Adaptive Timeout Policies for Wireless Links

George Xylomenos and Christos Tsilopoulos
xgeorge@aueb.gr and p3000173@dias.aueb.gr
Mobile Multimedia Laboratory
Department of Informatics
Athens University of Economics and Business
Patision 76, Athens 104 34, Greece

***Abstract*— A considerable body of evidence indicates that the use of reliable link layer protocols over error prone wireless links dramatically improves the performance of Internet protocols and applications. While traditional link layer protocols set their timeout values assuming that they fully control the underlying link, some wireless networks allow multiple link layer sessions to co-exist over the same link. Since the optimal timeout values for a reliable link layer protocol depend on the available bandwidth, with dynamic link sharing such a protocol should ideally adapt its timeout values accordingly. We have thus designed an Adaptive Selective Repeat protocol that modifies its timeout values based on the policy used by TCP. We compare the performance of Web Browsing over Selective Repeat when using our adaptive timeout scheme with a range of parameters, against a manually tuned fixed timeout version. Our measurements show that these adaptive timeout policies outperform the fixed one, regardless of the level of contention, and that the best adaptive timeout policy in this setting is not the one used by TCP.**

I. Introduction

While wireless networks are becoming increasingly popular as access network to the Internet, many researchers have found that the error prone nature of wireless links dramatically degrades the performance of popular applications such as Web Browsing [1]. A direct way to hide wireless link deficiencies is to employ a reliable link layer protocol; such protocols have been shown to improve the performance of higher layer protocols and applications, even when unaware of the nature of the higher layers used [2].

Traditional link layer protocols assume that they have full control of the underlying link when choosing their parameters, including timeouts. Due to the higher bandwidth offered by emerging wireless networks however, it is becoming common for multiple users and/or applications to dynamically share a single wireless link. For example, in the *Universal Mobile Telecommunications System* (UMTS) a single physical channel is shared among the link layer sessions of different users and/or applications. Even in links controlled by a single user, it is desirable for multiple such protocols to co-exist, so as to serve different applications.

In this paper we examine the impact of link layer timeout policy on application performance over shared links. In Section II we provide background information on Internet protocol and application performance over wireless links. In Section III we discuss the problems faced by a Selective Repeat protocol in this environment and propose an adaptive timeout scheme. Section IV describes our simulation setup for the performance evaluation. In Section V we show Web Browsing performance when operating in isolation over a wireless link, while in Section VI we show Web Browsing performance when contending for the link with a Media Distribution application. Our measurements show that the adaptive timeout policies outperform the fixed one.

II. Background and Related Work

The *Internet Protocol* (IP) offers an unreliable packet delivery service. Many real-time applications use the *User Datagram Protocol* (UDP) for direct access to IP, handling error, flow and congestion control themselves. Most other applications prefer delegating these tasks to the *Transport Control Protocol* (TCP) which offers a reliable byte stream service. TCP segments the application data stream into packets and reassembles it at the receiver. The receiver generates *acknowledgments* (ACKs) for segments received in sequence, returning duplicate ACKs for out of sequence ones. The next unacknowledged segment is retransmitted after receiving 3 duplicate ACKs or when a timeout occurs.

Since wired links are extremely reliable, TCP assumes that all packet losses are due to congestion, thus after a loss it abruptly reduces its transmission rate, and then gradually increases it so as to probe the network. Unfortunately, this means that losses due to wireless errors are mistaken for congestion signals, causing TCP to dramatically reduce its transmission rate [1]. Many modifications have been proposed to improve TCP performance over wireless links, but they all have two drawbacks: they require modifications to end hosts throughout the Internet and they can only retransmit lost data over the (possibly long) end-to-end path.

A simpler approach is to employ a reliable link layer protocol over the wireless link to locally hide wireless errors from TCP. An early proposal customized to TCP *snoops* inside each TCP stream at the access point bridging the wired and wireless parts of the path and retransmits lost segments when duplicate ACKs arrive, hiding them from the sender to avoid end-to-end recovery [3]. However, later work shows that TCP performance can be enhanced even with standard, TCP unaware, reliable link layer protocols [2].

It is important to note that not *all* Internet applications require full reliability. While TCP based applications perform better with reliable link layers, delay sensitive UDP based applications often prefer faster, albeit limited, error recovery.

