set of rectangular Bluetooth equipped sensor grids to model a portion of bridge span.

Fiber optic based sensors have been proposed for monitoring crack openings due to strain and corrosion of the reinforcement in concrete bridge decks and structures [Casas2003]. Possible use of different types of interferometer sensors for gradual structural degradation has been explored and their relative advantages and disadvantages are discussed. Impact of temperature on the accuracy of strain monitoring sensors, have also been pointed out and the use of a new family of inclinometers has been suggested to overcome the temperature sensitivity. Corrosion of steel bars is measured by using special super glue and angular strain sensors.

Feasibility of monitoring various risks for buildings using wireless acceleration sensors and consider microphones interfaced to Mica 2 Motes have been tested [Kuratawww]. The idea is to check a building for degraded structural performance, fatigue damage, gas leaks, intrusion, fires, etc. for appropriate actions such as structural control, maintenance, evacuation advice, alarms and warning, fire fighting and rescue operation, necessary security measures, etc. The acceleration and strain at different parts of house beam and columns, temperature/light and sound in each room can be used to detect earthquake/wind, fires and