Multi-Modal Network Protocols: Adapting to Highly Variable Operating Conditions[1]

Aditya Akella[2] Ashwin Bharambe[3] Suman Nath[4]
Srinivasan Seshan[5]

August 2002
CMU-CS-02-170

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Most network protocols are *uni-modal*: they employ a single set of algorithms that allows them to cope well only within a narrow range of operating conditions. This rigid design renders these protocols inefficient in the face of widely varying operating environments or in conditions different from the ones for which they are optimized. Such uni-modal protocols have great difficulty in the mobile computing world where the operating conditions, including number of nodes, computational capabilities and rate of mobility, are not fixed.

Consider, for example, routing in a network of ad-hoc nodes. Solutions like DSDV work well when the number of nodes is small. Unfortunately, such schemes scale poorly to larger population sizes. In such situations, more scalable algorithms that impose a structure on the network of ad-hoc nodes, in a manner similar to routing protocols in the Internet, provides better results. However, these scalable algorithms tend to incur high overheads in situations that DSDV handles well. Clearly, no single routing solution handles all situations that a node may encounter.

Motivated by such examples, this paper attempts to answer the following question: Is it possible to redesign the traditional protocols to take on very different operating modes when faced with different environments? We present a case for such *multi-modal* protocols in our paper. Specifically, we discuss multi-modal reliability and routing. We show the feasibility of designing multi-modal protocols by describing how these protocols can make operating mode decisions and switch modes without additional overhead.

[1]This research was sponsored by DARPA under contract F30602-99-1-0518. Additional support was provided by IBM. Views and conclusions contained in this document are those of the authors and should not be interpreted as respresenting the official policies, either expressed or implied, of DARPA, IBM or the U.S. government.

[2]Computer Science Department, Carnegie Mellon University (aditya@cs.cmu.edu).
[3]Computer Science Department, Carnegie Mellon University (ashu@cs.cmu.edu)
[4]Computer Science Department, Carnegie Mellon University (sknath@cs.cmu.edu)
[5]Computer Science Department, Carnegie Mellon University (srini@cmu.edu).

Keywords: Congestion Control, Active Queue Management

1 Introduction

Most of today's network protocols are designed only to operate in a specific environment. As a result, these protocols typically incorporate some basic assumptions about the target environment and can adapt to some expected range of conditions. These protocols perform poorly when these assumptions are violated and operating conditions exceed the typical ranges. For example, LAN network protocols often use broadcast based on the assumption that most LAN networks have few nodes. However, such techniques prove to be inefficient in large bridged LANs with many nodes. Fortunately, these assumptions are rarely violated in the Internet since conditions do not vary greatly and administrators often ensure that the assumptions are not violated.

Many of today's protocols adapt their behavior to the operating conditions. However, to cope with widely varying operating environments, we believe it is necessary to design network protocols that can dynamically change their operating mode depending on the estimated environmental conditions. These different modes of operation typically use different types of algorithms. We refer to such protocols as being *multi-modal*. The key objective of this paper is to explore the general challenges in building and benefits of using multi-modal designs for network protocols.

This design methodology is especially important in the wireless, mobile world where the number of nodes, the capabilities of nodes, and the wireless transmission quality can all change from situation to situation. We explore the design of multi-modal protocols using reliability and routing protocols in a wireless environment.

Today's reliable transport protocols have difficulty performing well in varied environments. For example, one assumption that TCP makes is that losses are relatively infrequent and are a result of congestion. TCP can be made to operate efficiently in wireless networks with transmission losses with the overhead of adding a Snoop protocol agent [8] at basestations. In environments with long disconnections and extremely high loss rates, it may be desirable to employ split-connection approaches that trade-off end-to-end semantics and higher overheads to better shield the sender from the wireless conditions [5, 9]. In order to obtain good wireless throughput, our multi-modal transport protocol chooses between schemes like split connections, intelligent link-layers and normal transmission. This protocol incurs the expenses of the higher overhead scheme only when the performance improvement justifies the expense.

The multi-modal routing protocol that we design adapts to the population and availability of location nodes. This design is necessary because the performance of ad-hoc routing protocols is sensitive to the size and number of nodes in the network. For example, solutions like DSDV [27] work well when the number of nodes is small. However, such schemes scale poorly to larger population sizes. More scalable algorithms that impose a structure on the network of ad-hoc nodes, in a manner similar to routing protocols in the Internet, provide better results. However, these scalable algorithms tend to incur high overheads in situations that DSDV handles well. Our multi-modal routing protocol is designed to work efficiently in more diverse situations.

While the above discussion indicates some of the possible benefits of using multiple operating modes, actually realizing these gains in a multi-modal protocol creates several challenges. First, a protocol must support a diverse set of operating modes and be able to identify the right mode given an estimate of the current operating conditions. Second, multi-modal protocols must incorporate a low overhead method of measuring the necessary environmental conditions in a accurate and timely way. Third, multi-modal protocols require efficient, low-overhead mechanisms for switching between the candidate modes. For both of our multi-modal protocols, we describe how these challenges can be met.

In this paper, we demonstrate the benefits of multi-modal network protocol design through the design of multi-modal reliability and routing protocols. For each of these protocols, we analyze the trade-offs between the various candidate operating modes as a function of operating environment and develop low-overhead mechanisms for switching from one mode to another. This paper is organized as follows. In Section 2, we discuss related work. Section 3 presents our multi-modal reliability protocol while Section 4 discusses our multi-modal routing protocol for a network of ad-hoc nodes. Section 5 describes some of our initial thoughts on other multi-modal protocols. Finally, Section 6 summarizes the contributions of this paper.

2 Related Work

Many network protocols by their very nature have to be adaptive. This form of adaptation is typically achieved through a single algorithm that works well within a predetermined range of operating conditions. An example of an adaptive algorithm is the Ethernet back-off algorithm. Backing off is a mechanism used by each sender to adapt to the presence of other possible senders. A possible multi-modal version of back-off could use different back-off algorithms based on long-term estimates of the number of nodes on the wire or the amount of competing traffic. The modes of operation could correspond to a less conservative back-off when the number of competing senders is low and a more conservative (exponential) back-off when the number of senders is high. In general, multi-modal algorithms have a number of candidate algorithms in stock, each of which is employed under a given set of estimated operating conditions.

In this section, we concentrate on work related to multi-modal design, not adaptive design. We also briefly discuss some of the related work in the areas relevant to the design of multi-modal reliability and routing.

2.1 Multi-Modal Design

Some network protocols already exhibit multi-modal behavior. However, this multi-modal behavior has largely been achieved by accident rather than by design. A different area where multi-modal design is more common is the area of adaptive application systems. Here, we describe some of the existing systems in both areas that employ multi-modal techniques.

The congestion control and loss recovery algorithms employed by TCP [17, 4] provide a good example of multi-modal behavior. When the congestion in the network is low and only a few losses are observed, a TCP flow (say a TCP Newreno flow) typically only reduces its congestion window by half, retransmits the lost packet(s) and continues normal operation thereafter. However, when the congestion in the network is high and multiple losses occur, a TCP flow adopts a completely different loss recovery algorithm: it incurs a timeout and undergoes slow start. These two modes result in grossly different loss-throughput relationships, as shown in [23]. Thus, by employing different mechanisms depending on the estimate of the operating conditions, TCP exhibits multi-modal behavior.

Two examples of multi-modal application systems are CoopNet [24] and Odyssey [22]. The CoopNet project is a multi-modal Web server for static Web content. In CoopNet, clients are willing to co-operate with a server to help reduce the server's load when the number of requests becomes very large. When the web-server is overloaded, it can redirect requests to the CoopNet collection of clients. Thus, CoopNet uses an estimate of its current operating environment (e.g. load) to switch between different modes (the web-server replying to all requests vs. the cooperative clients serving the request). The Odyssey system is designed to support multi-modal applications. The Odyssey software running on a mobile host provides measurements of various environmental factors like bandwidth, CPU cycles and battery power. Odyssey applications are expected to use these measurements to determine the proper mode of operation. For example, an application on a mobile that is running its CPU slowly may choose to use a less compute intensive rendering algorithm. In addition, given the current network conditions, the application may also offload the rendering to some remote compute server. The Odyssey project [22] has explored redesigning a number of applications using these techniques.