Reliable link layers should therefore expect to co-exist with other link layers over the same link, even if it is controlled by a single user. These competing link layers cause the bandwidth available to a reliable link layer protocol, and therefore its effective *Round Trip Time* (RTT), to vary as sessions start and stop. This leads to a problem when setting timeouts: timeouts should be higher than the RTT, to prevent premature retransmissions, but not too high, to prevent the protocol from stalling until a timeout occurs. As the RTT varies, the timeout should ideally follow.

A TCP aware link layer must set its timeouts by mimicking TCP retransmission behavior, so as to retransmit lost packets before TCP does [3]. This approach may be unsuitable for other higher layer protocols though, and it may not even be the most efficient for a link layer protocol. In this paper we therefore propose and evaluate a range of TCP based but TCP unaware adaptive timeout policies.

An alternative solution to the problems introduced by contention for the link is employed by the *Radio Link Control* (RLC) protocol used in UMTS networks in its Acknowledged Mode [4]. This protocol relies solely on status information from the receiver to trigger retransmissions. Since status reports may be lost, either the RLC sender or the RLC receiver is configured to periodically probe for or return status reports, respectively. Thus, RLC uses timers to ensure the periodic exchange of status information.

III. Adaptive Selective Repeat

In our past research we have found the Selective Repeat protocol to offer excellent performance for TCP based applications, including Web Browsing [2]. Since Selective Repeat is TCP unaware, it may be able to improve the performance of other higher layer protocols and applications that can benefit from a reliable link layer. In Selective Repeat, the sender transmits link layer frames in sequence within a transmission window of N frames, buffering them for possible retransmission. The receiver accepts frames within a reception window also of N frames; if a frame is received in sequence, it is delivered to the higher layer, the window slides upwards and an ACK is returned to the sender, confirming reception of all frames up to the one delivered. When the sender receives an ACK, it drops the frames covered by it and also slides its window upwards.

When a frame arrives out of sequence at the receiver, it is buffered but not delivered. The gap in the sequence indicates that some frames were lost, so a *negative acknowledgment* (NACK) is returned for each missing frame to the sender. The sender retransmits each frame indicated by a NACK. When a missing frames arrives, the receiver delivers to the higher layer all frames that are now in sequence, it slides its window upwards and it returns an appropriate ACK. In our Selective Repeat implementation, we delay returning each ACK for a short interval with the aim of piggybacking it into a data frame traveling in the reverse direction. If this interval expires, the ACK is sent as a separate frame. NACKs are always sent as separate frames.

If some ACKs and/or NACKs are lost, the sender may exhaust its transmission window and be unable to proceed, as the lack of receiver feedback prohibits sliding the transmission window upwards. To prevent this, the sender starts a timer after sending each frame. If a timeout occurs before an ACK has arrived for the frame, the frame is retransmitted [5]. The value of the timeout must be large enough to avoid retransmissions before an ACK had the chance to arrive, and small enough to prevent the sender from stalling too long; ideally, it should be slightly higher than the RTT of the link. The problem with link sharing is exactly that the effective RTT changes as competing sessions start and stop, modifying the available bandwidth. It is therefore desirable for a reliable link layer protocol to be able to adapt its timeouts, keeping them slightly higher than the RTT.

To achieve this, we modified Selective Repeat so that it tracks the effective RTT. For every packet transmitted or retransmitted, the sender stores its transmission time. When an ACK arrives for the packet, the difference between the current time and the transmission time provides an RTT $sample$. We use these samples to update smoothed estimates for the RTT, $srtt$, and its variance, $srttvar$, with:

$$srtt = 0.875 * srtt + 0.125 * sample$$

$$srttvar = 0.75 * srttvar + 0.25 * (sample - srtt)$$

The equations and smoothing factors used are exactly the same as those used by TCP, therefore these calculations can be performed efficiently with integer arithmetic [6]. As the effective RTT fluctuates, the estimators follow it in a smoothed manner: they react to changes with a time lag and are not dramatically affected by sporadic extreme values. We have tested a wide range of values for the smoothing factors and found that the TCP values provide good performance under most conditions. After updating the estimators, we calculate the new timeout value, $rtxto$, with:

$$rtxto = \alpha * srtt + \beta * srttvar$$

While this equation is also borrowed by TCP, we found that the actual values use by TCP, that is, $\alpha = 2$ and $\beta = 4$, led to many premature timeouts. We therefore tested an extended set of values for α and β, choosing a subset for presentation below. We refer to these adaptive timeout policies as $\alpha + \beta$; for example, the TCP policy is referred to as $2 + 4$.

