In Proceedings of IEEE GlobeCom. Rio de Janeiro, December 1999

ADAPTIVE INVERSE MULTIPLEXING FOR WIDE-AREA WIRELESS NETWORKS

Alex C. Snoeren

snoeren@lcs.mit.edu
Laboratory for Computer Science
Massachusetts Institute of Technology

Abstract

The limited bandwidth of current wide-area wireless access networks (WWANs) is often insufficient for demanding applications, such as streaming audio or video, data mining applications, or high-resolution imaging. Inverse multiplexing is a standard application-transparent method used to provide higher end-to-end bandwidth by splitting traffic across multiple physical links, creating a single logical channel. While commonly used in ISDN and analog dialup installations, current implementations are designed for private links with stable channel characteristics.

Unfortunately, most WWAN technologies use shared channels with highly variable link characteristics, including bandwidth, latency, and loss rates. This paper presents an *adaptive* inverse multiplexing scheme for WWAN environments, termed *Link Quality Balancing*, which uses relative performance metrics to adjust traffic scheduling across bundled links. By exchanging loss rate information, we compute relative short-term available bandwidths for each link. We discuss the challenges of adaptation in a WWAN network, CDPD in particular, and present performance measurements of our current implementation of Wide-Area Multi-Link PPP (WAMP) for CDPD modems under both Constant Bit Rate (CBR) and TCP loads.

1 Introduction

A large number of wide-area wireless access network (WWAN) technologies have recently emerged, including Metricom's Ricochet Packet Radio network, *Global System Mobile* (GSM) [13], IS-95 [10], and Cellular Digital Packet Data (CDPD) [15]. The defining characteristic of WWANs is their use of a shared channel with long and variable round-trip times (RTTs), typically on the order of 500ms, coupled with a relatively low and variable bandwidth (usually in the tens of Kb/s). Additionally, many of these wireless technologies support link-level ARQ schemes for reliable transmission. While such mechanisms provide better line error characteristics, they combine with channel access contention to introduce significant delay variability.

The low bandwidths provided by WWAN technologies are often insufficient to support demanding applications, such as voice recognition and high-resolution imaging. In these contexts, one obvious solution is to spread connections over multiple physical links. By *striping* data over many physical channels (called a *bundle*), possibly by breaking packets up into fragments, it is possible to present a single logical channel with increased available bandwidth. When using such techniques, however, variations in the channel characteristics of any particular link can have catastrophic impact on the performance of the bundle as a whole. In particular, variations in the perceived transmission delay of an individual member link may cause delays in demultiplexing and packet assembly. In the absence of sophisticated scheduling techniques to balance delay, bandwidth variability between member links can cause the performance of many transport protocols to be throttled by the slowest member link, even when traffic is allocated in proportion to link speeds. Similarly, losses on one link may prevent packet reassembly, increasing loss rates markedly. These *amplification effects* motivate our design of an adaptive inverse multiplexing algorithm in WWAN networks. In this paper, we study the Bell Atlantic Mobile CDPD network, but we believe our work is applicable to all WWAN technologies, and, more generally, to any shared links with variable characteristics.

Inverse multiplexing is fairly common in current network technologies. Most implementations, however, assume physical transport mechanisms with constant bit rates and stable link characteristics such as those found in circuit switched networks [1, 6, 7]. This is not the case with CDPD. Commercial CDPD networks use an IP-based transport mechanism such as TCP or UDP to transport data. These mechanisms are subject not only to the expected loss and delay characteristics of the Internet, but additional WWAN-specific issues as well, such as increased round-trip times and delay variability. The added synchronization requirements of an inverse multiplexing scheme only compound any losses or variability in bandwidth or delay.

This work was supported in part by Defense Advanced Projects Research Agency (DARPA) contract number DAAN02-98-K0003. The author is also supported by a National Defense Science and Engineering Graduate (NDSEG) Fellowship.

Reliable transport protocols require reasonable limits on packet reordering, jitter, and loss in order to function optimally. In particular, TCP has been shown to be extremely sensitive to variations in delay, as they affect both its ACK-based clock and RTT estimates for packet retransmission [3, 4, 5]. The Fast Retransmit algorithm [RFC2001] also makes assumptions about the level of packet reordering, and induces spurious retransmissions in the face of excessive reordering. It is therefore important to ensure that any multi-link scheme limit packet reordering and delay jitter as much as possible.