2.2 Reliability and Routing

Our multi-modal reliability and routing protocols borrow liberally from a host of past literature. One of our routing protocol modes is based on the location proxy algorithm presented in [13]. This paper extends geographic routing [14, 21] to networks in which some of the nodes are unaware of their geographic location (location-unaware nodes). This modes also relies on the development of location services like GLS [21]. We could have used a variety of other similar scalable and/or hierarchical solutions (e.g. Leach [16], SPAN [10]) to highlight the advantages of a multi-modal routing system. Our other routing protocol mode is based on DSDV [27]. Similarly, we could have used a variety of different solutions (e.g. DSR [19], AODV [26], TORA [25]) for this mode.

Our reliability protocol relies heavily on the insight provided by [8, 5, 9] on the challenges of reliable transfers over wireless links. We use the same taxonomy of reliable protocols as does [8] to guide our selection of different operating modes.

3 Multi-Modal Reliability

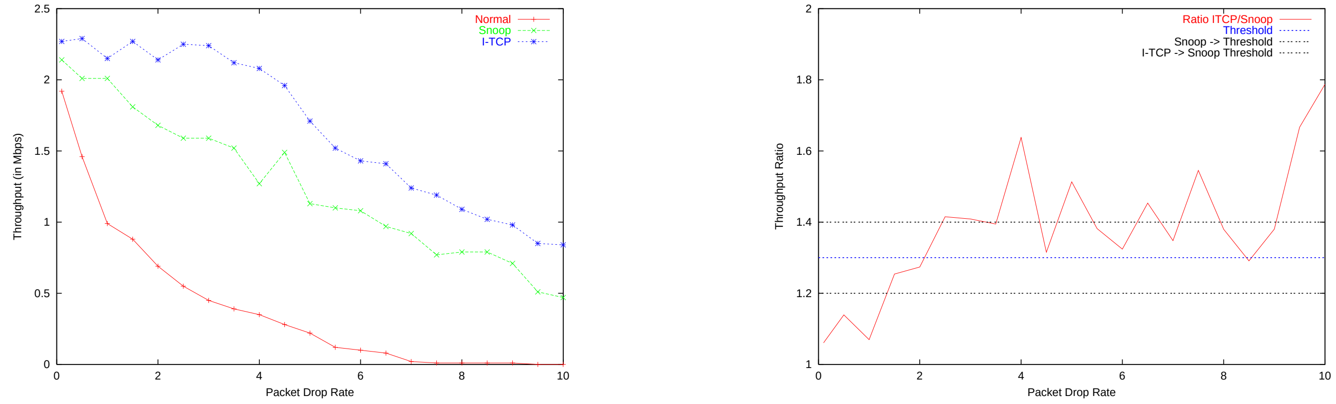
Networks with wireless and other lossy links suffer from packet drops due to bit errors and disconnections. When no mechanisms are employed to actively detect such local (wireless) losses and perform local retransmissions intelligently, the TCP sender is forced to assume that the observed losses are congestion-related and, therefore, reduces its congestion window unnecessarily. This improper behavior results in link underutilization and low throughput. Several schemes have been proposed in the past to alleviate the effects of non-congestion related losses. These techniques include TCP-Aware link-level retransmissions (Snoop) [8], split TCP connections (`I-TCP`) [5] and end-to-end selective acknowledgements with or without explicit loss notification (SACK or SACK with ELN). However, as we show in this section, none of these techniques are optimal in all conditions. In this section, we describe how to implement a multi-modal approach to this problem that addresses the three key challenges of incorporating a set of modes, measuring the operating conditions and performing the switch between modes.

3.1 Modes of Operation

In order to determine which modes are appropriate for multi-modal reliable data transmission, we must first identify the tradeoffs between the different techniques. We evaluate the following three techniques based on the taxonomy presented in [7]:

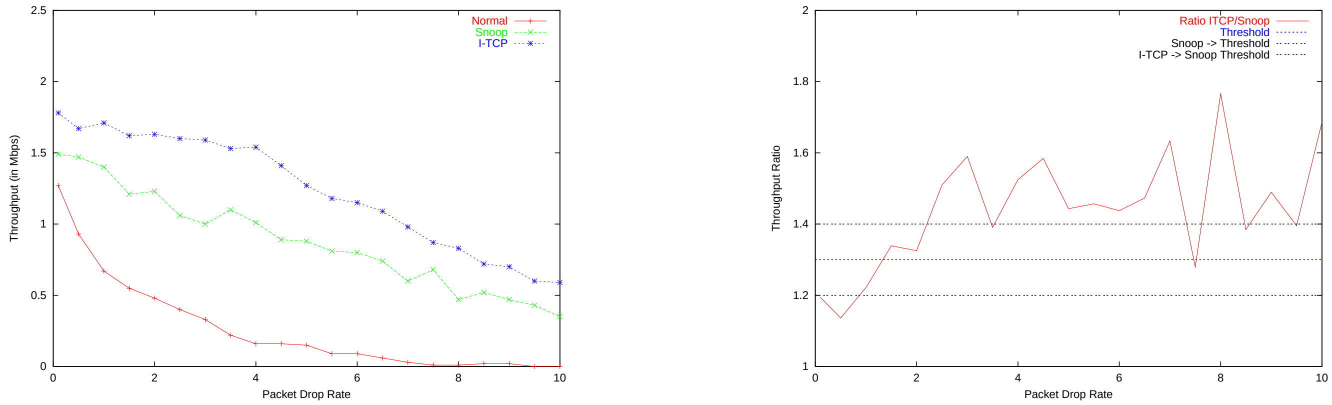
- **Normal.** No special mechanism is employed and only the standard (SACK) end-to-end loss recovery is used.
- **Snoop [8].** Losses are recovered by a TCP-aware reliable link-layer. This scheme is described in [8].
- **I-TCP [5].** The connection is split into two parts at the last wired node along the path (which we refer to as the basestation). Loss recovery on the two parts proceeds independently. In our implementation, the wireless part uses a modified version of TCP SACK with fine grain timers.

We consider the behavior of these schemes in a typical cellular-style network, where wireless links are used to provide last hop connectivity to a set of nodes. In addition, we only consider the issue of data transfer to the mobile host.



(a) Throughput as a function of the packet drop rate (in percentage) (b) Comparison of the performance of I-TCP and Snoop

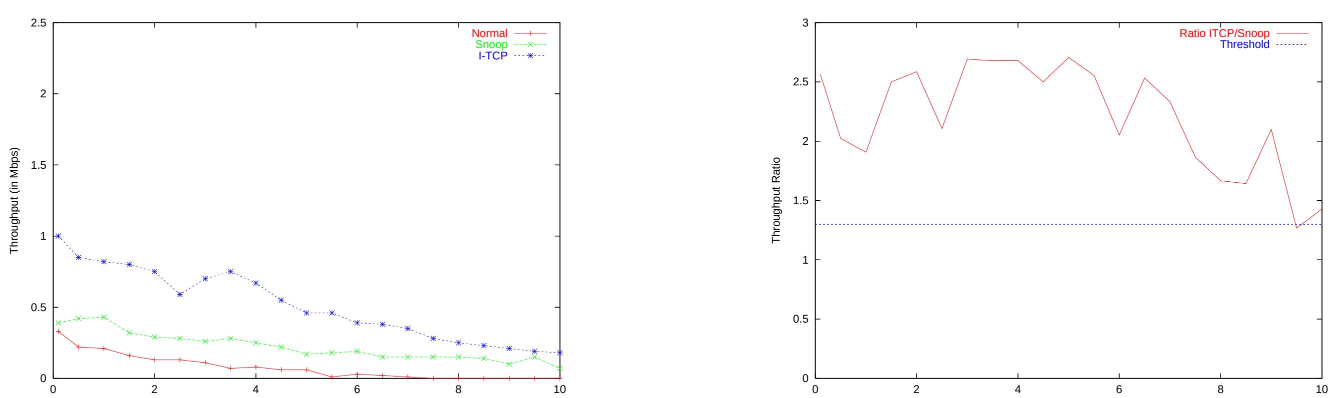
Figure 1: Plots showing the performance of the three schemes at high levels of connectivity