Our adaptive scheme calculates samples from every acknowledged packet, with three exceptions. First, NACKs are not used to calculate samples, since they do not reflect reception of the frame indicated but of a following one. Second, when an ACK covers multiple frames, only the last frame acknowledged is used to calculate a sample, since the previous ones may have been received long ago. Third, when duplicate ACKs arrive, only the original ACK provides a sample; the following ones are mere repetitions.

Our adaptive timeout policy diverges from TCP in that the timeout value is *not* modified after a timeout, unlike TCP which doubles its current timeout value whenever a timeout occurs. While TCP assumes that a timeout indicates congestion, hence consecutive timeouts indicate very serious congestion, timeouts over a wireless link are most likely due to wireless losses which do not impact the RTT.

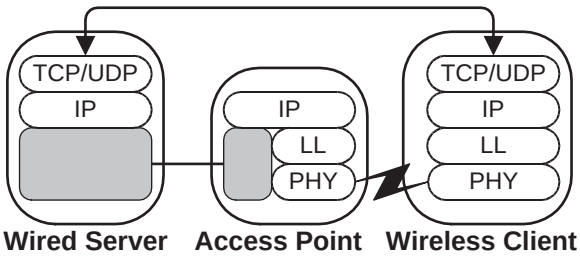

Fig. 1. Simulated network topology

IV. Simulation Setup

The performance results reported below are based on simulations with ns-2 [7], extended with additional wireless links, link layer protocols and applications [8]. Each test lasted for 2000 s and was repeated 30 times with different random seeds. The results shown reflect average metrics across these runs, as well as their 95% confidence intervals. The simulated topology is shown in Figure 1: a Wired Server communicates with a Wireless Client via an Access Point. In all applications tested, the server was located at the wired end of the network and the client at the wireless end. The wired link has a bandwidth of 10 Mbps and a propagation delay of 1 ms; the same conclusions are reached when a 2 Mbps / 50 ms wired link is used instead.

The wireless link has a bandwidth of 64 Kbps, a propagation delay of 50 ms and uses a frame size of 250 bytes plus a header, typical characteristics for cellular links, where bit interleaving inflates propagation delay. To avoid packet fragmentation, applications also used 250 byte packets. Two error models were used for the wireless link. In the *Uniform* error model each frame may be independently lost with a probability of 1.5%, 2.5%, 5.4% or 9.8%. In the *Two State* error model the link can be either in a good state, with a bit error rate of 10^{-6}, or in a bad state, with a bit error rate of 10^{-2}. Both states have exponential durations, with the average duration of the good state being 10 s and the average duration of the bad state being 100 ms, 200 ms, 500 ms or 1000 ms. With these parameters the average *Frame Loss Rate* (FLR) of the Two State model is 1.5%, 2.5%, 5.4% or 9.8%, matching the FLRs of the Uniform model. The error processes in each link direction were identical but independent. As a baseline, we also show results with no errors.

To evaluate our protocol we used Web Browsing, the most popular Internet application [9], over TCP Reno with 10 ms granularity timers. In Web Browsing the client-server interaction consists of *transactions*: the client requests a Web page from a server, the server returns the page which contains pointers to embedded objects, the client requests each embedded object, and the server returns them, ending the transaction. The next transaction begins when the client requests another page. The ns-2 HTTP module provides empirical distributions for request, page and embedded object sizes, as well as for the number of objects per page [9]. Only one transaction was in progress at any time with no pauses between transactions. The performance metric was Web Browsing throughput, defined as the amount of *application* data transferred from the server to the client divided by time taken. Client requests influence throughput only by introducing delays. All results shown reflect the state at the end of the last completed transaction during the simulation.

