Appearing in 1st IEEE International Workshop on Sensor Net Protocols and Applications (SNPA). Anchorage, Alaska, USA. May 11, 2003.

RMST: Reliable Data Transport in Sensor Networks [1]

Fred Stann, John Heidemann

Abstract – **Reliable data transport in wireless sensor networks is a multifaceted problem influenced by the physical, MAC, network, and transport layers. Because sensor networks are subject to strict resource constraints and are deployed by single organizations, they encourage revisiting traditional layering and are less bound by standardized placement of services such as reliability. This paper presents analysis and experiments resulting in specific recommendations for implementing reliable data transport in sensor nets. To explore reliability at the transport layer, we present RMST (Reliable Multi-Segment Transport), a new transport layer for Directed Diffusion. RMST provides guaranteed delivery and fragmentation/reassembly for applications that require them. RMST is a selective NACK-based protocol that can be configured for in-network caching and repair.**

1 Introduction

Wireless sensor networks provide an economical, fully distributed, sensing and computing solution for environments where conventional networks are impractical. This paper explores the design decisions related to providing reliable data transport in sensor nets. The reliable data transport problem in sensor nets is multi-faceted. The emphasis on energy conservation in sensor nets implies that poor paths should not be artificially bolstered via mechanisms such as MAC layer ARQ during route discovery and path selection [1]. Path maintenance, on the other hand, benefits from well-engineered recovery either at the MAC layer or the transport layer, or both. Recovery should not be costly however, since many applications in sensor nets are impervious to occasional packet loss, relying on the regular delivery of coarse-grained event descriptions. Other applications require loss detection and repair. These aspects of reliable data transport include the provision of guaranteed delivery and fragmentation/ reassembly of data entities larger than the network MTU.

Sensor networks have different constraints than traditional wired nets. First, energy constraints are paramount in sensor networks since nodes can often not be recharged, so any wasted energy shortens their useful lifetime [2]. Second, these energy constraints, plus relatively low wireless bandwidths, make in-network processing both feasible and desirable [3]. Third, because nodes in sensor networks are usually collaborating towards a common task, rather than representing independent users, optimization of the shared network focuses on throughput rather than fairness. Finally, because sensor networks are often deployed by a single organization with inexpensive hardware, there is less need for interoperability with existing standards. For all of these reasons, sensor networks provide an environment that encourages rethinking the structure of traditional communications protocols.

The main contribution is an evaluation of the placement of reliability for data transport at different levels of the protocol stack. We consider implementing reliability in the MAC, transport layer, application, and combinations of these. We conclude that reliability is important at the MAC layer and the transport layer. MAC-level reliability is important not just to provide hop-by-hop error recovery for the transport layer, but also because it is needed for route discovery and maintenance. (This conclusion differs from previous studies in reliability for sensor nets that did not simulate routing. [4]) Second, we have developed RMST (Reliable Multi-Segment Transport), a new transport layer, in order to understand the role of in-network processing for reliable data transfer. RMST benefits from diffusion routing, adding minimal additional control traffic. RMST guarantees delivery, even when multiple hops exhibit very high error rates.

2 Architectural Choices

There are a number of key areas to consider when engineering reliability for sensor nets. Many current sensor networks exhibit high loss rates compared to wired networks (2% to 30% to immediate neighbors)[1,5,6]. While error detection and correction at the physical layer are important, approaches at the MAC layer and higher adapt well to the very wide range of loss rates seen in sensor networks and are the focus of this paper. MAC layer protocols can ameliorate PHY layer unreliability, and transport layers can guarantee delivery. An important question for this paper is the trade off between implementation of reliability at the MAC layer (i.e. hop to hop) vs. the Transport layer, which has traditionally been concerned with end-to-end reliability. Because sensor net applications are distributed, we also considered implementing reliability at the application layer. Our goal is to minimize the cost of repair in terms of transmission.

This work was supported by DARPA under grant DABT63-99-1-0011 as part of the SCAADS project, and was also made possible in part due to support from Intel Corporation and Xerox Corporation.
Fred Stann and John Heidemann are with USC/Information Sciences Institute, 4676 Admiralty Way, Marina Del Rey, CA, USA
E-mail: fstann@usc.edu, johnh@isi.edu.

2.1 MAC Layer Design Choices

Link layer Automatic Repeat Request (ARQ) refers to the hop-to-hop recovery of frames that arrive with errors The primary design choice we investigated at the MAC layer was whether or not to employ link layer recovery via ARQ for packets. The MAC layer used in our evaluations was 802.11 [7]. The primary reliability mechanisms provided by 802.11 are RTS/CTS, ACK, and randomized slot selection. RTS/CTS is the media access control packet exchange that guarantees that single transmitter will gain exclusive access to a shared transmission space. The ACK packet is sent by the receiver upon receipt of a data packet to inform the transmitter when successful transmission has occurred. This is a basic "stop-and-wait" ARQ mechanism where the transmitter times out and retransmits when an ACK does not arrive within a window of expectation. The 802.11 MAC does not employ RTS/CTS or ACK for multicast and broadcast transmissions due to ACK and CTS "implosion." It does, however, attempt to reduce the probability of broadcast collision by randomly selecting a transmission slot once an idle media is sensed. Clients of this MAC layer can choose to employ ARQ or not by selecting unicast or broadcast addresses. We utilized three different modes when considering MAC layer ARQ:

No ARQ: all transmissions are sent with a randomized send time and a broadcast MAC address. Unicasting is accomplished by address screening at the routing (in our case diffusion) layer. Such transmissions do not employ MAC layer reliability mechanisms such as RTS/CTS and ACK. In this mode, reliability is completely deferred to the transport or application layer. There are several possible benefits to this scheme. Firstly, there is a significant amount of overhead over time connected with the exchange of RTS/CTS and ACK packets that is avoided. Secondly, routing protocols like diffusion attempt to select high quality (lower error rate) paths for data transmission. The reliability mechanisms in 802.11 can make poor paths mistakenly look reliable to higher layers.

ARQ Always: all transmissions are sent via a stop-and-wait ARQ protocol with a single node address. This transmission method utilizes RTS/CTS and ACK with retries to bolster perceived reliability. When a node wishes to communicate with multiple neighbors, each neighbor must be sent a unicast packet. The number of ARQ retransmissions attempted before giving up is configurable. This method also has certain benefits for sensor nets. Packets that travel on the links identified in route discovery will be delivered with a high degree of reliability, despite the transient interference typical in a wireless domain.