(a) Throughput as a function of the packet drop rate (in percentage) (b) Comparison of the performance of I-TCP and Snoop

Figure 2: Plots showing the performance of the three schemes at moderate levels of connectivity

Figure 1 illustrates some of the performance tradeoffs between the different techniques. In Figure 1(a), we plot the throughput of a long-lived TCP-SACK connection between a wired sender and a wireless receiver as a function of the packet drop rate for the three schemes listed above. These results were obtained using ns-2 [2] simulations. In Figure 1(b), we plot the ratio of the throughputs seen by I-TCP and Snoop. From these two figures, the following inferences can be drawn: (1) At very low packet drop rates ($< 1\%$), the throughputs achieved by the three schemes are similar. (2) At packet drop rates larger than 1%, the throughputs of Snoop and I-TCP are significantly larger than that of the Normal scheme. (3) The throughput of I-TCP is clearly better than that achieved by Snoop at all packet drop rates. However, the I-TCP approach suffers from several drawbacks like the hard state that it maintains, the computation and memory usage at the basestation and the non-compliance with end-to-end semantics (discussed in greater detail below). Nevertheless, we can consider I-TCP to be the right choice for the operating mode in spite of these disadvantages as long as the throughput that it achieves is at least a factor f of that achieved by Snoop. In our simulations, we used $f \approx 1.3$. Combining this fact with Figure 1(a), we can conclude that I-TCP is a good candidate mode when the packet drop rate is higher than 2% and Snoop is a good candidate for packet drop rates between 1% and 2%.



(a) Throughput as a function of the packet drop rate (in percentage) (b) Comparison of the performance of I-TCP and Snoop

Figure 3: Plots showing the performance of the three schemes at low levels of connectivity

In the previous set of graphs, the wireless node experiences very good signal strength from the basestation. We also conducted simulations in which the wireless nodes experienced moderate to high levels of disconnection. The results are shown in Figures 2(a), (b) and 3(a), (b).

In addition to performance differences in terms of the throughput achieved, the different schemes have important overhead and functionality tradeoffs. Unlike the `Normal` scheme, both the Snoop and I-TCP techniques require the basestation to buffer packets and perform local retransmissions. In terms of overhead, Snoop typically maintains less state than I-TCP, since the latter will have to keep some additional state associated with the split connections. An additional drawback of I-TCP is that its operation violates the end-to-end semantics of TCP. As a result, the buffers it maintains at the basestation are hard state, i.e. they are necessary for the correct reliability of the transfer. If the basestation fails or the buffers are lost, the end-to-end TCP transfer also fails. This violation of the fate-sharing principle [12] also complicates and slows down the procedure for the handoff between basestations [6]. A related benefit of I-TCP is that it completely isolates the sender from the behavior of the wireless link. This enables it to handle long disconnections of the link effectively [9] unlike the other schemes. Given these considerations, the Normal scheme is preferable to Snoop which is preferable to I-TCP as long as the performance of the schemes are similar.

Thus, it is clear that the Normal, Snoop and I-TCP modes of operation each make sense under different error rates, disconnection rates and delay conditions. The correct mode-to-condition mapping is summarized in Table 1.

Disconnection	**Packet Drop Rate**	**Best Operating Mode**
Very rare	Low	Normal
Very rare	Moderate	Snoop
Very rare	High	I-TCP
Somewhat frequent	Low/Moderate	Snoop
Somewhat frequent	High	I-TCP
Very frequent	Any	I-TCP

Table 1: Operating mode choice as a function of the disconnection rate and packet drop rate on the wireless last hop.

3.2 Estimating the Operating Conditions

In order for a wireless node to pick a mode (Table 1), the node will have to employ a scheme to estimate the loss rate, disconnection frequency, and last hop delay. Since the selection of mode is not highly sensitive to the actual conditions, the thresholds for selection in Table 1 are very coarse grain (i.e. low, moderate, high). As a result, the measurement techniques we incorporate are meant only to provide a very rough estimate for these parameters.

Many people have used low-level radio measurements such as signal-to-noise ratio (SNR) to predict the packet drop rate. Here, we are more interested in a longer-term estimate about the environment. This estimate is influenced by factors such as the mobile nodes location and movement rate, the interference present in the environment and the quality of the receiver interface. As a result, the estimate is typically unique to each mobile node. Therefore, we simply have each mobile host keep an average of its own observed packet drop rate over time. In practice, a node uses a CRC error at the link layer to identify the corruption of a packet. This corruption information is passed up to the node's transport layer, which keeps track of the packet drop rate and makes the final decisions about the operating mode.

While the packet drop rate is primarily related to factors unique to each node, the rate of disconnection is more dependent on the quality of overall signal coverage of a basestation. Therefore, we have each basestation measure and report the rate of disconnection within its region. Many wireless link technologies [15, 3] explicitly monitor the state of a link through the use of beacons, time synchronization signals, acknowledged messages, etc. Therefore, the basestation either can directly identify a disconnection event or be told about the disconnection at a later time by the mobile node. The basestation collects statistics about such events. When a mobile node first enters a cell, the basestation uses the history of these statistics to report typical disconnection patterns within its cell. The mobile node's transport layer can then use this information to make any mode switch decisions that are necessary.

3.3 Changing the Operating Mode

While the current conditions and the measurements of different operating modes may suggest that a different mode is desirable, actually switching between modes may be a challenge. If the actual switching introduces a high overhead, this may destroy any potential gains from the multi-modal approach.

In our design, the wireless end-node controls the selection of operating mode. When necessary, the mobile node provides information about its current operating mode, the desired next operating mode (if different from the current) and the connection ID of the transfer for which a mode shift is desired, to the basestation. Upon receiving a mode change request from an end-node, the basestation takes a set of actions based on the actual transition being performed.

In order to prevent frequent mode changes we also allow for some hysteresis. For example, as shown in Figure 1(b), we let the throughput achieved by I-TCP be at least 1.4 times better than that achieved by Snoop in order to switch from Snoop mode to I-TCP mode. However, we let the throughput ratio go lower than 1.2 when shifting mode from I-TCP to Snoop. In other words, we trigger the shift I-TCP→Snoop when loss rate is lower than 1% and the shift `Snoop`→I-TCP when the loss rate is higher than 4%.

Since there are three operating modes, six different mode changes are possible. We discuss the details of how the basestation implements each of these mode changes in turn.

`Normal → I-TCP` The basestation begins the Normal to I-TCP transition for a connection by recording the sequence number S of the next packet that arrives at the basestation for that connection. The basestation also allocates a buffer of size B for the connection and stores new packets that arrive. The basestation then waits for the instant of time T', when it receives a cumulative ACK from the wireless end-node indicating that S has been received at the destination. Between T and T', all acknowledgements from the wireless end-node are forwarded onward. However, the advertised window in any acknowledgment from the mobile host is modified to reflect the space available in the basestation buffer. After time T', the basestation starts sending ACKs for packets arriving from the sender and buffers them for transmission over the wireless last hop, without having to wait for the end-node to ACK them. At this instant of time, the basestation has changed the semantics of the connection from end-to-end to a split TCP connection.