Contention is provided by a UDP based real-time Media Distribution application simulating a lecture where a speaker sends audio / video to an audience including the wireless client. The speaker alternates between *talking* and *silent* states with exponential durations, averaging 1 s and 1.35 s, respectively [10]. Packets are transmitted isochronously at the talking state with a rate of 56 Kbps, consuming 87.5% of the available bandwidth in that state, but only 37.5% on average. Thus, the bandwidth available for Web Browsing is abruptly modified whenever Media Distribution changes state. To assess the impact of Selective Repeat retransmissions on Media Distribution, we also measured the average end-to-end delay of media packets.

V. Performance Without Contention

In this section we examine the performance of Web Browsing when operating without contention over the wireless link. Figure 2 shows the Web Browsing throughput achieved by five link layer protocols over a Uniform error model. The *Raw Link* curve shows performance without error control. The *Adaptive* curves show Selective Repeat performance with the timeouts calculated using the $2+4$, $3+2$ and $4+0$ policies. Finally, the *Fixed* curve shows the performance achieved with a fixed timeout of 1.1 s, a value found to provide the best tradeoff between the cases with and without contention. All adaptive schemes use an initial RTT estimate and timeout value of 1.1 s, and an initial RTT variance estimate of 0 s, but even much higher initial estimates do not significantly affect their performance, as the estimators quickly converge to appropriate values.

More details are provided in Table I, where for each FLR we show the throughput achieved by the Fixed variant, as well as the improvements provided over it by each Adaptive variant. The table includes results for the $4+2$ and $4+4$ Adaptive policies which are omitted from the figure to reduce clutter. The Adaptive variants outperform the Fixed one by margins of between 2.5% and 17%. Regarding the relative ranking of the Adaptive variants, the best performance is offered by the $3+2$ policy, closely followed by the TCP $2+4$ policy. The more relaxed variants lead to timeout values that are rather high, thus reducing performance.

On the other hand, Figure 3 shows the Web Browsing throughput achieved over the Two State error model, with additional details provided in Table II. Note the difference in scale with the previous figure. The Adaptive variants again outperform the Fixed one by margins of between 0.5% and 4%. The sole exception is the standard TCP $2+4$ policy which is either marginally better or considerably worse than the Fixed one. Regarding the other Adaptive variants, the more relaxed variants work best, especially at higher FLRs; the $4+0$ policy seems to be the best compromise.

Overall, the Web Browsing throughout results without contention indicate that the Adaptive protocol outperforms its

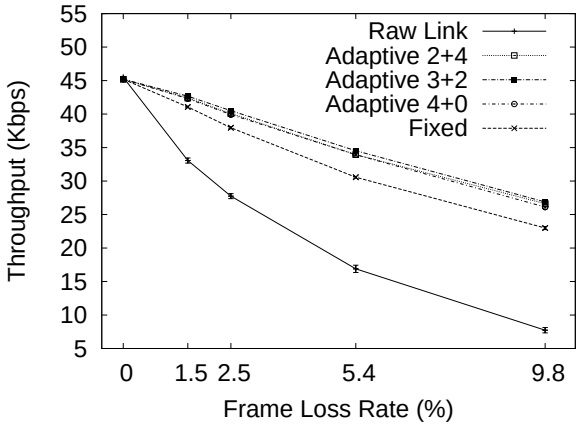

Fig. 2. Web Throughput (Uniform)

TABLE I
THROUGHPUT IMPROVEMENT (UNIFORM)

FLR	Fixed	Improvement over Fixed (%)				
(%)	(Kbps)	2+4	3+2	4+0	4+2	4+4
1.5	41.1	3.2%	3.9%	3.0%	2.8%	2.5%
2.5	37.9	5.6%	6.7%	5.2%	4.6%	4.4%
5.4	30.6	11.1%	12.9%	11.0%	9.2%	7.1%
9.8	23.0	15.9%	17.0%	13.6%	9.5%	5.8%