Selective ARQ: a combination of No ARQ and ARQ. In this scheme packets sent to single neighbors employ a stop-and-wait ARQ mechanism. Packets sent to multiple neighbors have no ARQ. This method attempts to combine the benefits of both ARQ and No ARQ. Data and control packets traveling on established paths are unicast, using ARQ to bolster reliability. Packets used in route-discovery are broadcast to all neighbors without ARQ. Poor paths are statistically not selected for reinforcement, and the route-discovery procedure does not pay the overhead for reliability.

2.2 Transport Layer Design Choices

The transfer of data that is larger than the network MTU is a particularly difficult task in wireless communication and, more specifically in directed diffusion. Although protocols such as 802.11 have fragmentation and reassembly facilities, there are limits on the size of an entity can be broken up, and guaranteed delivery is not provided [6,7]. A single missing fragment from a large binary object (such as executable code) may render the data entity useless; therefore, transport layer facilities are required. Traditional transport layers, like TCP, assume that the primary cause of packet loss is congestion. As such, their focus is on congestion control. In sensor nets the primary problem is packet loss due to interference or low power.

The design decisions examined by this paper for the transport layer are primarily concerned with the balance of hop-by-hop vs. end-to-end functionality. Repair requests could be initiated by sinks (receiver end-points), or by in-network nodes on an established path. Obviously the type of MAC that any transport layer runs over will have a profound effect on how well the transport layer performs. This Section looks only at the transport layer.

Two transport layer paradigms will be examined in this paper and employed in the evaluation of RMST. The two transport layer schemes are:

End-to-End Selective Request NACK: The need for repair and the generation of repair requests takes place only at the sinks. Repair requests for specific missing fragments travel on a reverse reinforced path from sink to source, where the missing data is retransmitted.

Hop-by-Hop Selective Request NACK and Repair from Cache: In this paradigm, each caching node on the reinforced path from source to sink caches the fragments that make up a larger data entity. When such nodes sense a missing fragment, a repair request is sent to the next hop on the reverse reinforced path toward the source. If the requested fragment is in the local cache, a response is sent. If not, the NACK is forwarded to the next hop toward the source.

2.3 Application Layer Design Choices

Reliability can also be provided at the application layer. For sensors that automatically generate data periodically, a very simple reliability scheme is simply to wait for the next sensor reading [8]. This simple approach does not generalize to *large* objects, however. Even moderate per-packet loss rates quickly make the odds of ever getting a complete object over multiple hops small. (We analyze this case in Section 5).

Applications could handle both fragmentation/reassembly and end-to-end attempts at repair. In our evaluation (Section 6) we included an application layer reliability scheme (*End-to-End Positive ACK*) as a benchmark of what performance is achievable using standard diffusion for guaranteed delivery without the addition of a new transport layer.

End-to-End Positive ACK: In this approach a sink requests to receive a large data entity, which is fragmented at the source. When all fragments have arrived at the sink, it deletes its request. Sources send the entire set of fragments at pre-calculated intervals (a posteriori RTT) until request is deleted. We use this "transportless" paradigm to gauge if the overhead introduced by a transport scheme brings marginal benefits in terms of energy usage.

3 RMST Architecture

RMST was designed to run in conjunction with directed diffusion. In this Section we briefly review directed diffusion, give an overview of RMST, and then describe the protocol in some detail.

3.1 Diffusion Architecture Review

Directed diffusion [9,10] provides multipoint-to-multipoint communication for sensor nets much like traditional multicasting does so for wired nets. Sensitivity to energy conservation, data-centric routing, and the limitation of traffic volume via in-network processing are examples of some motivations that helped shape directed diffusion.

In diffusion, a sink *subscribes* to an *interest* that names a particular type and source of data. The naming of data is accomplished via attribute-value pairs. For example, an interest in counting how many people pass through a particular geographic region could be injected into an arbitrary node in a sensor net (typically an access point). Sensor node applications that have data available *publish* the fact, alerting the local diffusion code to look for matching interests. The sensors whose publications match a given interest, and the collection of sinks that expressed that interest, constitute a group that will eventually be connected by a distribution tree.

An interest is propagated from a sink toward a source. If geographic information is available, sources can be targeted geographically [11]. Each node that the interest passes through remembers the interest and which neighbors expressed it. Such local information is called a *gradient*. Every unique interest has an associated set of gradients. A source node sends data, when its publication matches a received interest. Initial data sent by a source across the sensor net is marked *exploratory* and disseminated along the reverse paths of the gradients. This amounts to a reverse-gradient propagation of data emanating at the sources and traveling to the sinks. Once exploratory data is sent, a single optimal *reinforced path* is established from sources to sinks. The sink uses an application dependent heuristic to decide which arriving exploratory message represents an optimal choice for *reinforcement*. Reinforcements travel from the sink back to the source creating a single reinforced path. When there are multiple sources or sinks, a distribution tree is formed. Subsequent data emanating from sources is called *reinforced data* because it is unicast along the reinforced tree.

Because wireless sensor nets are prone to rapidly changing conditions, such as the expiration of nodes or radio interference, sources periodically send out new exploratory messages to discover new routes that may be superior to the existing reinforced tree (old routes will be timed out). When a sink no longer wishes to express a certain interest, it *unsubscribes* that interest. This will eventuate in the removal of gradients and reinforced path elements from the sensor net.

Diffusion is built on a modular architecture that allows for great flexibility in adapting diffusion to specific applications. There is a diffusion core that communicates directly to the MAC layer below it and installable modules, called *filters*, above. Much like streams modules that can be assembled to create a particular networking stack, filters are installed via core diffusion to influence routing or add arbitrary application-specific behavior to a sensor net. The basic directed diffusion algorithm is implemented in the *gradient* filter. One could think of the gradient filter as a networking layer pushed on top of the MAC layer. Message traffic arriving at the core is passed to the highest priority filter.

The RMST protocols presented in this paper were implemented as a filter. One could consider RMST to be a transport layer pushed onto the diffusion stack, above the gradient filter (i.e. at a higher priority). Figure 1 demonstrates the relationship of RMST to a basic diffusion node.