`I-TCP → Normal` A basestation responds to this transition by sending a new ACK (in response to NEW data arriving from the original sender) that advertises a receiver buffer size of zero. This forces the sender into persist mode if there is additional data destined for the wireless end-node. The basestation continues to transmit data to the mobile-node until the basestation buffer is emptied (with requests for retransmissions all fulfilled). The last ACK from the wireless end-node, indicating that the buffer flush from the basestation was successfully completed, is forwarded unaltered to the sender. This ACK contains the actual receive buffer size at the wireless end-node forcing the sender out of persist mode. Note that since the wireless end-node decides to go into `Normal` mode only when the packet drop rate and the delay are low, the buffer flush phase should take a negligible amount of time, if any. In fact, the number of packets outstanding in the buffer is also likely to be low. Finally, upon exiting the persist mode, the sender can then resume normal operation. In addition, after the buffer flush is complete, the basestation will remove all the state associated with the connection.

`Normal → Snoop` Since there is no change in the semantics of the TCP connection, shifting modes from `Normal` mode to the Snoop mode is relatively easy. Essentially, as soon as a request for a change in mode is received, the basestation starts caching packets destined for the wireless end-node and monitors the ACKs being sent to the sender. Any *dupacks* which can be satisfied locally result in retransmission from the cache. All other *dupacks* are forwarded to the sender.

`Snoop → Normal` Just as in the previous case, as soon as the request is received the basestation simply stops to check if any *dupacks* can be locally satisfied and blindly forwards all ACKs to the sender. In addition, the basestation will also erase all state associated with the connection.

`Snoop → I-TCP` This is similar to the mode shift from Normal mode to I-TCP mode except that requests for retransmissions are served both the Snoop cache as well as the I-TCP buffer until the TCP connection is split.

`I-TCP → Snoop` Again, this shift in mode is similar to the mode change `I-TCP → Normal` except that the basestation employs Snoop functionality after the buffer has been flushed instead of blindly forwarding all ACKs to the sender.

Note that during each of these transitions, the performance of the end-to-end TCP connection is affected minimally, if at all. In addition, we assume that the wireless end-node makes a decision to shift its operating mode only after a threshold number of packets have been received on a connection. In general, irrespective of the operating conditions, all short connections will be operated in the Normal mode.

3.4 Simulation Results

We use an ns-2 [2] based simulation to compare and contrast the performance of our multi-modal protocol with the three schemes outlined in Section 3.1. We simulate these protocols in an environment with variable operating conditions. In our simulations, we have a single mobile wireless node involved in a long-lived TCP transfer from a wired sender via basestation node(s). Unless otherwise specified, we use TCP-SACK on both the wired and the wireless hops of the links. As mentioned in at the beginning of Section 3.1, we use finer grained timers for the TCP connection on the wireless hop when the connection is in the `I-TCP` mode.

The total simulation time is 200s. At regular intervals of Ts, the mobile wireless node is handed off to a new basestation with a possibly different quality of coverage (or rate of disconnection). We use three possible types of coverage: good (very rare disconnection), medium (somewhat frequent disconnection) and bad (very frequent disconnection) as mentioned in Table 2. When a hand-off occurs in our simulation, the packet loss percentage experienced by the wireless node is assigned to a random value in the interval $[0, 10]$. In addition, we assume that in the period between consecutive hand-offs, the packet drop rate experienced by the wireless node varies in a smooth manner around the value at the beginning of the period. In addition, the wireless node goes into states of disconnection and re-connection with probabilities as determined by the type of coverage in the current cell. This choice of probabilities is discussed below.

We show the results obtained for an example situation where we choose $T = 25$s. In addition, when disconnections are rare (good coverage), we assume that the conditional probability of disconnection, given that the wireless node is in connected state, is low ($\rho_{cd}^{good} = 0.05$). In addition, the conditional probability of moving to a connected state given that the node is in disconnected state is high ($\rho_{dc}^{good} = 0.95$). Following the same notation for probabilities, we have $\rho_{cc}^{good} = 0.95$ and $\rho_{dd}^{good} = 0.05$. Similarly, for medium coverage, we set $\rho_{cc}^{medium} = 0.8$ and $\rho_{dd}^{medium} = 0.1$. For bad coverage, we set $\rho_{cc}^{bad} = 0.5$ and $\rho_{dd}^{bad} = 0.1$. When the wireless node moves into the disconnected state, we assume that it stays disconnected for 1s on average. For the multi-modal case, the TCP connection starts out in the Normal mode at time 0. Also, at time 0, the wireless node starts in a cell with good coverage and with low packet drop rate. For the sake of clarity, we summarize the operating conditions and the operating mode changes as a function of time in Table 2. In this simulation, we do not model the hysteresis of our multi-modal protocol. Since mode changes are not too frequent, we believe that this does not significantly impact the results.

Time(s)	Coverage	Packet Drop Rate	Operating Mode
0.0	Good	Low	Normal
25.0	Good	Moderate	Snoop
50.0	Good	High	I-TCP
75.0	Bad	High	I-TCP
100.0	Bad	Low	I-TCP
125.0	Good	Low	Normal
150.0	Medium	Moderate	Snoop
175.0	Medium	Moderate	Snoop

Table 2: Summary of simulation scenario

Figure 4 shows the results of our simulation. We plot the throughput achieved by our multi-modal scheme as well as the three schemes discussed above as a function of time. Note that the performance of multi-modal TCP is better than that of Snoop and Normal. However, the throughput of multi-modal TCP is lower than that obtained by using I-TCP. However, as we argue in Section 3.1, using I-TCP, is not always desirable. Our multi-modal reliability protocol, chooses I-TCP only when the packet drop rate and disconnection probabilities warrant it. For example, our protocol decides to stay in the Snoop mode until time 50s. However, beyond 50s, I-TCP is likely to offer a much higher performance than Snoop given the way the operating conditions change. Hence, our protocol shifts mode to I-TCP mode. Note that the way operating modes are chosen in our protocol ensures that the performance is never lower than a fixed fraction of that of I-TCP on small time scales.

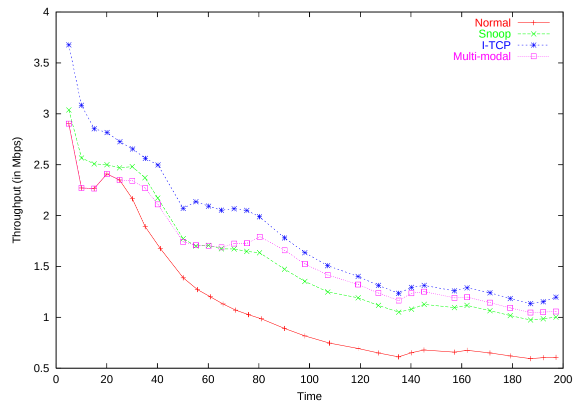


Figure 4: Plots showing the performance of the three schemes at moderate levels of connectivity

4 Multi-modal Ad-hoc Routing

Over the past decade, a wide variety of ad-hoc routing protocols [27, 19, 25, 26] have been proposed. Most of these protocols (e.g., DSDV, DSR) are unstructured: they do not impose any sense of hierarchy on the underlying network of ad-hoc nodes. The unstructured nature of these algorithms typically results in poor scaling to large network sizes. At such extremes, the overhead due to routing is high and communication becomes inefficient. Some schemes for ad-hoc routing (e.g., [13, 10]) use a loose sense of hierarchy. These schemes have been shown to perform well for large network sizes. Clearly, in a setting where the population of ad-hoc nodes is dynamic, a protocol with a sense of the population size and the ability to operate at different modes depending on the estimated size is desirable. In this section, we present an example of such a multi-modal routing protocol for ad-hoc networks.