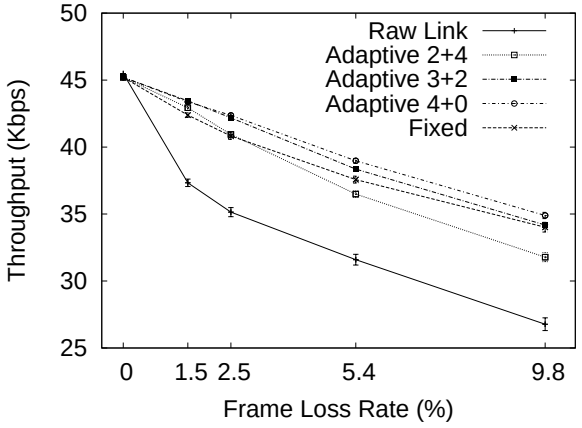

Fig. 3. Web Throughput (Two State)

TABLE II
THROUGHPUT IMPROVEMENT (TWO STATE)

FLR	Fixed	Improvement over Fixed (%)				
(%)	(Kbps)	2+4	3+2	4+0	4+2	4+4
1.5	42.4	1.3%	2.5%	2.4%	2.3%	2.1%
2.5	40.8	0.3%	3.3%	3.8%	3.4%	3.5%
5.4	37.6	-2.9%	2.1%	3.7%	3.6%	4.0%
9.8	34.0	-6.6%	0.5%	2.6%	2.7%	2.0%

Fixed counterpart for a wide range of adaptive policies. And in contrast to the Fixed variant which needs manual tuning, the Adaptive variant automatically adapts to the underlying link. Another observation is that the TCP $2 + 4$ policy is not only suboptimal in all cases, it is actually worse than the Fixed one in some situations, indicating that the α and β values used by TCP are not universally applicable.

VI. PERFORMANCE WITH CONTENTION

In this section we examine the performance of Web Browsing when contending with Media Distribution. In this case, higher layer packets are checked as they enter the link layer and are then processed by two independent link layer modules. The UDP based Media Distribution always uses the *Raw Link* scheme, that is, no error control. The TCP based Web Browsing uses one of the schemes discussed above. The two link layer modules share the link on a first-come first-served basis. Contention only occurs in the direction from the server to the client, since Media Distribution is unidirectional; this is however the same direction in which we measure Web Browsing throughput.

Figure 4 shows Web Browsing throughput performance with contention from Media Distribution over a Uniform error model, with additional details provided in Table III. Even with contention, the Adaptive variants outperform the Fixed one by margins of between 0.8% and 13.2%. The best performance among the Adaptive variants is offered again by the $3 + 2$ policy, closely followed by the $4 + 0$ policy, indicating that a slightly more relaxed policy works well with contention. The $2 + 4$ policy is good, but not optimal.

On the other hand, Figure 5 shows Web Browsing throughput with contention over a Two State error model, with additional details provided in Table IV. The Adaptive variants outperform the Fixed one by margins of between 4.3% and 13.8%; the sole exception is the TCP $2 + 4$ policy, as in the no contention case. Again, the Two State model favors the more relaxed adaptive policies, more so at higher FLRs; the $4 + 0$ policy offers a good compromise.

We conclude by examining the impact of retransmissions on Media Distribution delay. In general, whenever a frame is retransmitted by Selective Repeat, it increases contention for the link and, therefore, the delay experienced by Media Distribution packets. Figure 6 shows the Media Distribution Delay introduced by each reliable link layer protocol over a Uniform error model. Note that delay *decreases* with higher FLRs when Raw Link is used, reflecting reduced Web Browsing performance. Interestingly, as shown in Table V, the Adaptive variants outperform the Fixed one by a margin of between 3.3% and 30.9%, with the exception of the $2 + 4$ policy, indicating that the Adaptive policies introduce fewer retransmissions than the Fixed one.

Similar observations can be made from Figure 7 which shows Media Distribution Delay over a Two State error model. Again, as shown in Table VI, the Adaptive variants outperform the Fixed one by a margin of between 7.5% and 32.2%, with the exception of the $2 + 4$ policy. This means that with both error models the Adaptive variants introduce lower

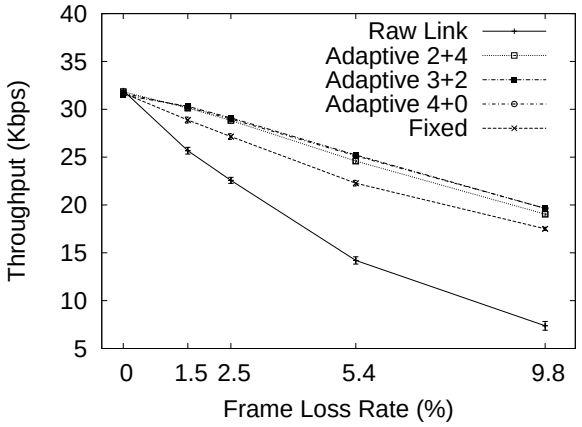

Fig. 4. Web Throughput (Uniform)