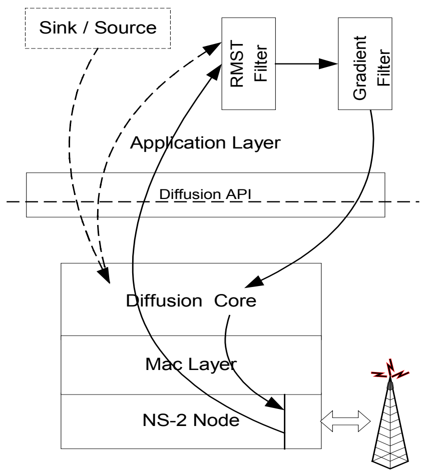

Figure 1

There is a strong resemblance in the above description to various forms of multicast delivery and routing. Some important distinctions should be made however. Perhaps the most interesting thing about diffusion is the extent to which it is "data-centric" [9,10]. The semantics of addressing, group management, and routing are entirely expressed by attribute-based naming. Rather than layering attribute-based naming on top of addressed-based routing, diffusion uses attribute vectors to dynamically establish routes on demand. There are no membership reports exchanged in diffusion; rather, the movements of interests and data, are used to establish and repair distribution routes. Another distinction of diffusion is the degree to which it is distributed. There is no concept of end-points and routers in diffusion. Every node runs the same diffusion algorithm and participates equally in routing and data forwarding. Sinks and sources are simply considered to be local agents. When a local agent publishes, subscribes, or sends attribute vectors, the essential diffusion code in that node accepts the "traffic" in the same fashion that it would accept traffic from neighbors.

3.2 RMST Overview

The RMST protocol presented in this Section was implemented as a filter that could be attached to any diffusion node on an as needed basis without recompilation of the diffusion core or Gradient filter. The caching vs. non-caching mode was made configurable at run time.

Reliability in RMST refers to the eventual delivery to all subscribing sinks of any and all fragments related to a unique RMST entity. A unique RMST entity is a data set consisting of one or more fragments coming from the same source. Delivery order, which is not guaranteed, is transparent to the clients of RMST. RMST does not include any real time guarantees.

There are two distinct transport services that need to be added to diffusion: effective management of the fragmentation and reassembly of units based on application semantics, and guaranteed delivery. Although these requirements are orthogonal, many applications require both. The division of a JPEG compressed image into "bands" that fit into the network MTU requires fragmentation/reassembly and guaranteed delivery to ensure reassembly.

In RMST, receivers are responsible for detecting whether or not a fragment needs to be re-sent. The term "receiver" here, however, does not necessarily mean sink. In the non-caching mode, only sinks monitor the integrity of an RMST entity in terms of received fragments. In caching mode, an RMST node collects fragments and is capable of initiating recovery for missing fragments to the next node along the path toward the source.

There are two types of loss detected by a "receiver": a "hole" in a sequence of fragments, and a truncated sequence. When a hole in a sequence of fragments is detected, the missing fragments should be specifically requested. This amounts to a selective-request ARQ-based behavior. The truncation of a sequence is really a special case of a hole, sensed by the receiver via a timeout geared to the expected receipt time of the next fragment. In our experiments we used timers set on our understanding of the network configuration, but we are exploring self-adaptive timers.

When a node fails, the normal behavior of Diffusion is to reestablish a new set of data gradients via an exploratory interest. To this extent sensor networks are self-repairing. RMST benefits from the underlying diffusion behavior related to failed nodes. Unlike PGM which must gather and maintain path state, or RMTP and SRM which watch and possibly generate message traffic to make decisions related to node failure, RMST can rely on the mechanisms in diffusion that guarantee the eventual discovery of a path from source to sink.

In caching mode, the caching of fragments along reinforced paths is used to limit power loss due to end-to-end retransmission. In non-caching mode, the underlying MAC layer is exploited to limit the transport layer overhead. It is precisely this tradeoff that is explored in the experiments.

3.3 RMST Basic Services

Unique identification in sensor networks is data-centric, therefore the transport layer must be able to recognize several new attributes added to support reliable traffic. An unfragmented data entity must have an application specific attribute (RmstNo) or set of attributes that serves to distinguish a particular *reliable* flow of data from source to sinks. In applications where complete disambiguation is difficult, a random ephemeral ID can be generated at the source [12]. Each fragment that makes up a fragmented data entity must also contain a sequential fragment id (FragNo). The total number of fragments that make up a data entity must be known (MaxFrag). There is a single control message generated by RMST, the NACK, which must be defined by an attribute.

3.4 RMST Support for Loss Detection and Repair

Loss detection is primarily timer driven. Where loss detection occurs depends on whether a node is configured for caching or non-caching mode. In non-caching mode, only sinks set timers to detect loss. In caching mode, each caching node on the reinforced path from source to sink detects loss. The basic mechanism for loss detection is a watchdog timer. A watchdog timer is instantiated for each new flow (RmstNo) that is added to a caching node's RMST database. The timer handler inspects the hole map and sends a NACK for any holes that have aged for too long. Multiple hole numbers are aggregated into a single NACK to conserve on control traffic. The maximum wait time heuristic could be dynamically adjusted. We plan to explore adaptive timers. Caching at the sinks makes delivery order unimportant, and allows for wide latitude in the relationship between the watchdog timer interval and the source send rate. When the watchdog interval is significantly slower than the send rate, the cache size must increase to retain more incomplete sets of fragments. Setting the watchdog timer faster than the send rate, minimizes the required cache size, but increases NACK traffic.

The only control message added to normal diffusion by RMST is the NACK. NACKs are unicast in the reverse direction along the reinforced path from source to sink. When the RMST filter gets a cache hit for a NACKed fragment, it unicasts that fragment to the requesting neighbor. When an RMST filter intercepts a NACK, and it cannot find the missing fragment in its local cache (or it's not in caching mode), it forwards the NACK on the reinforced path toward the source. In caching mode, the natural progression of traffic from source to sink, causes holes to be sensed sooner upstream, thus making NACK forwarding an unlikely event.

3.5 The back-channel

This particular implementation detail is a key enabler of efficiency in RMST. The reinforced paths that diffusion constructs for control and data are unidirectional, from source to sink. RMST needs a *back-channel* in order to deliver NAKs to upstream neighbors. RMST filters snoop at the last hops of reinforcement messages in order to construct reverse reinforced paths (in a distributed fashion). The creation of the *back-channel* is "free of charge" in that no additional transmissions or control data traffic are needed to create it. There is a back-channel associated with every flow. The back-channel is maintained in both caching and non-caching modes.