The multiplexor must then schedule packet fragments so that they are received and reassembled in roughly the same order they arrived, with a minimum of added delay. This requires balancing the fragment distribution according to the current *bandwidth*delay* product of each individual link. Due to the depth of network queues relative to the *bandwidth*delay* product, and channel access asymmetries inherent in CDPD, it is extremely difficult to obtain accurate measurements. Instead, we use relative performance metrics to adjust the traffic distribution between links, and allow the end-to-end transport protocol's bandwidth probing algorithm to function as it would normally. We term this dynamic adaptation *Link Quality Balancing*.

The rest of this paper is organized as follows. Section 2 details related work on improving performance in similar WWAN environments. Section 3 sets forth the assumptions made concerning the CDPD network used in our study. In Section 4 describes the details of our inverse multiplexing scheme. We examine the factors affecting TCP performance over multiplexed CDPD links in Section 5, and discuss possible techniques for addressing them. Section 6 provides details of our sample implementation, built using the techniques discussed in Sections 4 and 5. Finally, we present our conclusions in Section 7.

2 Related Work

Many commercially available network devices support inverse multiplexing using special hardware at the sender and receiver. The BONDING consortium [6] specifies techniques for packet striping on 56 and 64kbps circuit switched channels. More general inverse multiplexing schemes, as surveyed in [7], often fail to provide fair bandwidth allocation in the presence of variable size packets, or introduce significant reordering. The *strIPe* protocol [1] addresses these shortcomings, providing a general mechanism for fair bandwidth allocation with limited packet reordering, but does not adapt to changing channel capacity.

Previous work has shown Wireless TCP throughput to be sub-optimal due to interactions between the link layer and transport layer protocols. Asymmetry in media access, as found in CDPD up-channels, causes ACK-compression, and large delay variations due to a reliable link layer mechanism conspire to confound the TCP congestion control mechanisms [5]. While many of these problems have previously been studied and understood, most solutions assume a *basestation* paradigm, where agents are inserted at the interface between the wired and wireless links [3, 4]. In a commercial CDPD environment, however, users do not have access to the Mobile Data Base Stations (MDBS), and therefore cannot completely control the buffering, queuing, and retransmission mechanisms being used.

Inverse multiplexing restores the basestation view, as one can consider the entire path from multiplexor to demultiplexor to be one logical link. Traffic shaping and other known techniques can (and should) be employed, considering the multiplexor as the basestation. In particular, we believe *split-connection* methods similar to I-TCP [2] may be especially well-suited; there is little additional benefit to be gained from a transparent agent in this case, since network connectivity is lost in the event of a multiplexor failure. By terminating the connection at both ends, attention can be focused exclusively on the WWAN portion of the overall path, perhaps utilizing novel transport protocols specially designed or tuned for WWANs [9, 11, 14, RFC2488]. Alternatively, mechanisms such as Snoop agents [4] could be used if a transparent solution was desired in some particular environment.

Rather than cloud our results with additional factors by considering various proposals for transport protocols, we instead use the performance of standard TCP as a benchmark for our multi-link implementation. Performance gains achieved through the use of different transport protocols over individual WWAN links should transfer directly to our multi-link environment.

3 Assumptions

We assume each of the CDPD links in the bundle traverses the same wired path. This allows us to remove congestion control from each individual link and deal with the end-to-end congestion in totality (see Section 4.2.3). This assumption is flawed; each CDPD modem in the bundle may be operating in a separate cell[1], so the paths may diverge at some point. Close examination of the topology has shown, however, that this point is well within the

[1] Bell Atlantic (and, to our knowledge, most present CDPD carriers) deployed its CDPD network with one dedicated data channel per cell. It is therefore necessary for each modem to operate in a separate cell to realize any increase in available bandwidth.

CDPD network itself, and therefore insulated from non-CDPD cross traffic.

We further assume that the CDPD modems remain stationary. This is an implementation issue caused by the standard CDPD channel acquisition algorithm, which causes each modem to select the "best" channel. Clearly when they are co-located, roaming causes the modems to converge to the same channel, competing with each other for the same bandwidth. We are currently working to modify the channel selection algorithm to support mobility.