TABLE III
THROUGHPUT IMPROVEMENT (UNIFORM)

FLR	Fixed	Improvement over Fixed (%)				
(%)	(Kbps)	2+4	3+2	4+0	4+2	4+4
1.5	28.9	4.3%	5.0%	4.6%	4.3%	4.2%
2.5	27.1	6.2%	7.2%	6.8%	6.2%	5.5%
5.4	22.3	10.4%	13.2%	12.8%	11.3%	9.6%
9.8	17.5	8.7%	12.1%	12.2%	7.2%	0.8%

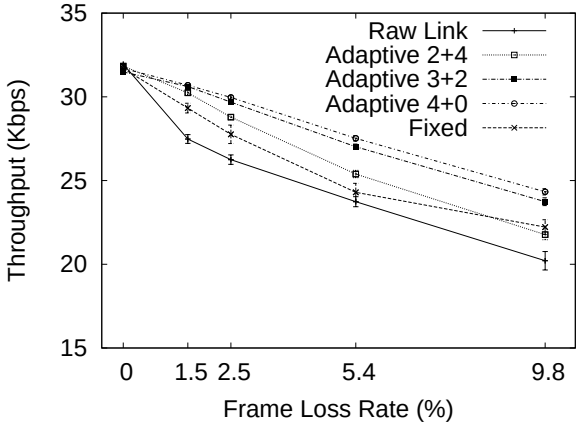

Fig. 5. Web Throughput (Two State)

TABLE IV
THROUGHPUT IMPROVEMENT (TWO STATE)

FLR	Fixed	Improvement over Fixed (%)				
(%)	(Kbps)	2+4	3+2	4+0	4+2	4+4
1.5	29.3	3.1%	4.4%	4.6%	4.3%	4.3%
2.5	27.8	3.7%	6.9%	7.9%	7.8%	7.5%
5.4	24.3	4.5%	11.2%	13.3%	13.6%	13.8%
9.8	22.2	-2.0%	6.8%	9.5%	10.7%	10.7%

delays than the Fixed one, while also providing higher Web Browsing throughput. The standard TCP $2 + 4$ policy is the sole exception, albeit only at the highest FLR.

VII. CONCLUSIONS

In this paper we have discussed the problems faced by reliable link layer protocols when sharing a wireless link with competing link layer sessions. In order to overcome these problems, we have proposed an Adaptive Selective Repeat protocol that dynamically sets its timeouts based on prevailing conditions. Our protocol uses a TCP based scheme to continuously estimate current link conditions and set its timeouts accordingly. Our measurements indicate that our Adaptive Selective Repeat protocol outperforms its Fixed variant, regardless of the level of contention, the frame loss rate and the underlying wireless error model, for a wide range of adaptive policies. At the same time, the Adaptive variants introduce lower delays for competing applications.

Our measurements also show however that while the general policy used by TCP works very well, the actual expression used by TCP is *not* optimal in this setting. We have found instead that the $3 \times srtt + 2 \times srttvar$ policy is optimal with the Uniform error model and the $4 \times srtt + 0 \times srttvar$ policy close to optimal with the Two State error model; in contrast, the standard TCP expression $2 \times srtt + 4 \times srttvar$ is suboptimal, and may even be worse than the Fixed policy. In the vast majority of cases however, all the Adaptive variants tested outperformed the manually fine tuned Fixed one, without any need for manual tuning themselves.