4.1 Modes of Operation

Based on the tradeoffs in ad-hoc routing we consider two different modes of operation: unstructured and hierarchical. For the generic unstructured protocol, we use a variant of DSDV. For the hierarchical protocol, we must provide a light-weight mechanism to maintain a heirarchy over the ad-hoc nodes. One way of creating such a hierarchy is to employ a few *location-aware nodes* that know their positions via GPS or other location service systems. In the following subsections, we describe the details of the unstructured DSDV mode and the hierarchichal Proxy mode. We then describe the trade-offs between these two modes.

4.1.1 The DSDV Mode

In the DSDV mode, all nodes in the network use the DSDV-like protocol, as described in [27], for routing. This mode handles node arrivals and departures as follows. Node departures are detected by the absence of routing updates from the departed nodes. Information about a departure is propagated in subsequent updates. A newly arriving node inserts itself into the network by simply listening to the distance vector announcements from the other nodes in its radio range and participating in the subsequent route updates. One important difference from regular DSDV is that a special *proxy bit* is used in route updates and in routing tables to indicate the proxy nodes. The actual routing itself is completely oblivious to the existence of proxy nodes. In other words, proxy nodes are not treated specially in this mode. This change is necessary to support the efficient transition from DSDV to Proxy mode.

4.1.2 The Proxy Mode

In a network with a medley of location-aware and location-unaware nodes, one scalable routing approach is *geographical forwarding with location proxies* [13]. This technique uses an adaptive local routing protocol to extend geographic forwarding techniques [14, 21] to location-unaware nodes. In this scheme, a location-unaware node selects a nearby location-aware node as its proxy. To receive packets, the location-unaware node advertises the location of its proxy as its own location. Packets in the network are forwarded between proxy nodes using geographic forwarding and from a proxy node to the corresponding destination using a local routing protocol like DSDV. When originating or forwarding data, all nodes use their local neighbor tables to determine the best next hop for geographic forwarding.

Our Proxy mode is modified version of the protocol described in [13]. In Proxy mode, the following facts hold true for the routing tables in the network:

I Each proxy node knows the route to its *neighbor* proxy nodes, where two proxies **P** and **P′** are called to be neighbors if there are two nodes **X** and **Y** one hop away from each other such that **X** is registered to **P** and **Y** is registered to **P′** (a proxy is assumed to be registered to itself). Each of the proxies can communicate with the other proxies (may be through non-proxy nodes). Thus, the proxy nodes form a connected overlay structure among themselves.

II Each proxy knows the route to all of the non-proxy nodes registered to it.

III Each node knows the route to its proxy, some non-proxies and possibly some proxies that are neighbors to its own proxy.

To keep the above facts about routing tables accurate, the protocol must carefully handle the addition and removal of nodes in the network and the processing of routing messages.

Addition of a new node. When a new node **A** joins, the following steps are taken:

1. Node **A** broadcasts a ProxySearch message with a `ttl` of 1. If the network is operating in DSDV mode, neighbors (i.e., nodes in the radio-range of **A**) reply back with their routing messages and **A** starts operating in DSDV mode. Otherwise, each of the neighbors responds to this message with a ProxyReply message that contains the address of its own proxy and its distance from that proxy. Node **A** then selects a neighbor **B** that has the minimum distance (hop count) from its proxy **P**. **A** then registers to the proxy **P** by sending a Register message to it. Note that **A** knows the route to its new proxy **P** from its neighbor **B**.

2. Node **B** forwards a routing table entry for **A** by a hop-limited flood with initial hopcount k, where k is the distance from **A** to **P** via **B**. This allows a path to be established from node **P** to node **A**, so that a packet addressed to **A** can be delivered from **P**.

3. Node **A** then sends a NewNode message to all its neighbors. The message contains the address of the proxy **P** and its distance from **A**. If a neighbor finds that its distance to **P** via **A** is less than its current proxy's distance, it switches its proxy by sending a Unregister message to its old proxy and a Register message to **P**. When a node switches to a new better proxy, it forwards the NewNode message to its neighbors, otherwise it drops the message. This phase allows existing nodes to register to a nearer proxy if it is reachable via a newly joined node.

Note that if the newly joined node **A** is a proxy, it only needs to perform Step 3. The proxy node **A** also announces the fact that it is location-aware in its NewNode message. This announcement could lead to a series of updates resulting in some location-unaware nodes choosing node **A** as a proxy (in a manner similar to that outlined in Step 3).

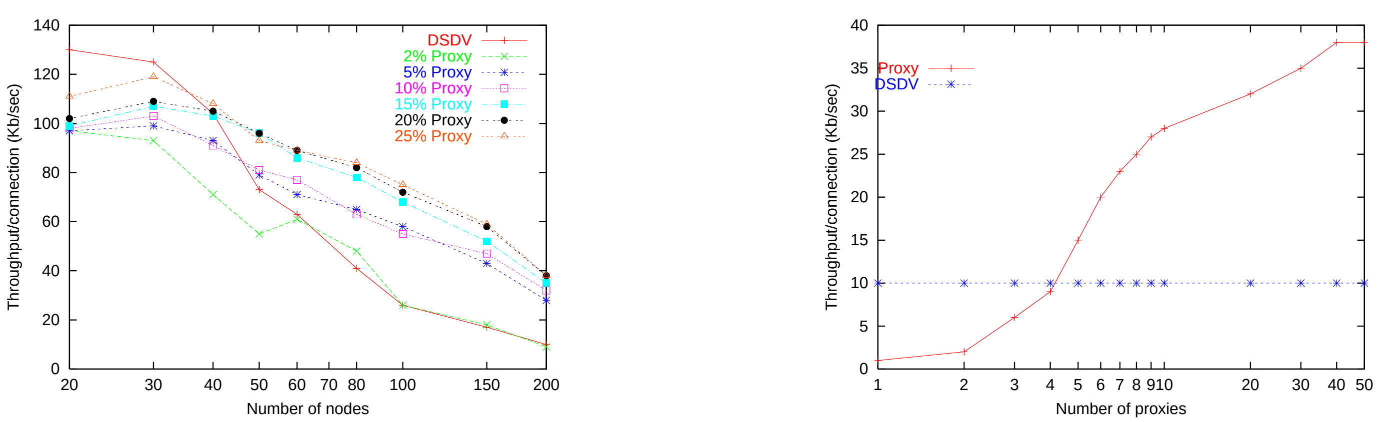
Removal of an existing node. When a node (proxy or non-proxy) departs the network, its neighbors detect the event from the lack of route updates (alternately, we could have used more frequent, periodic HeartBeat messages). The local DSDV protocol propagates this departure event as part of its route table updates using hop-limited flooding. During this update, each location-unaware node tries to maintain connectivity with its current proxy by trying to choose alternate routes. If a location-unaware node detects that it no longer has a route to its proxy, it first notifies its neighbors of the event and then tries to join the network in the way described earlier.

Routing message processing. A routing table entry for a destination node **N** is propagated through the network only to k hops, where k is the distance in hopcount from **N** to its proxy **P**. Due to this hop-limited flooding of routing table entries, a non-proxy node **N** knows the next hops to reach only the nodes $\mathbf{N}'$ with proxy $\mathbf{P}'$ if $\texttt{hopcount}(\mathbf{N}',\ \mathbf{N}) \leq \texttt{hopcount}(\mathbf{N}',\ \mathbf{P}')$, where $\texttt{hopcount}(x,\ y)$ denotes the number of hops needed to reach y from x. This condition ensures that each node can be reached from its proxy. However, in our forwarding technique, we assume that proxies maintain information about their neighboring proxies and can communicate with them, and thus form a connected overlay. This crucial condition is ensured as follows. When a location-unaware node **N** finds (during the process of joining the network or later when it gets a new neighbor) that its neighbor has a proxy $\mathbf{P}'$ not present in its routing table, it adds a route to $\mathbf{P}'$ in its routing table. In addition, it announces its route to $\mathbf{P}'$ in a flood that is hop-limited by the hopcount between **N** and its own proxy **P**. This results in proxy **P** adding a routing table entry to reach $\mathbf{P}'$ via **N**. When done throughout the network, this ensures that each proxy is connected to the proxies of its neighboring clusters.