4 Multi-Link Characteristics

The method used for inverse multiplexing IP packets is standard. We stripe each packet across some number (possibly only one) of outgoing links, encapsulating it with a PPP Multi-Link (ML) header [RFC1990]. The resulting *fragments* are then sent using a transport mechanism over IP to the demultiplexor, where they are reassembled. The reconstructed packet is then forwarded to the appropriate destination.

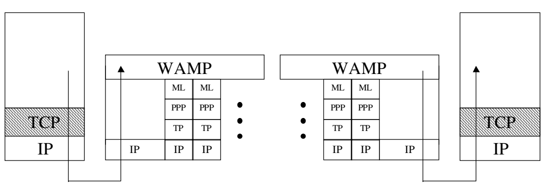

Figure 1: The WAMP Architecture

As can be seen in Figure 1, the end-to-end TCP packets are fragmented by the multiplexor and tunneled through multiple CDPD links using ML PPP over a link layer transfer protocol (TP). An inverse multiplexing strategy of this type has two main components: a fragmentation and scheduling mechanism and a transport protocol. In this section, we examine the possible choices for each, and discuss how the various alternatives affect reliable transport performance.

4.1 Scheduling Techniques

Once packets have been fragmented, the multiplexor must decide how to assign fragments to the available links: that is when and in what order to transmit them. This problem is more commonly studied in the inverse, when multiple links are multiplexed across a single, shared resource. As noted in [1], however, the problem maps directly to our situation. The obvious Round Robin approach allocates fragments amongst all the available links in an ordered fashion. In the long run, Round Robin scheduling provides a perfectly fair distribution of fragments. When these fragments are of roughly the same size, this translates into a balanced spread of bandwidth.

An entire class of scheduling mechanisms attempts to compensate for Round Robin's deficiencies in scheduling packets of varying sizes. Various fair queuing strategies select the appropriate link for the next fragment based upon the fragment's size and the queue lengths at the outgoing interfaces (see, for example, [8]). Unfortunately, many techniques of this flavor require accurate queue length information, which is difficult to obtain in a CDPD network. Furthermore, such techniques often lead to significant packet reordering when used with variable length packets.

A crucial property of WWANs, and CDPD networks in particular, is that while all of the links may be physically identical, their performance at any point in time is highly variable. Since each modem is operating in a separate cell, with possibly widely differing signal characteristics, some variation in channel throughput may be experienced even in the absence of channel contention. Furthermore, if additional CDPD modems are operating on some of the same channels, the link bandwidth available to our modems is reduced.

We propose a novel scheduling technique, similar to Weighted Round Robin, based on the ratio of short-term averages of observed throughput for each of the member links in a bundle. *Link Quality Balancing* dynamically adjusts the MTU of each link in proportion to the available bandwidth. By splitting packets into fragments that can be transmitted in roughly the same amount of time by each link, reassembly can proceed without delay.

In order to ensure traffic is actually distributed as intended, the fragmentation algorithm fragments each packet to ensure that the entire MTU of the current link is utilized before moving to the next link. If the last fragment of the previous packet did not fully utilize the relative MTU of the current link, the first fragment of the new packet is sized to fill the MTU, and scheduled accordingly. Note the two fragments are actually sent separately, so transmission of the previous packet is not delayed. In the case of uniformly sized packets, whose length is a multiple of the link MTU, this scheduling discipline reduces to Round Robin for bundles of identically performing links.

4.2 Link Layer Transport

Unlike many traditional multi-link applications, we do not have exclusive access to the physical links in question. Instead, we are forced to tunnel data over the Internet to the CDPD network, where it is sent across the cellular network to the modems. Empirical evidence shows the selection of an appropriate transport mechanism is critical to achieving acceptable performance. The fundamental design decision is whether to provide reliable transport and/or congestion control.

4.2.1 Reliability

This can be done through a reliable PPP connection, perhaps built using Numbered Mode [RFC1663], or by tunneling PPP over a reliable transport protocol. While arguments can be made for either choice, we claim an unreliable mechanism is superior in the multi-link environment, as it allows us to dispense with link layer congestion control, and reduces jitter in packet reassembly.

If a reliable transport mechanism is selected for each link, it requires a congestion control/avoidance scheme, since retransmitted fragments will travel over the shared Internet for a portion of their journey. Previous work has shown introducing an additional retransmission layer below TCP degrades overall performance [3].