3.6 Node Failure

In the case of node failure a new reinforced path will be established by diffusion. Some nodes that weren't on the original reinforced path may now reside on it. The RMST filter automatically adapts to any such changes by creating a new back-channel that mirrors the new reinforced path. In caching mode, an RMST node that suddenly finds itself on a reinforced path (mid-sequence) will begin caching from the first received fragment on. NACKs for earlier fragments will be forwarded on the back-channel toward the source. In effect, the RMST filter will transfer responsibility for back-channel and caching maintenance to the new reinforced path.

3.7 Support for Caching

In caching mode, a node maintains a local cache of traffic in progress or recently transmitted. In non-caching mode, only the sources and sinks maintain a cache. The cache is indexed by the application specific flow id. Each cache entry has an associated fragment map and hole map. The fragment map contains the actual cached data indexed by the fragment Id. The hole map, used by the watchdog timer, is a list that contains missing or overdue fragments for a particular flow. Hole map entries contain a fragment id, a timestamp indicative of when a NACK for this fragment was sent, and a flag indicating whether or not a NACK is outstanding. On receipt of a fragment a caching mode filter must identify missing or late fragments and add them to the hole map. Currently caching nodes accomplish cache flushing via a fixed timer, although we plan to switch to an LRU algorithm.

4 Analysis of MAC Layer Retries

Because one of our primary design decisions concerns the use of MAC layer ARQ, we analyze the effect that the number of retries attempted by a transmitter has on the resultant probability that a packet will arrive between end points. If we define p as the probability of success for a single attempt across one hop, and R as the number of MAC-level attempts, the probability of success with MAC-level ARQ is:

$$p_h = \sum_{i=0}^{R-1} p \cdot (1-p)^i \qquad (1)$$

Which simplifies to the probability of not failing at all R tries:

$$p_h = 1-(1-p)^R \qquad (2)$$

When considering H hops, the end-to-end probability of arrival is:

$$p_e = ph^H \qquad (3)$$

Figure 1 graphs end-to-end arrival rates for different numbers of MAC-level retries. As can be seen in Figure 1, the use of at least 3 retries is vital to reliable data delivery in this scenario.

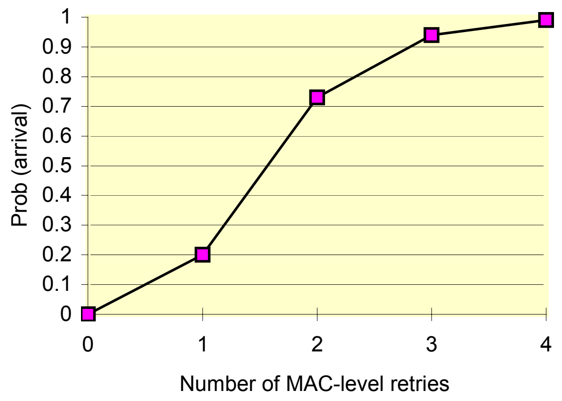

Figure 1: Probability of arrival across 40 hops with an average error rate of .10 per hop, given R retries per hop.

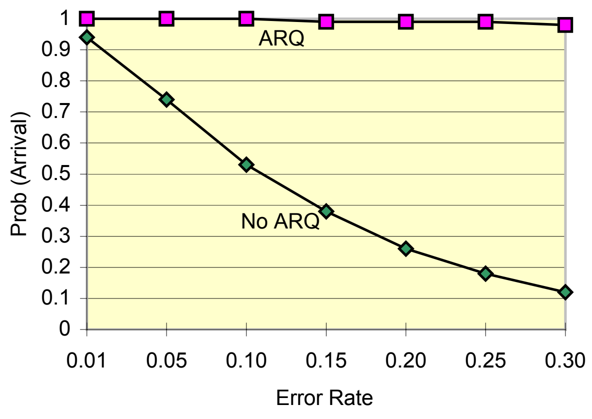

Figure 2: Probability of arrival across 6 hops

Because wireless nets have physical level error rates that can be relatively high compared to wired nets [5] we compare ARQ vs. non-ARQ MAC layers for a particular scenario. Figure 2 shows two data series plotting the probability that a packet will arrive at its final destination after traversing 6 hops at the given error rate with the number of retries set to three. As Figure 2 demonstrates, the probability of arrival at the end point plummets with the error rate for the non-ARQ case, but remains quite high with ARQ set to three retries.

5 Analysis of Transport Layer Hop-by-Hop vs. End-to-End

In order to do a basic analysis of the benefits of hop-by-hop repair vs. end-to-end repair at the transport layer, we assume that the MAC layer provides a particular level of reliability expressed as a probability of success per link. The intent is to examine transport layer efficiency given the success rate presented to it from lower layers.

First we look at the cost of doing end-to-end repair without caching or intermediate transport layer repair.

If a large data object is broken into *M* fragments and transmitted a single time from a source across H hops to a sink, the expected number of those fragments to arrive at the sink can be derived using the end-to-end probability from eq. 3 as follows:

$$E[f(M,H)] = \sum_{m=1}^{M} m \cdot p_e{}^{m} \cdot (1-p_e)^{M-m} \tag{4}$$

We would also like to know the expected number of hops that a failed packet will travel (f_h). (We need to adjust p_e for each value of n).

$$E[f_h(H)] = \sum_{n=1}^{H-1} n \cdot p_e{}^{n-1} \cdot (1-p_e) \tag{5}$$

Therefore the approximate cost in terms of link-wise fragment transmissions to accomplish one attempt to send *M* fragments (with an end-to-end transport layer) is:

$$H \cdot E[f(M,H)] + E[f_h(H)] \cdot (M - E[f(M,H)]) \tag{6}$$

Using a small program we can calculate *E[Tx(H,M)]*, the total number of link-wise transmissions required to get a set of *M* fragments across *H* hops. We apply equation 6 to get a transmission count, calculate the number of fragments expected not to make it using equation 4, and recursively call on the remainder to be sent (accumulating the count).

If we are caching data at each node and doing transport layer recovery on a per-hop, the expected number of retries to move a fragment one hop is:

$$E[r(K)] = \sum_{k=1}^{\infty} k \cdot p_h \cdot (1-p_h)^{k-1} \tag{7}$$

For the caching case, the number of link-wise transmissions required to get a set of M fragments across H hops is:

$$E[Tx(H,M)] = M \cdot N \cdot E[r(K)] \tag{8}$$

Importantly, *E[Tx(H,M)]* grows much faster with the non-caching method as M and n increase. For example, if we hold the probability for success at .9 we get the results accumulated in Table 1.