Data forwarding. As an end result of the routing message exchanges, a non-proxy node maintains a route to reach a set of nearby nodes, its own proxy, and possibly routes to some other nearby proxies. The routes to other proxies are only needed to make sure that the proxy nodes are in fact connected to each other. A proxy node's routing table contains routes to each of the non-proxy nodes registered to it, as well as all of its neighbor proxies. Routes to these neighbor proxies may go through non-proxy nodes.

Each non-proxy node advertises its proxy's location as its own location. A location service (e.g. GLS) is used to keep track of the current location of a node given its node ID. To route a packet to a destination **D**, a node first searches its routing table to find a next hop entry to reach **D**. If there is an entry for **D**, the packet will be forwarded to the corresponding next hop (similar to the DSDV routing). Otherwise, only the nodes whose locations are known in the routing table will be considered and the packet will be forwarded to the node that is geographically nearest to the destination (similar to geographic forwarding). We assume here that the geographical forwarding avoids holes and always delivers the packet to destination. This can be done using techniques such as those in [20, 21].

4.1.3 Tradeoffs Between Modes



(a) Throughput at different population sizes and proxy densities (b) Throughput as a function of proxy density (population size = 200)

Figure 5: Plots showing the throughput of the DSDV and proxy protocol at different population sizes and proxy densities.

Consider the behavior of our unstructured ad-hoc routing protocol, DSDV mode, in a network of N ad-hoc nodes. For small values of N, this protocols incurs very little overhead. However, at higher values of N, the size and overhead of the routing messages becomes prohibitive. Similarly, the performance of our structured protocol, Proxy mode, is sensitive to the number of location-aware nodes, N_{aware}. When N_{aware} is comparable to the total number of nodes, N, the system is very efficient. However, when N_{aware} is small compared to N, the hierarchical scheme announces routing information widely and forces data to be routed circuitously through the network. Based on these considerations, we use the value of N and N_{aware} to determine when each mode is appropriate.

In order to determine the thresholds for choosing different modes, we perform a set simulations in ns-2 with the CMU wireless extensions [1]. In these simulations, we vary the total number of ad-hoc nodes, N, and the percentage of nodes that are aware of their geographic position, $\frac{N_{\text{aware}}}{N}$. These simulations were performed for both DSDV and Proxy modes (with varying percentages of location-aware nodes). The geographic forwarding simulation uses a perfect location service that allows a node to lookup another node's most recently advertised location at any time. Some of the other key parameters of the simulation are listed in Table 3.

In each simulation run, we start a few (0.5 * N) TCP connections between randomly chosen pairs from amongst the N nodes. We start some (.25 * N) background CBR flows between the ad-hoc nodes. We measure the average throughput attained by the TCP connections as a function of the values of N and N_{aware}. The average throughput seen by a TCP connection in this population encapsulates a few key factors: the efficiency of routes taken (via the RTT and the drop rate) and the correctness of routes computed (via the drop rate). Note that we do not consider the routing overhead, since some of the traditional schemes like DSDV are likely to always look worse with respect to overhead.

In Figure 5(a), we show the average TCP connection throughput as a function of N. We note that, as expected, the Proxy mode does better that DSDV for larger values of N. For a given percentage of location-aware nodes, we can determine the point at which we should switch modes by locating the intersection between the DSDV and the appropriate Proxy line. For example, when 25% of the nodes are location-aware and there are less than 35 nodes, the network should use DSDV mode. Otherwise, the Proxy mode should be used. In Figure 5(b), we show the impact of change N_{aware} for a fixed value of $N = 200$. We see that the performance of the Proxy mode dips dramatically as we remove location-aware nodes from the system. We can determine the threshold for switching to Proxy mode in a network of a particular size by looking at the intersection point on this graph. For example, for a network of size 200, we need at least 11 location-aware nodes to provide better performance than DSDV.

Our multi-modal routing protocol is configured with the results from these graphs. With this information, a node simply uses the estimates of the population size and the density of location proxies to choose the right mode. In practice, we must avoid having the system repeatedly switch between these modes if it is near this threshold. Therefore, we simply add hysteresis to the threshold values.

4.2 Estimating the Operating Condition

The two critical factors that must be estimated for the multi-modal routing protocol are the total number of nodes and the density of location-aware nodes. In addition, these estimates must be available in both modes of operation.

In DSDV mode, each node maintains a forwarding table with a full list of possible destinations. This table is filled using the periodic routing message exchange that is done as part of the protocol. Our modifications to normal DSDV also add information about whether a node is location-aware or not. As a result, getting population estimates and location-aware node density is trivial in DSDV mode.

Unfortunately, since routing information does not flood the whole network in the Proxy mode, the DSDV estimation mechanism cannot be used. To provide the necessary estimates, the location-aware nodes form a spanning tree rooted at the location-aware node with the lowest ID. This lowest ID node initiates the formation of this spanning tree after the mode is switched from DSDV to Proxy mode. We believe that many of the proxies may be part of a deployed infrastructure and would not leave the system. If the departure rate of these nodes is rare, then maintaining the spanning tree is inexpensive. When a new location-aware node joins the system, it attaches itself to the spanning tree via one of its neighbor proxies. The root of the current spanning tree elects one of its child proxies as the *potential root*. If a non-root location-aware node leaves the system, each of its children can attach itself to the spanning tree again via another of its neighbor proxies. If the root node leaves the system, its neighbors detect the event and the potential root

Parameter	Value
MAC	IEEE 802.11
Nominal range	250 meters
Radio rapacity	2 Mbps
Movement model	Random waypoint
Pause time	30 sec
Maximum speed	5 m/s
Triggered update period	1 sec
Periodic route update interval	15 sec
Physical region size	670 x 670 meters

Table 3: Parameters used in the simulations

node initiates the formation of a new spanning tree and becomes a new root.

The population size and the density of location-aware nodes are collected along the spanning tree. Each proxy announces the count of location-unaware nodes that are registered with it up the spanning tree. Since a location-unaware node is registered exactly to one proxy node, the root of the tree can use the announced counts to estimate the total number of location-unaware nodes and number of proxies. The root uses this estimate to make the decision about switching to DSDV mode. If the root decides to switch to DSDV mode, it broadcasts this decision, and the switch to DSDV mode takes place.

4.3 Changing the Operating Mode

In many ways, switching modes in a routing protocol is much more difficult than switching modes for reliability. The key problem is coordinating the switch by many nodes and ensuring some compatibility between the modes.

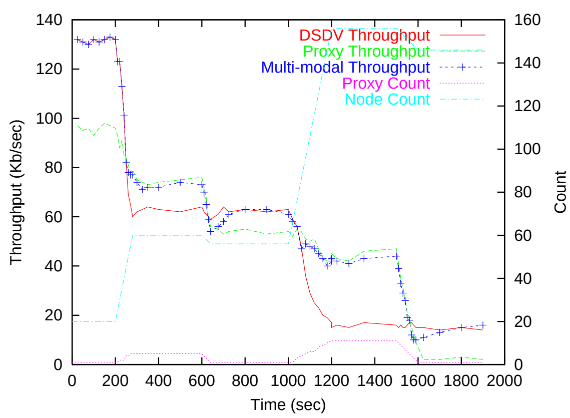
In Proxy mode, we place a number of requirements on what the routing tables of different nodes must contain. Fortunately, the information kept in the DSDV routing table is actually a superset of this information. Therefore, the transition from DSDV to Proxy mode can be handled relatively smoothly. Any node that identifies that the appropriate conditions exists begins to route using Proxy mode. However, all the nodes may not have the same consistent view of the system. So it is possible that some of the nodes decide to switch to the Proxy mode while others remain in the DSDV mode. To alleviate this problem, each node deciding to switch to proxy mode piggybacks the decision in its next routing update, so that a neighbor node can switch to Proxy mode if necessary.