REFERENCES

[1] G. Xylomenos and G. C. Polyzos, "Internet protocol performance over networks with wireless links," *IEEE Network*, vol. 13, no. 5, pp. 55–63, July/August 1999.
[2] G. Xylomenos and G. C. Polyzos, , "A multi-service link layer architecture for the wireless Internet," *International Journal of Communication Systems*, vol. 17, no. 6, pp. 553–574, 2004.
[3] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," in *Proc. of the ACM SIGCOMM '96*, 1996, pp. 256–267.
[4] 3rd Generation Partnership Project (3GPP), "Radio Link Control (RLC) protocol specification (Release 6)," Technical Specification 25.322, V6.2.0, December 2004.
[5] P. T. Brady, "Evaluation of multireject, selective reject, and other protocol enhancements," *IEEE Trans. on Communications*, vol. 35, no. 6, pp. 659–666, June 1987.
[6] V. Jacobson, "Congestion avoidance and control," in *Proc. of the ACM SIGCOMM '88*, August 1998, pp. 314–329.
[7] UCB/LBNL/VINT, "Network Simulator - ns (version 2)," Available at http://www.isi.edu/nsnam.
[8] G. Xylomenos, "Multi service link layers for ns-2," Available at http://www.mm.aueb.gr/~xgeorge/codes/codephen.htm.
[9] B. A. Mah, "An empirical model of HTTP network traffic," in *Proc. of the IEEE INFOCOM '97*, 1997, pp. 592–600.
[10] S. Nanda, D. Goodman, and U. Timor, "Performance of PRMA: a packet voice protocol for cellular systems," *IEEE Trans. on Vehicular Technology*, vol. 40, no. 3, pp. 584–598, 1991.

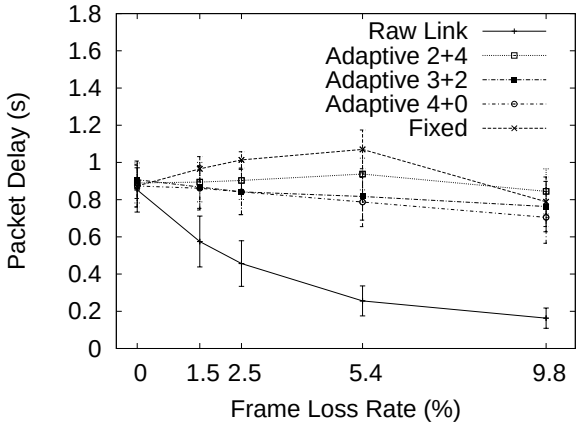

Fig. 6. Media Distribution Delay (Uniform)

TABLE V
DELAY IMPROVEMENT (UNIFORM)

FLR	Fixed	Improvement over Fixed (%)				
(%)	(sec)	2+4	3+2	4+0	4+2	4+4
1.5	0.96	7.4%	10.1%	10.8%	11.3%	11.2%
2.5	1.01	10.9%	17.0%	16.9%	16.6%	18.1%
5.4	1.07	12.4%	23.7%	26.4%	27.4%	30.9%
9.8	0.79	-7.1%	3.3%	10.6%	20.6%	29.3%

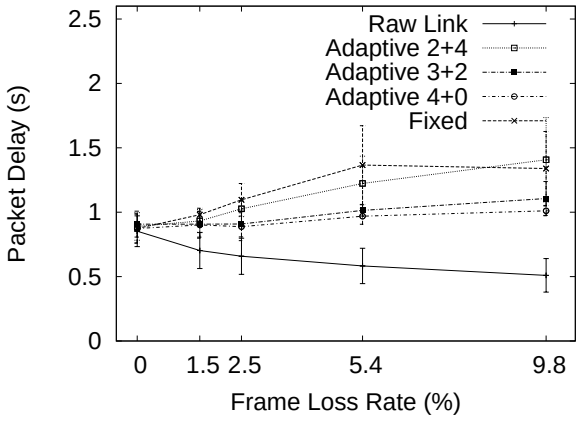

Fig. 7. Media Distribution Delay (Two State)

TABLE VI
DELAY IMPROVEMENT (TWO STATE)

FLR	Fixed	Improvement over Fixed (%)				
(%)	(sec)	2+4	3+2	4+0	4+2	4+4
1.5	0.98	4.9%	7.5%	8.1%	10.1%	8.8%
2.5	1.10	6.4%	17.1%	19.0%	20.3%	19.9%
5.4	1.37	10.4%	25.8%	29.1%	31.5%	32.2%
9.8	1.34	-5.0%	17.4%	24.5%	28.7%	31.3%