4.2.2 Cached Retransmission

In some cases, however, the path between multiplexor and demultiplexor may be sufficiently lossy as to benefit from a Snoop-like [4] scheme to realize the savings of link-layer retransmission in the absence of a reliable transport mechanism. In such cases, PPP Numbered Mode [RFC1663] can be used. The multiplexor already tags each fragment with a Multi-Link (ML) identifier for reassembly. We propose to extend it with a hash of the flow ID and TCP sequence number, allowing the demultiplexor to associate it with a particular TCP packet even if reassembly fails. Then, if the end-to-end TCP requests a packet retransmission, the demultiplexor can identify exactly those fragments that are necessary to complete reassembly, and requests only their retransmission by referencing the Numbered Mode ID. We term this a *Lost Fragment Request*, or LFR.

For its part, the multiplexor simply caches each fragment. It snoops on the return path, observing ACK sequence numbers. Since TCP only transmits a *window*'s worth of data unacknowledged, the amount of data that needs to be sent before a cached fragment can be verified to have been safely received is bounded and small. Recall the *bandwidth*delay* product of CDPD links is itself small—on the order of two or three packets. Further study of the performance impact of LFRs is ongoing work; the results reported here do not reflect the benefit of fragment caching.

4.2.3 Congestion Control

If retransmissions are triggered only by the end-to-end transport protocol (either in the absence of a reliable link level, or using LFRs as described above), congestion control is unnecessary at the link level, and is effectively dealt with by end-to-end transport protocols.

Under our single path assumption, any congested bottleneck (except the final cell-specific MDBS) encountered by one link is also traversed by the other links. When one link receives notice of congestion (through packet loss), the appropriate response is for all of the links to respond by reducing their rate. Since loss is exposed to the upper level transport protocol, the logical link is determined to be congested, and the transport layer's estimate of the available bandwidth is adjusted appropriately. This adjustment affects all of the links equally (by equally, we mean in proportion to their measured throughput as a function of our scheduling policy).

Care must be taken, however, to ensure one lossy link does not limit the throughput of the bundle. In the case of transient Internet congestion, the individual links drop packets with equal probability. If congestion is occurring in a particular cell, packet losses will be concentrated on the associated link. While the individual fragment losses will indeed cause the end-to-end transport layer to slow, the demultiplexor records the relative loss rates of each link. It exchanges this information periodically with the multiplexor, allowing the scheduler to reduce its usage of each link in proportion to the estimated available channel throughput. As the end-to-end transport protocol continues to probe for bandwidth, the logical link places less demand on the physical link that previously failed, which leads to stable behavior.

4.3 Link Quality Metric

The effectiveness of such an adaptive technique hinges on the selection of a reliable metric for available link throughput. One approach might be to monitor the queue lengths at the outgoing interface, similar to weighted or fair queuing schemes described previously. While effective for the mobile multiplexor (which is directly connected to the CDPD modems), this method is fundamentally flawed if applied at the wired multiplexor. Recall that packets must first traverse some section of the Internet before entering the CDPD network. In this work, we assume the path to every CDPD modem is the same, up until some point well within the CDPD network (where it must diverge to reach separate cells). Therefore any inter-channel variability is only evident in queues far into the network, and would not be exposed at the local interface.

Instead, we passively monitor each link's performance using an extension to PPP's Link Quality Monitoring (LQM) standard [RFC1989]. By exchanging information about the loss rates and perceived throughput of each link, both multiplexors are able to make an informed and independent evaluation of channel performance (due to MAC contention on the cellular up-link and obvious discrepancies in transmitter power between the modem and tower, performance may be markedly different in opposite directions).