What's significant in this analysis is that we can look for a loss rate that might obviate the advantage of caching in the transport layer. We have already demonstrated in the last section the dramatic improvement available by raising the retry count for MAC layer ARQ. If we hold the number of fragments and hop count constant and vary the probability for success we get curves such as in Figure 3. The loss rate presented to the transport layer by the MAC layer needs to get below one percent for the advantages of caching and hop-by-hop repair to be marginalized. Without MAC layer ARQ, physical layer loss rates would need to be below one percent. Such low loss rates are common in wired nets, but atypical in wireless [1,5,6].

# Fragments	5 Hops	10 Hops
5	27.77 / 42.33	55.55 / 143.39
10	55.55 / 84.67	111.11 / 286.79
20	111.11 / 169.35	222.22 / 573.59

Table 1: Number of total transmissions required to send M fragments across N hops (with cache/without cache)

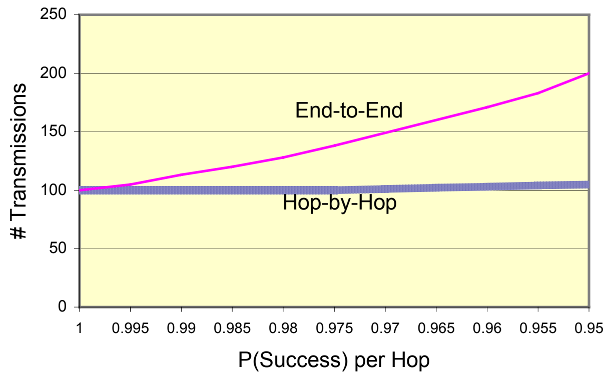

Figure 3: Number of transmissions required to send 10 fragments across 10 hops. Hop-by-hop vs. End-to-End repair.

6 Evaluation of RMST

Sections 4 and 5 presented simple analysis of reliability at the MAC and transport levels. We next present the simulation studies of our RMST implementation to evaluate the interaction of reliability at different layers. The analysis suggests the importance of hop-by-hop recovery, but it could reside at the MAC or transport layer (or both). The experiments were run in ns-2 [13] using an 802.11 MAC layer and directed diffusion attached to wireless nodes. The 802.11 MAC uses non-ARQ for broadcast packets and ARQ for unicast packets. Diffusion is capable of delivering unicast messages over a broadcast MAC. It does so by embedding its own unique node addresses in packets. Therefore non-ARQ MAC unicasting was achieved by broadcasting at the network-layer, but doing unicast address resolution in diffusion.

Parameters that could be altered in the experiments include:

Error Rate: This refers to the physical layer lost packet rate. In ns-2, errors at the physical layer can be injected by attaching an "error model" to each node. Three error rates were used in initial experiments: 0%, 1%, and 10%. Several experiments were also done at the elevated error rates of 20% and 30%.

Hop Count: The number of hops required to traverse a rectangular grid of nodes from source to sink. Because diffusion is a route discovery protocol, this can vary slightly in simulation or the real world.

Number of Retries: This parameter applies to MAC layer ARQ count. It is the number of MAC layer retries attempted before abandoning a transmission.

Size of Blob: The number of bytes to be transmitted from source to sink as a series of fragments.

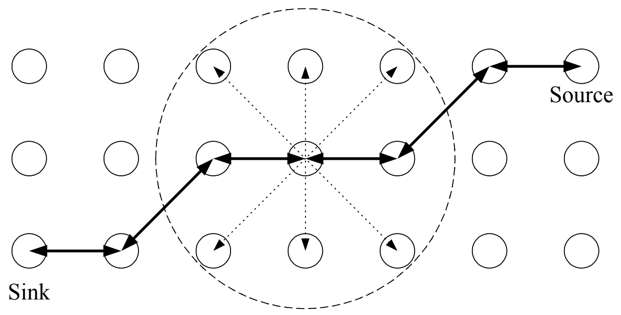

Figure 4
Layout of test grid showing radius of central node, reinforced path from source to sink and back-channel

Because of the potentially large number of experiments that can be performed in this multi-dimensional problem space, three of the variables were held constant: Hop Count (6 hops, 21 nodes), retries (4), size of blob (5k). The simulations employed a grid with equidistant nodes that could only communicate with "immediate" neighbors (for example, the central node in Figure 4 sees eight neighbors). The source was placed in the upper-right position of the grid and the sink was placed in the lower-left position. We selected this simple topology for ease of evaluation and comparison to the analysis. We are currently implementing RMST in a sensor node testbed and plan to evaluate realistic topologies there.

The simulation logs were enhanced to count every byte transmitted, including CTS/RTS and ACK bytes when ARQ was used. The 5KB blob was broken into 50 100-byte fragments that were transferred from source to sink across the grid. The transmission byte totals included the control messages used by diffusion to propagate the interest, reinforce a gradient, and maintain the interest and gradients over time.

We normalized all results to the cost of sending the messages without ARQ or transport layer overhead, i.e. 87,818 bytes. The idealized byte count was measured by sending the entire set of 50 fragments with zero errors, no MAC level ARQ and no transport scheme. The count includes the ongoing message traffic exchanged by diffusion to propagate interests, establish and reinforce routes, and otherwise maintain state. It also includes per packet overhead including headers and trailers. This provided a baseline of the best that could be achieved with no reliability overhead. Experiments in which individual simulations exhibited a low variation were repeated 10 times to calculate an average. Experiments with error rates higher than 10% were repeated 20 times.

6.1 Baseline End-to-End Positive ACK

This experiment was run to establish a baseline of what is achievable with standard diffusion without the addition of a transport layer. The entire set of fragments was sent at regular intervals until the sink unsubscribed. The results are summarized in Table 2.

PHY Error Rate	No ARQ	ARQ All	Selective ARQ
0	.93 (.07)	.57 (.03)	.65 (.03)
.01	.51 (.04)	.56 (.03)	.61 (.05)
.10	.21 (.05)	.47 (.09)	.54 (.06)

Table 2: End-to-End Positive ACK
Normalized byte transmissions required for diffusion to transfer 50 fragments of 100 bytes across 6 hops without any transport layer

Not surprisingly, with no physical layer errors the overhead associated with sending CTS/RTS and ACK frames is made apparent in the first row. Using ARQ, even selectively, adds significant overhead. The two right columns demonstrate that using Selective ARQ is consistently about 7% more efficient than using ARQ for every packet. Selective ARQ is attractive from another aspect in that route discovery packets do not have their reliability boosted at the MAC layer. This allows the network layer (diffusion) to make better decisions about what routes are best. The results in the No ARQ column plummet with increased error rate. This is a direct result of the exponential decay of reliability demonstrated in Figure 2. Clearly, when average error rates are high, using ARQ at the MAC layer, when there is no transport layer, is extremely beneficial.