The transition from Proxy mode to DSDV mode is more complicated. The root of the estimation spanning tree initiates this transition by broadcasting the switch to all the nodes in the system. Due to the unreliable nature of broadcast, some nodes may not know about the mode switch and continue operating in Proxy mode. As before, nodes deciding to switch mode piggyback their current mode in their next updates. This allows neighbors to switch mode if necessary. However, even after this switch, it takes time for each node to build a routing table containing routes to all the existing nodes in the system. During this transition phase, when the routing tables are still incomplete, Proxy nodes work as the default gateway and use geographic forwarding to route packets. However, as the routing tables become more and more complete, routing will switch over to using DSDV mode.

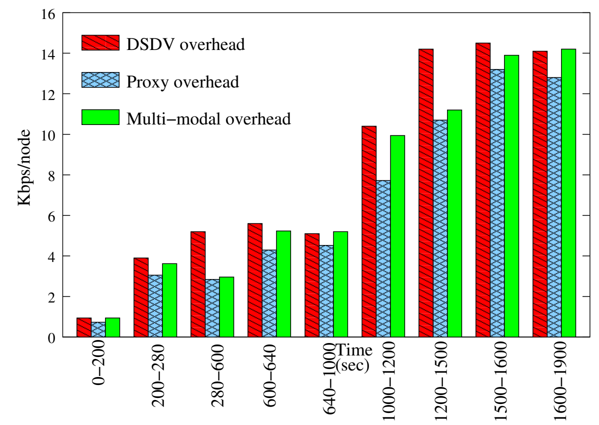
4.4 Simulation Results

To evaluate the performance of our multi-modal routing protocol, we simulate its behavior in a dynamic environment. Our simulation setup is similar to the one described in Section 4.1.3. One important change is that the number of total nodes, N, and number of location-aware nodes, N_{aware}, vary as follows within a single experiment. At time $t = 0$s, the scenario contains 20 nodes of which one is location-aware node. At $t = 200$s, 40 nodes start joining the network at an average rate of 0.5 nodes/second. The probability of each of these new nodes being location-aware is 0.10. At $t = 600$s, all but one proxies start leaving the network, with an average rate of 0.1 nodes/second. At time $t = 1000$s second, 100 new nodes (with probability 0.10

of being location-aware) start joining the network at an average rate of 0.5 nodes/second. At $t = 1500$s, all but one proxies start leaving the network, with an average rate of 0.1 nodes/second. Finally, the simulation ends at $t = 1900$s. When the number of nodes in the network changes, new TCP and CBR connections are initiated or old connections are terminated to maintain the fixed ratio between connections and nodes.



(a) Throughput in different operating conditions



(b) Routing overhead in different operating conditions

Figure 6: Plots showing the performance and overhead of the three schemes in a dynamic environment

In this dynamic environment, we simulate the DSDV-only, the Proxy-only and multi-modal protocols. Figure 6(a) shows how the average TCP connection throughput varies during this experiment. The performance of TCP in the DSDV and Proxy modes varies exactly as described in Section 4.1.3. The performance of the multi-modal protocol tracks closely to the better of DSDV-only and Proxy-only. This indicates that it is performing the desired switching between nodes. This switch occurs each time the DSDV-only and Proxy-only mode plots intersect. During a short period after the switch from Proxy to DSDV mode (at $t = 600$s and $t = 1600$s), we see that the multi-modal system takes some time to match the DSDV-only's performance. This is because of the characteristics of this transition, described in Section 4.3. Also, in comparison with the Proxy-only protocol, the multi-modal protocol in Proxy mode incurs some extra overhead to maintain the estimation spanning tree. This overhead results in the multi-modal protocol in Proxy mode having slightly lower throughput than the Proxy-only protocol (e.g., from $t = 280$s to $t = 600$s).

Figure 6(b) shows the routing overhead incurred by the protocols during different periods of our simulation. During periods with a stable set (t = 0-200s, 280-260s, 640-1000s, 1200-1500s, and 1600-1900s), the overhead of the DSDV-only protocol is roughly proportional to the population size. Since the minimum triggered update time is set to 1 second, at most one routing update packet is generated per node per second. However, the size of the routing update packet depends on the network size and mobility of the nodes. Overhead for the Proxy-only protocol is less, because of the limited flooding of routing table entries. The overhead of multi-modal protocol depends on which mode it is operating in. When it is in DSDV mode, the overhead is almost the same as the DSDV-only protocol. When it is in Proxy mode, the overhead is slightly higher than the Proxy-only protocol because of the maintenance of the estimation spanning tree. Routing overhead of all the protocols increases when nodes are joining or leaving (t = 200-280s, 600-640s, 1000-1200s, 1500-1600s). In DSDV, the increase is insensitive to whether the joining/leaving node is location-aware or not. However, as can be seen during the t = 600-640s period, routing overhead increases dramatically for the Proxy-only protocol when location-aware nodes leave. In all cases, the overhead of the multimodal protocol is in between the overheads of the other two protocols.

Overall, the multi-modal protocol is able to adapt to a much wider set of conditions than either the DSDV-only or Proxy-only protocol. In addition, it incurs only a minor performance penalty over the best scheme at any time. Finally, the multi-modal routing system does not add significant additional overhead to the system.

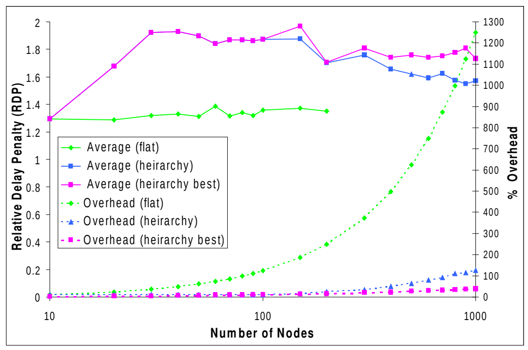


Figure 7: Performance Evaluation of Flat and Heirarchical Schemes for End-System Multicast. The graph plots the relative delay penalty, a measure of how sub-optimal the overlay paths are compared to the shortest path and the overhead in terms of bytes/second. The flat scheme is compared with the heirarchical organization output by the simulator and with theoretically obtained optimal heirarchical organization.

5 Other Instances of Multi-Modal Protocols

In this paper, we have shown two examples of multi-modal protocols. We believe that the general approach employed in designing the protocols presented in this paper can be extended to a variety of other situations and protocols. In this section, we present some of our preliminary thoughts on applying multi-modal design to overlay routing and power management.

5.1 Overlay Routing: End-System Multicast

An interesting example of a situation that could benefit from the multi-modal protocol design philosophy is the End-System Multicast protocol, Narada [11]. Narada builds a highly connected overlay (mesh) of the end-systems participating in multicast. A spanning tree, that would serve as the multicast tree, is built on top of this mesh. Narada performs expensive management activities like probing for available bandwidth, delay and possibility of tree reorganization over the mesh. However, the highly connected *flat* organization of the mesh, and the resulting overhead due to management and probing, renders it unscalable beyond a few nodes. In [18], the authors propose a heirarchical extension to Narada which maintains the mesh as a two level heirarchy of the end-systems. This scheme is shown to scale well for a large number of nodes. However, when the number of nodes is smaller, the overhead of maintaining a hierarchy far exceeds that of maintaining a flat highly connected mesh, as shown in Figure 7 which plots the result of our simulations with the Narada simulator and the simulator from [18]. For small group sizes, the overheads due to the two schemes are identical. However, the flat scheme out-performs the heirarchical scheme in terms of delay (the latency between end nodes in the overlay). However, for larger group sizes, the overhead from flat schemes is orders of magnitude higher than that due to heirarchical schemes. Thus, though the delay is higher, heirarchical schemes are likely to yield much better performance (in terms of throughput) at larger group sizes. We are currently investigating schemes that help End-Sytem Multicast estimate the group size and switch between flat and heirarchical organizations respectively.