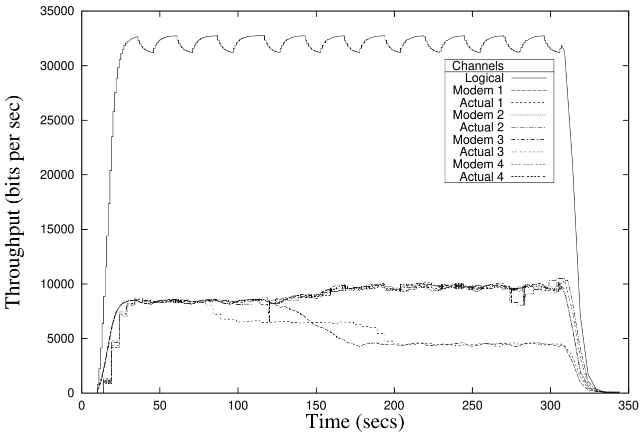

Figure 2: Link Quality Balancing across four CDPD modems

The obvious metric, as in traditional wired networks, is loss rate. Figure 2 shows an approximate Constant Bit Rate (CBR) source transmitting across the WAMP logical link. Initially the traffic is split equally across all four available CDPD links. At approximately T+75s, a fifth modem begins transmitting at a CBR of 6000bps on the same channel as modem 1. Almost immediately the throughput can be seen to drop at the demultiplexor. Due to the deep network buffers, however, no disproportional loss is perceived until T+125s, when excessive fragment loss on modem 1 causes the multiplexor to multiplicatively decrease its utilization of the link by some specified amount, δ.

The decrease in utilization of link 1 causes a proportional increase in the use of links 2-4, since the offered load is steady. The net effect, then, is a multiplicative decrease whose magnitude depends the utilization of the remaining links. WAMP continues to decrease its usage of the lossy link until the error rate returns to within an acceptable threshold, at which point utilization stabilizes.

At T+325s the CBR source terminates, and bandwidth utilization on all links drops off proportionately as traffic decreases. Note that even if the competing CBR source is removed, relative utilization of link 1 may not increase. If the remaining links are not saturated (which, in fact, they are not), and error rates remain tolerable, WAMP has no incentive to continually probe for additional bandwidth on link 1. Only when additional load arrives on the logical link (whether through a new connection or bandwidth probing of the end-to-end transport protocol), which over-saturates the remaining links, does WAMP increase utilization of link 1.

Note that WAMP never actually explicitly increases utilization of a link. Instead, as it decreases utilization of congested links, reliance on alternate links increases proportionately. This allows the end-to-end congestion control algorithms to operate as designed, without additional loss caused by bandwidth probing. While WAMP could generate synthetic traffic with which to probe links, CDPD networks do not support priority queuing, so packet loss is uniform. Hence any additional traffic, synthetic or not, may cause losses in the real traffic.

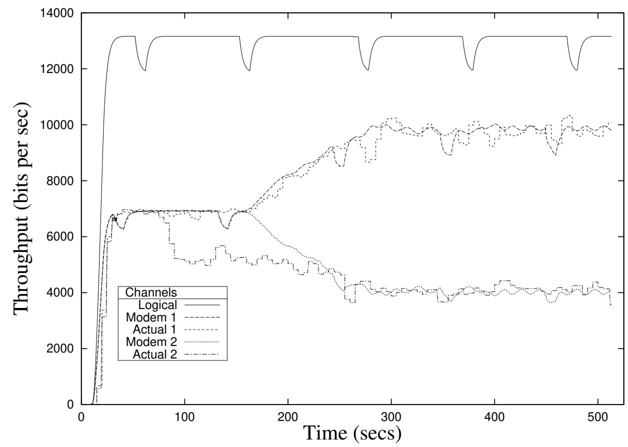

Figure 3: Link Quality Balancing across two CDPD modems

Figure 3 shows a similar experiment using only two links, which produces a much more dramatic effect. In this scenario, the CBR source provides an offered load of 13000bps. At T+75s a third modem begins to generate cross traffic on the same channel as modem 1, using a CBR stream of 5000bps. The decrease in available bandwidth becomes noticeable almost immediately, with the first loss event occurring at T+175s. WAMP then begins to decrease the utilization as before, leading to stable long-term behavior.

5 TCP Pathologies

We now turn our attention to the performance of a reliable transport protocol, TCP in particular, over our Multi-Link channel. This section identifies the difficulties in adapting link utilization for TCP flows, and suggests methods for improvement

5.1 Deep Buffering

TCP adapts to drastic changes in bandwidth by detecting losses. The assumption is that in the face of sudden changes in available bandwidth, packets will queue up and be dropped. Unfortunately, in a CDPD environment with few flows, the buffers are very deep compared to the receive windows. So sudden changes in bandwidth are often not detected by losses, but instead simply space out the ACKs from the receiver, which slows TCP's transmission rate down to the speed of the ACKs.