6.2 RMST with Hop-by-Hop Recovery and Caching

In this experiment we ran the RMST filter in caching mode at every node in the grid with in-network NACKing. Results are summarized in Table 3.

There are two significant results in this experiment. Comparing the columns for ARQ and Selective ARQ from this experiment with the results of the first experiment, we see very slight improvement in all cases. This would seem to suggest that transport layer hop-by-hop recovery adds little to the reliability available from a robust MAC layer. At the same time, if we focus on No ARQ at the 10% error rate, we see the other interesting result. Using hop-by-hop recovery at the transport layer instead of the MAC layer was 15% more efficient than Selective ARQ at the same error rate.

PHY Error Rate	No ARQ	ARQ All	Selective ARQ
0	.99 (.05)	.60 (.06)	.68 (.06)
.01	.95 (.06)	.57 (.06)	.67 (.07)
.10	.76 (.07)	.48 (.07)	.61 (.07)

Table 3: Hop-by-Hop Selective NACK and Caching
Normalized byte transmissions required for diffusion to transfer 50 fragments of 100 bytes across 6 hops with hop-by-hop caching and repair

This result would make it appear that doing hop-by-hop recovery at the transport layer is preferable to doing it at the MAC layer, where reliability overhead is paid for each and every unicast packet. Log analysis for this case showed that many more Exploratory packets were sent before a reinforced path between source and sink was established. Nonetheless, once a path was established, the hop-by-hop recovery at the transport layer was extremely efficient (with no ARQ overhead).

The No ARQ result (at 10%) is somewhat specious in light of the exponential decay of arrival probability as the hop count or error rate increase. Transient error rates in excess of 10% and paths longer than 6 hops are not uncommon in sensor nets [1]. Non-transport-layer messages used in path reinforcement must be propagated from sink to source. If the probability of arrival decays beyond a certain point, basic diffusion has a difficult time maintaining routes. A fourth experiment (see below) was instantiated because of this partial result.

6.3 RMST with End-to-End Recovery

This experiment was run with the RMST filters in non-caching mode. The only in-network recovery was when the MAC layer was configured to use ARQ or Selective ARQ. Transport layer recovery was accomplished end-to-end via NACKs sent along the back-channel from sink to source. This is similar to traditional transport recovery in wired networks. Results are presented in Table 4.

There are two important observations to be made about these results. On top of the No ARQ MAC, at the 10% error rate, the simulation did not terminate within the 600 seconds allotted. Log analysis showed that there were numerous holes that required NACKs, which had a difficult time making it from sink to source. This result was not surprising. It simply means that some sort of hop-by-hop recovery is required either at the MAC or transport layer in order to implement guaranteed delivery. There was another important result. If you compare the two columns for ARQ and Selective ARQ with those of the previous experiment, you will see virtually no change in efficiency. Comparative log analysis revealed that MAC layer ARQ or Selective ARQ made NACKs so rare that hop-by-hop vs. end-to-end NACKing at the transport layer had little difference in performance.

PHY Error Rate	No ARQ	ARQ All	Selective ARQ
0	1.0 (.05)	.61 (.08)	.67 (.07)
.01	.90 (.06)	.60 (.10)	.66 (.07)
.10	n/c	.49 (.09)	.61 (.07)

Table 4: End-to-End Selective NACK
Total byte transmissions required for diffusion to transfer 50 fragments of 100 bytes across 6 hops with end-to-end repair.

PHY Error Rate	Hop by Hop RMST NoARQ	Hop by Hop RMST Sel ARQ	End to End RMST Sel ARQ
.20	.48 (.19)*	.40 (.18)	.40 (.17)
.30	n/c	.24 (.23)	.27 (.25)

Table 5: High Error Rate Test
Total byte transmissions required for diffusion to transfer 50 fragments of 100 bytes across 6 hops with high error rates.

6.4 Performance under High Error Rates

The purpose of this experiment was to further examine the partial results of the three previous experiments on selected combinations of MAC and transport layer. In sensor nets, correlated losses due to interference can exhibit transient error rates that are quite high [1]. This experiment was performed on selected schemes that had performed best at the 10% error rate. From the schemes that employed MAC layer ARQ, we chose those that used selective ARQ. The non-ARQ scheme, which previously outperformed all others, was RMST in caching mode over a MAC with no ARQ. Results are summarized in Table 5.

We see from the No ARQ column that transport layer hop-by-hop recovery without MAC layer ARQ broke down somewhere between .20 and .30 error rate. Log analysis showed that standard control messages used by diffusion to reinforce and maintain paths rarely succeeded in establishing any viable routes. At .20 error rate (see asterisk in Table), the efficiency rating is misleading. Log analysis showed that an inordinate amount of time (on the order of minutes) was spent establishing a viable route. Such delays would be considered unacceptable in a real test bed.

Notice that RMST running over Selective ARQ had very similar efficiency in both caching (hop-by-hop) and non-caching (end-to-end) mode. There are several additional considerations that need to be pointed out in this regard. When dealing with multiple sinks or applications that require localized in-network processing, experiments with NACK-based multicast reliability protocols, like SRM [14] and PGM [15], have demonstrated an advantage to caching at strategically selected nodes.

7 Related Work

There are several areas that we looked to for related work: ad-hoc sensor networks, TCP over wireless nets, and multicast transport schemes.

7.1 Sensor Nets

Much of the existing work related to reliability in sensor networks deals with route discovery and maintenance, and not reliable data transport. Mobile ad-hoc networks employ a variety of routing protocols concerned with finding high quality paths. For example Signal Stability Adaptive Routing (SSA) [16] attempts to differentiate high quality routes by monitoring signal strength. DSR [17] discovers new routes via flooding, usually accomplished by broadcasting without ARQ. Higher quality routes are statistically selected more often.