5.2 Power Management

Current systems support two forms of power management at a very high level: PM-OFF and PM-ON. These modes are statically configured by the end-user. The PM-ON mode allows the network interface card of the host to sleep and wake up at deterministic times to receive or send packets. In the PM-OFF mode, the network interface of the node has to remain powered on throughout. Clearly, the PM-ON does result in substantial savings in power. However, these savings come at the expense of performance. This is because, when in PM-ON mode, the end-node is forced to synchronize with the basestation (which buffers its packets) and receive data at fixed intervals, which could be quite large. If the last hop network (typically wireless) is the bottleneck, then this could affect the throughput achieved by large tranfers. While the PM-ON mode has little effect on observed performance at the end-node for long transfers, the PM-ON mode is clearly not well-suited for interactive applications or short transfers. One could imagine designing a multi-modal power management scheme, which uses application or traffic demands to dynamically decide the choice of the power management mode to operate in.

In addition, there is another mode of operation, which we believe, could be interesting: the predictive mode. In the predictive mode, the end-node monitors the arrival patterns of packets it receives and uses this to estimate when it would next receive a packet. If the inter-packet spacing is reasonably uniform and large, then the end-node could benefit by predicting this gap and turning its interface card to sleep mode during these intervals (if this gap is sufficiently large). This scheme is well-suited to short transfers which are usually in slow-start during their entire lifetime since the inter-arrival times for packets (bursts of packets, to be precise) when the connection is in slow-start are much larger than that in steady state. Since using this scheme allows the end-node to by-pass synchronization with the basestation, this scheme is also well suited for interactive applications. However, in situations in which the traffic patterns are not smoothly varying and predictable, this scheme will not be beneficial due to errors caused by wrong predictions. This scheme is also not likely to yield any additional benefits for long duration, high-bandwidth transfers, as the power savings during slow-start are minimal and the inter-packet spacing is very low. Given the characteristics of this predictive scheme, we believe it can play an important role in a multi-modal power management protocol. A multi-modal protocol that uses these three modes should be able to provide good performance and power-savings under a variety of workloads and conditions.

6 Summary

In this paper, we have presented a case for multi-modal protocols, protocols which dynamically change their operating mode depending on the environmental conditions. Our aim was to study the general challenges in building and the potential benefits of using such protocols.

Using two examples – multi-modal reliable transport and multi-modal ad-hoc routing – we have shown how the operating mode decisions can be made by analyzing the trade-offs between the various possible operating modes. In addition, we have also shown how the environmental conditions can be measured accurately enough to guide the operating mode decision process. Finally, we have shown how these protocols can be designed so as to incur only a small over-head during a change in operating mode. Our protocols are robust to changing operational conditions and exhibit good overall performance in comparison with exiting protocols. All our results are based on simulation experiments and we have yet to analyze the benefits of our design in a real world scenario.

We conclude that multi-modal protocols are both necessary and useful for operation in highly variable operating conditions, as is typical of most mobile and wireless networks. We believe that our work will influence the general design methodology for protocols in the mobile world to incorporate the ability to monitor operating conditions and switch operating modes in a light-weight manner, when necessary.

References

[1] The cmu monarch group home page. http://www.monarch.cs.cmu.edu.

[2] The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[3] Ieee std. 802-11, ieee standard for wireless lan medium access control (mac) and physical layer (phy) specification.

[4] Allman, M., Paxson, V., and Stevens, W. TCP congestion control. Internet Draft, Internet Engineering Task Force, Feb. 1999. Work in progress.

[5] Bakre, A. V., and Badrinath, B. R. I-TCP: Indirect TCP for mobile hosts. In *International Conference on Distributed Computing Systems* (1995), pp. 136–143.

[6] Bakre, R. B. A. Handoff and system support for Indirect TCP/IP. *Proceedings of the 2nd USENIX Symposium on Mobile and Location-Independent Computing* (1995).

[7] Balakrishnan, H., Padmanabhan, V. N., Seshan, S., and Katz, R. H. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking 5*, 6 (Dec. 1997), 756–769.

[8] Balakrishnan, H., Seshan, S., and Katz, R. H. Improving reliable transport and handoff performance in cellular wireless networks. *ACM Wireless Networks 1*, 4 (1995).

[9] Brown, K., and Singh, S. M-TCP: TCP for mobile cellular networks. *ACM Computer Communication Review 27*, 5 (1997).

[10] Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, 2002.

[11] Chu, Y.-H., Rao, S. G., and Zhang, H. A case for end system multicast. In *ACM SIGMETRICS 2000* (Santa Clara, CA, June 2000), ACM, pp. 1–12.

[12] Clark, D. D., and Tennenhouse, D. L. Architectural considerations for a new generation of protocols. In *SIGCOMM Symposium on Communications Architectures and Protocols* (Philadelphia, Pennsylvania, Sept. 1990), IEEE, pp. 200–208. Computer Communications Review, Vol. 20(4), Sept. 1990.

[13] Couto, D. D., and Morris, R. Location proxies and intermediate node forwarding for practical geographic forwarding. Tech. Rep. MIT-LCS-TR-824, MIT Laboratory for Computer Science, 2001.

[14] Finn, G. G. Routing and addressing problems in large metropolitan scale internetworks. Tech. Rep. SRNTN 44, USC-Information Sciences Institute, Marina del Ray, CA, July 1986.

[15] Group, B. S. I. Specifications of the bluetooth system.

[16] Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. Energy-efficient communication protocols for wireless microsensor networks. *Proc. Hawaaian Int'l Conf. on Systems Science* (jan 2000).

[17] Jacobson, V. Congestion avoidance and control. *ACM Computer Communication Review 18*, 4 (Aug. 1988), 314–329. Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988.

[18] Jain, S., Mahajan, R., and Niswonger, B. Self-organizing overlays. Tech. rep., University of Washington.

[19] Johnson, D. Routing in ad hoc networks of mobile hosts. In *Workshop on Mobile Computing Systems and Applications* (Santa Cruz, CA, U.S., 1994).

[20] Karp, B., and Kung, H. T. GPSR: greedy perimeter stateless routing for wireless networks. In *Mobile Computing and Networking* (2000), pp. 243–254.

[21] Li, J., Jannotti, J., De Couto, D., Karger, D., and Morris, R. A scalable location service for geographic ad-hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)* (Aug. 2000), pp. 120–130.

[22] Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, J. E., Flinn, J., and Walker, K. R. Agile application-aware adaptation for mobility. In *Proc. of the 16th ACM Symposium on Operating Systems and Principles* (Saint-Malo, France, Oct. 1997).

[23] Padhye, J., Firoiu, V., Towsley, D. F., and Kurose, J. F. Modeling TCP reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking 8*, 2 (Apr. 2000), 133–145.

[24] Padmanabhan, V., and Sripanidkulchai, K. The case for cooperative networks. *First International Workshop on Peer-to-Peer Systems* (2002).

[25] Park, V. D., and Corson, M. S. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM (3)* (1997), pp. 1405–1413.

[26] Perkins, C., Royer, E., and Das, S. Ad hoc on demand distance vector (AODV) routing. Internet Draft, Internet Engineering Task Force, Jan. 2002. Work in progress.

[27] Perkins, C. E., and Bhagwat, P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM Computer Communication Review 24*, 4 (Oct. 1994), 234–244. SIGCOMM '94 Symposium.