While TCP's bandwidth probing algorithm will continue to increase the window size one packet per (its estimation of the) RTT (which, as discussed in [3] and [5], is likely to be much larger than accurate), the decreased rate of ACKs will cause this to take longer than it should. Furthermore, the next loss event will occur with a smaller window (since TCP's congestion avoidance algorithm converges), so the network buffers are even more likely to be able to absorb the excessive traffic for long enough to slow the ACK clock down.

Figure 4 shows a single TCP sender operating over a WAMP link with two separate channels. The characteristic saw-tooth pattern of TCP's window probing algorithm is readily apparent. As before, at time T+125s, an additional modem began sending a CBR flow at 5000 bps on the same cellular channel as modem 2. The drop in available bandwidth is noticeable almost immediately, as a loss event occurs, and the saw-tooth turns downward. Finally, at T+175s the loss rates become disproportionate, and the multiplexor begins to decrease its utilization of link 2. Notice this occurs two probing periods later, after TCP has markedly decreased its sending rate due to the slower rate of returning ACKs.

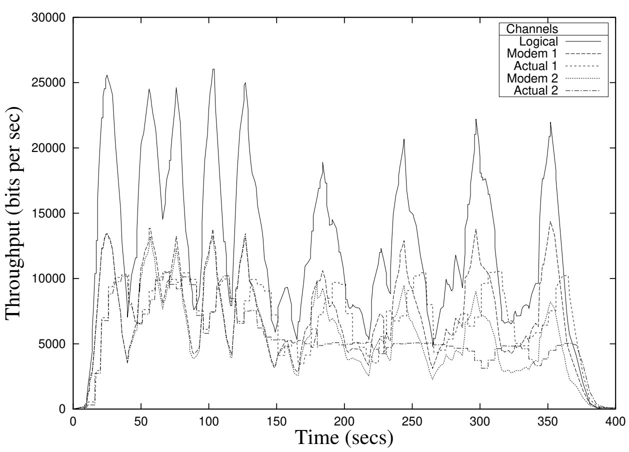

Figure 4: Link Quality Balancing across two links for a single TCP sender.

As WAMP continues to decrease utilization of the congested link, TCP can be seen to succeed in trying larger and larger window sizes. While this process will eventually converge, due to the stability properties of TCP's linear increase, multiplicative decrease congestion avoidance, it occurs over a very large time span. The transfer concludes at T+375s with an overall throughput approximately 85% of the theoretical maximum (if WAMP had adjusted instantaneously to the available channel bandwidth).

While this effect disappears rapidly as additional flows are established (since they are each asymmetrically probing for additional bandwidth), it is reasonable to assume the number of simultaneous flows will be small, given the low link bandwidth. This motivates our search for an alternate metric that could enable rapid adaptation to changes in available link throughput, without waiting for the network queues to admit a loss event.

Even if such a metric existed, considerable control issues arise when one attempts to make sudden, significant changes. In order to avoid the TCP problem, the decrease would have to be significant. Clearly false triggers would be catastrophic. Furthermore, care would need to be taken to ensure sufficient time for stabilization before additional adaptations were made. Relative short-term throughput averages, inter-packet separation averages, and inter-packet deviations have all initially proved too unstable for use. We are continuing to investigate dampening methods that might allow their use.

5.2 Packet Reordering

Depending on the scheduling policy in use, the top level TCP stack often may receive packets out of order due to the relative delay of individual channels. Our study of individual CDPD links shows that reordering frequency is similar to that expected in the general Internet [12], but *very rarely* by more than one packet.

If, as a general rule, we assume packets will be reordered at most once on each link, we can then examine the effect of striping fragments. Some scheduling policies then admit bounds on reordering. For our modified Round Robin techniques, reordering on the logical link is clearly just a function of packet size versus the number of links.

5.2.1 Fast Retransmit Modifications

TCP's fast retransmit algorithm exists to trigger rapid retransmission of lost packets. As specified, the receipt of three or more out of order packets signals packet loss, causing immediate retransmission. The value of three DUPACKs was arrived at empirically, and seems to function well for the wired Internet [12].

In our case, however, we can use any bounds provided by our scheduling mechanism to modify the retransmit algorithm. If, for instance, we are guaranteed never (actually *with high probability*, based on our assumption the individual links only reorder by one packet) to receive more than one out of order packet, we could reduce the number of DUPACKs to two, thereby causing a faster retransmission, improving throughput. If, on the other hand, our scheduling algorithm were such that packets were regularly delivered more than two packets out of order, the standard fast retransmit algorithm would incorrectly assume packet loss and cause spurious retransmissions. In this case, it is clearly beneficial to adjust the number of DUPACKs up to the higher value.