TAG [18] is a tree-based aggregation and routing system for ad-hoc sensor nets. The identification of "capable neighbors" is a major concern in TAG, relying on a scheme in which nodes monitor the quality of the links with their parents. When a node sees that the quality (loss rate) to a parent is "significantly worse" than that of another potential parent, the node "re-parents" to improve on the probability of loss. This is very similar to the occasional transmission of exploratory messages in diffusion to discover better paths. Another device investigated by TAG was the use of in-network caching. Caching in TAG deals with remembering the "state" of children (in terms of values used in aggregation) rather than providing repair for specific packets (TAG does not guarantee reliable data transport). It nonetheless demonstrates that caching can ameliorate the elevated loss in sensor nets.

PSFQ (Pump Slowly Fetch Quickly) [4] is a transport layer paradigm for sensor nets that is very close to our work. It is characterized by hop-by-hop error recovery, repair requests via NACKs that are delivered at a rate faster than the source transmission rate, and in-network caching. PSFQ is assumed to run over a non-ARQ MAC layer. Two important results of their work are that end-to-end recovery is not appropriate for sensor networks, and that recovery is best accomplished at the transport layer. The failure of purely end-to-end recovery is consistent with our analysis and simulation results.

We differ with their suggestion that ARQ is best provided at the transport layer instead of the MAC. If data transport were the only service in a sensor network benefiting from ARQ, then it would make sense to push ARQ as high up the stack as possible. However, our simulations of diffusion found that lack of reliable routing prevented operation entirely at high error rates (more than 30% for 6 hops). The PSFQ study did not observe this problem because they considered an idealized, "omniscient" multicast in simulation that implemented routing out of band. Ignoring the problem of route establishment, our results agree with theirs that hop-by-hop transport layer caching improves performance, not just at very high error rates (the 30-70% they observe), but also at rates as low as 10%. Our different conclusions are based on the assumption that it is possible to provide an energy efficient MAC layer with ARQ. We agree that the RTS/CTS mechanism in 802.11's ad hoc mode is too energy expensive for long-term operation since it requires continuous listening; we look to recent work in energy–conserving ARQ schemes to reduce this cost [19].

7.2 TCP Over Wireless

The extension of TCP into heterogeneous networks that includes wireless links has yielded a number of interesting observations about the nature of reliability in wireless networks. The most obvious problem addressed by these investigations is that TCP assumes the primary cause of packet loss to be congestion rather than lossy links. When this is not the case, as in wireless, TCP behaves poorly [20,21]. Some solutions assume that hop-by-hop reliability is provided by a link layer that does any number of retries required to ultimately move a packet forward [22]. Such assumptions may not be completely relevant in sensor nets where the cost of link layer retries and the occurrence of failing nodes for packets must be taken into account. The SACK (Selective ACK) [23] and SMART [20] are attempts to refine the coarse granularity of TCP's cumulative ACK. Research into TCP aware link layers includes doing link repair at a rate faster than the TCP timeouts and Explicit Loss Notification [20], which allows non-congestion-related losses (i.e. packets dropped by the link layer) to be identified so that retransmissions may be performed without invoking congestion-control. The Snoop Protocol [20] places a *snoop agent* in a base station. Snoop agents cache packets from senders, and watch for duplicate acknowledgments from receivers. They provide somewhat localized repair and shield senders from doing congestion control for wireless losses. Nonetheless, because snoop agents only reside in base stations they are not generally applicable to distributed multi-hop sensor networks.

7.3 Multicast Transport Layers

IP-based multicast delivery systems generally assume that packet loss is inevitable. Information from a particular source is carried a single time on each link, only registered group members reside on the distribution tree, and routing is maintained by adaptive protocols such as DVMRP [24] and PIM [25]. These efficiencies were introduced because of difficulties encountered when using traditional unicast methods to support distributed applications, such as the multicasting of real-time multimedia information. Three multicast protocols which each represent a different class of reliability solutions are: RMTP, SRM, and PGM.

Reliable Multicast Transport Protocol [26] is an ACK-based protocol that avoids the well-known ACK implosion problem via a hierarchy of special in-network nodes called *designated receivers* (DRs). DRs receive ACKs from multiple down stream nodes and send ACKs to a single upstream DR or sender. They also do in-network caching of data to satisfy any subset of downstream nodes that are missing a data fragment. Several ideas from RMTP would appear to map well to the distributed nature of sensor nets. One could think of DRs as both virtual senders and virtual receivers. Immediate downstream nodes cannot distinguish between a DR and a real sender. Upstream nodes are not aware if they are sending to a small set of actual receivers or a set of DRs. This sort of anonymity and distributed responsibility is built into directed diffusion [9,11].

The Scalable Reliable Multicast protocol, SRM [14], guarantees eventual delivery of sequenced data to all multicast group members, albeit not delivery order. ACK implosion avoidance is accomplished via NACKs, which are multicast by receivers when discontinuities in sequence numbers are perceived. Because NACKs are multicast, any receiver that has cached the data in question can restore the missing fragment. An appealing aspect of SRM and other NACK-based protocols is the "on-demand" nature of repair requests. A negative aspect of RMTP for sensor nets, however, is the routine stream of control traffic generated regardless of loss rate. A problem with both SRM and RMTP, in terms of porting sensor nets, is the strict reliance on RTT, which can exhibit a larger variance in wireless.

PGM, Pragmatic General Multicast [15], is a commercially evolved multicast reliability standard. It relies heavily on in-network processing via "PGM-aware" routers. It is a NACK-based scheme. NACK implosion is controlled in several ways. When a receiver recognizes a skipped sequence number, it sets a random timer and listens for any restorations of the same packet. PGM routers also "fuse" NACKs by not forwarding duplicate NACKs upstream. PGM allows for in-network caching via "designated local repairers" (DLR). Several ideas in PGM would appear to translate well into sensor net transport schemes. PGM's reverse path routing from receivers to sinks and the path repair mechanism bears some resemblance to directed diffusion's path reinforcement and repair paradigm. The in-network caching by path-aware DLRs aligns well with the distributed nature of sensor net routing algorithms. Nonetheless, the proliferation of control packets in PGM and sheer volume of implementation details render a straight port sensor nets impractical.

8 Future Work

We consider it especially important to try RMST in an actual sensor network. At the time of this writing we have begun initial testing in the ISI testbed. Preliminary results have demonstrated the basic operability of Rmst; nonetheless, we have identified several areas that need to be addressed as part of real-world deployment. First, we observed that incorrectly short timeouts can cause excessive NACK traffic (similar to congestion collapse [27]). We plan to automatically tune timing. Second, we have observed interactions between reliable and unreliable links, where the resultant selection of bad paths causes high loss (as observed previously [10]). We are exploring ways to address this problem.

The ability of RMST to dynamically configure itself for caching constitutes important future work that will provide a valuable resource for diffusion clients that rely on in-network processing.

9 Conclusions

We contend that the best implementation for reliability in distributed sensor network architectures involves both the transport and MAC layers. It is beneficial to employ MAC level ARQ for control and data packets that are unicast on the paths selected for data transfer. Route discovery packets should be broadcast without any MAC layer reliability mechanism. The periodic discovery of best routes should reflect the statistical probability of arrival. The selective use of ARQ is available for sensor networks via an energy aware MAC layer such as S-MAC [19]. In order to support guaranteed delivery in sensor nets, which can demonstrate high error rates, a NACK based transport layer running over a selective-ARQ MAC layer is an appropriate solution. We conclude that RMST constitutes a good basis for expanding the application domain of directed diffusion into areas requiring guaranteed delivery and fragmentation/reassembly. It does so in a fashion that leverages the strengths of diffusion yet minimizes the amount of extra overhead required to support itself. The principles applied in creating RMST are applicable to other sensor network routing protocols. Applications that require in-network processing or have large numbers of sinks will benefit from the capability of RMST to be dynamically configured for caching.

REFERENCES

[1] Jerry Zhao, Ramesh Govindan. Connectivity Study of a CSMA based Wireless Network. Technical Report TR-02-774, USC/ISI, Los Angeles, CA, 2002.

[2]Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, pages 263-270, Seattle Washington, Aug 1999.

[3]Gregory J. Pottie and William J. Kaiser. Embedding the Internet: wireless integrated network sensors. *Communications of the ACM*, V.43 (N .5), pages 51-58, May, 2000.

[4] C. Wan, A. Campbell, L. Krishnahmurthy. PSFQ: A Reliable Transport Mechanism for Wireless Sensor Networks. ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, Georgia, Sept 2002.

[5] Jean Tourrilhes. Robust Broadcast: Improving the reliability of broadcast transmissions on CSMA/CD. Hewlett Packard Laboratories, Bristol U.K.

[6] Ken Tang, and Mario Gerla. Mac Reliable Broadcast in Ad Hoc Networks. MILCOM 2001, McLean, Virginia, October 2001.

[7] LAN MAN Standards Committee of the IEEE Computer Society, Wireless LAN medium access control (MAC) and physical layer (PHY) specification. IEEE Std 802.11, IEEE, 1997.

[8] W. Chu, J. Hellerstein, and M. Lan. The Exclusive-Writer Protocol: A Low Cost Approach for Updating Replicated Files in Distributed Real Time Systems. In Proceedings of the 3rd International Conference on Distributed Computing Systems, pages 269-277, Oct 1982 IEEE.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking*, pages 56-67, Boston, MA, USA, August 2000. ACM.

[10] John Heidemann, Fabio Silva, Charlemek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low level naming. In *Proceedings of the Symposium on Operating System Principles*, pages 146-159, Chateau Lake Louise, Banff, Alberta, Canada, October 2001. ACM.

[11] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks. Technical Report TR-01-0032, University of California, Los Angeles, Computer Science Department, 2001.

[12] Jeremy Elson, and Deborah Estrin. Random Ephemeral Transaction Identifiers in Dynamic Sensor Networks. In *Proceedings of the Twenty-first International Conference on Distributed Computing*, Phoenix, AZ Apr 2001.

[13] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in Network Simulation. *IEEE Computer*, V.33 (N. 5), pages 59-67, May 2000.

[14] Sally Floyd, Van Jacobson, Ching-Gung Liu, and Lixia Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *Proceedings of the ACMSIGCOMM Conference*, pages 342-356, Cambridge, MA, August 1995, ACM.

[15] Tony Speakman, Dino Farinacci, Steven Lin, Alex Tweedly, Nidhi Bhaskar, Richard Edmonstone, Rajitha Sumanasekera, and Lorenzo Vicisano. PGM Reliable Transport Protocol. *Internet-Draft (describing protocols used by Cisco and Whitebarn)*, Feb 2001.

[16] R. Dube, C. Rais, K. Wang, and S. Tripathi. Signal Stability Based Adaptive Routing for Ad Hoc Mobile Networks. In *IEEE Personal Communications*, Feb 1997.

[17] D. Johnson, and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks in Mobile Computing, pages 153-181. Kluwer Academic, 1996.

[18] S. Madden, M. Franklin, and J. Hellerstein. TAG: a Tiny Agregation Service for Ad-Hoc Sensor Networks. OSDI, Dec 2002.

[19] W. Ye, J. Heidemann, D. Estrin. An Energy Efficient MAC Protocol for Wireless Sensor Networks. Technical Report TR-543, USC/ISI, Los Angeles, CA, September 2001.

[20] H. Balakrishna, V. Padmanabhan, S. Seshan, and R. Katz. A Comparason of Mechanisms for Improving TCP Performance over Wireless Links. In *IEEE Transactions on Network*ing, 756 -769, Vol. 5 1997, ACM.

[21] H. Chaskar, T. Lakshman, U. Madhow. TCP Over Wireless with Link Level Error Control: Analysis and Design Methodology. In *IEEE Transactions on Network*ing, pages 605 -615, Vol. 7 1999, ACM.

[22] H. Chaskar, T. Lakshman, U. Madhow. On the Design of Interfaces for TCP/IP over Wireless. In *Proceedings of IEEE Milcom*, 1996.

[23] V. Jacobson and R. T. Braden. TCP Extensions for Long Delay Paths. RFC, Oct 1988. RFC-1072.

[24] D. Waizman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol, *DARPA Request for Comments (RFC) 1075*, Nov 1989.

[25] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu , and L. Wei. The PIM architecture for wide-area multicast routing. In *IEEE Transactions on Network*ing, pages 153-162, April 1996, ACM.

[26] John C. Lin, and Sanjoy Paul. RMTP: A Reliable Multicast Transport Protocol. In *Proceedings of IEEE INFOCOM*, pages 1414-1424, March 1996.

[27] Van Jacobson. Congestion Avoidance and Control. In Proceedings of the SIGCOMM '88, pp. 314-329. Stanford, California, ACM. August, 1988.