Given the large timeout delays caused by not triggering the fast retransmit algorithm, it may be desirable to be a bit more aggressive in setting the DUPACK level than one would be on a wired network. For the purposes of this paper, however, we choose not to modify the algorithm to preserve standard TCP semantics.

5.3 Loss Amplification

The major drawback of multi-link operation is that all of the characteristics of the underlying channel, both positive and negative, are amplified. Take the case of packet loss for example. The probability of successful packet transmission for the logical link, p_t, can be expressed in terms of the loss probabilities for each of the n constituent links, p_i: $p_t = 1 - (1 - p_1)(1- p_2)\ldots(1- p_n)$. Splitting a packet across four links with 10% packet loss probability results in an incomplete packet 35% of the time. This lowest common denominator effect becomes increasingly evident when conditions deteriorate.

This effect implies there exists a point at which a link's membership in the bundle detracts from the overall bundle's performance. Consider, for example, the case of three links operating optimally, with one link dropping packets at a rate of 50% (not unheard of during peak hours in our CDPD network). Regardless of the retransmission mechanism used on the lossy link, the added delay caused by the required retransmission of lost data fragments causes TCP throughput on the logical link to be lower than it would be if it just used the three perfect links.

We are already monitoring link quality for use in our scheduling mechanism, so the data is available. The difficult part is determining at exactly what point to remove a link. There are obviously many factors, including the relative performance of a link to the others in the bundle, the absolute performance of the link, and so on. It is not clear how to calculate its exact value on-line. Instead, we suggest empirically computing a reasonable default threshold, and basing a link's membership in the bundle on that specified value. There is no danger in setting the value too low, as the resulting performance is no worse than the unmodified system. Care needs to be taken, however, not to set the value too high. Determining the optimum value for this threshold is ongoing work.

6 Implementation

The inverse multiplexor described herein was deployed using Sierra Wireless MP200 CDPD modems. The modems are connected via a Specialix SIO serial board to a PII/400 running FreeBSD 2.2.7. The other end of the virtual link is a Pentium-class machine running FreeBSD 3.2. This machine connects directly to the Bell Atlantic CDPD network over a 56K PVC Frame-Relay circuit.

WAMP is implemented as a user-level PPP daemon that runs PPP [RFC1661] with HDLC framing [RFC1662] over UDP. The CDPD modems operate on Bell Atlantic Mobile's digital cellular network. During initialization, the mobile multiplexor scans for available CDPD channels, and allocates the four best channels to the MP200s. Each modem locks on its prescribed channel, and begins PPP link establishment negotiation with the wired multiplexor/demultiplexor.

6.1 Tuning

Several aspects of our inverse multiplexing algorithm require constants to be set at appropriate values for the current operating environment.

6.1.1 LQM Interval

The current WAMP implementation exchanges LQM information every 5 seconds. Each LQM packet is 56 bytes, so this translates to an overhead of ~10 bytes/sec per link, on the order of 1% of the channel capacity. Shorter intervals not only increase overhead, but also provide less stability in the scheduling mechanism. Internet traffic is by nature bursty. When one considers the typical user's traffic on the CDPD network (mail clients, web browsers, and the like), it can be expected to be especially bursty. We clearly don't want to be too quick to predict a link's performance.

Given current CDPD speeds, one 576-byte packet (our MTU) requires on the order of half a second, five seconds allows for approximately 10 packets to be transmitted between LQM exchanges. Loss rates become less useful at lower intervals, as the granularity becomes too coarse. Furthermore, when multiple flows are active, we are likely to have significant buffering, so any changes will take several seconds to propagate through the channel anyway.

6.1.2 Loss Threshold & Decrease Delta

Loss thresholds and decrease deltas must be tightly correlated if stability is to be achieved. The higher the loss rate threshold, the slower WAMP adapts to long-term changes in available throughput. At the same time, it will be more resilient to transient bursts. Due to the asymmetric nature of CDPD's channel access, different values are required for each direction. For the down channel, where there is no media contention, empirical evidence suggests that during off hours, most bursts can be accommodated in the network and modem buffers, and any loss exposed to the link layer is significant. For the up channel, on the other hand, losses are much more common, due to the CSMA/CD channel access method.

At present, we manually adjust the loss threshold to current operating conditions, but are investigating methods for separating load-induced loss with load-independent loss. We have found thresholds of 2% during evening hours and 5% during peak hours work reasonably well.

Optimum values of the decrease parameter, δ, however, have proved to be considerably more stable over long time frames. A decrease factor of .85 is the largest factor that leads to stable behavior with reasonable reaction times. While this is a considerably smaller decrease than found in TCP Reno, one must recall that scheduling is relative. Hence any decrease in one link leads to an increase in the others. Small values of δ lead to huge oscillations in link scheduling, especially when few links are involved, as well-behaved links are driven into overload by the increased load caused by the sudden decrease in utilization of a lossy link.

7 Conclusions

We have presented an adaptive approach to inverse multiplexing reliable transport protocols in WWAN environments based upon three key observations. First, when the component links share a path to the CDPD network, we note that congestion control is appropriately and effectively handled by the upper level transport protocol, and is impeded by additional control at the link layer. This does not, however, imply the individual links could not provide some form of enhanced reliability, as proposed by our LFR scheme.

Secondly, we argue that optimum fragment scheduling requires knowledge of the current channel characteristics of each link. By adapting fragment size to the current effective throughput of each link, we enable packet reassembly and delivery with a minimum of delay, thereby preventing slow links from throttling the performance of the entire bundle.

Finally, while obtaining accurate measurements of instantaneous channel transmission delay is problematic, we note that relative channel performance is sufficient, since the transport protocol's congestion control algorithm will appropriately increase or decrease bandwidth utilization. LQM loss data sent from the receiver at regular intervals is sufficient to maintain a stable short-term approximation of relative throughput. Determining optimum sampling intervals is ongoing work.

8 Acknowledgements

Rusty Hemenway and his colleagues at Bell Atlantic Mobile provided invaluable assistance in setting up the experimental testbed. Ty Sealy and Ron Wiken aided in the installation of the mobile equipment. We are indebted to John Wroclawski and Hari Balakrishnan for their constructive criticism and willingness to support this work.

9 References

[1] H. Adiseshu, G. Parulkar, and G. Varghese. A Reliable and Scalable Striping Protocol. In *Proc.of ACM SIGCOMM*, August 1996

[2] A. Bakre and B. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. In *Proc. of ICDCS*, May 1995.

[3] B. Bakshi, P. Krishna, N. Vaidya, and D. K. Pradhan. Improving Performance of TCP over Wireless Networks. In *Proc. of ICDCS*, May 1997.

[4] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP Performance over Wireless Networks. In *Proc. of ACM MOBICOM*, November 1995.

[5] H. Balakrishnan, V. Padmanabhan, and R. Katz. The Effects of Asymmetry on TCP Performance. In *Proc. of ACM/IEEE MOBICOM*, September 1997.

[6] Bandwidth ON Demand Interoperability Group. Interoperability Requirements for Nx56/64 kbit/s Calls, September 1992.

[7] C. Brendan, S. Traw, and J. Smith. Striping within the Network Subsystem. *IEEE Network*, 1995.

[8] A. Demers, S. Keshav, and S. Shenker. Analysis of a Fair Queuing Algorithm, *Journal of Internetworking Research and Experience*, September 1989.

[9] R. Durst, G. Miller, and E. Travis. TCP Extensions for Space Communications. In *Proc. of ACM/IEEE MOBICOM*, September 1996.

[10] Electronic Industry Alliance/Telecommunications Industry Association. IS-95: Mobile Station-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System, 1993.

[11] M. Mehta and N. Vaidya. Delayed Duplicate-Acknowledgements: A Proposal to Improve Performance of TCP on Wireless Links. Technical Report, Computer Science Dept., Texas A&M University, February 1999.

[12] V. Paxson. End-to-End Internet Packet Dynamics. In *Proc. of ACM SIGCOMM,* September 1997.

[13] M. Rahnema. An Overview of the GSM System and Protocol Architecture. *IEEE Communications Magazine*: 31, April 1993.

[14] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks. In *Proc. of ACM/IEEE MOBICOM*, August 1999.

[15] Wireless Data Forum. Cellular Digital Packet Data System Specification, Release 1.1, January 1995.