A Robust Acknowledgement Scheme for Unreliable Flows

Hoi-Sheung Wilson So, Ye Xia, Jean Walrand
Department of Electrical Engineering and Computer Science
University of California
Berkeley, CA 94720, USA
E-mail: {so,xia,wlr}@eecs.berkeley.edu

Abstract— **The increasing presence of UDP traffic in the Internet and the emergence of sensing applications which do not require full reliability motivates the search for a robust acknowledgement scheme for unreliable flows. TCP-styled cumulative ack relies on the eventual retransmission of *all* lost packets and hence cannot be used for unreliable flows. Our contribution is the design and analysis of a simple, yet robust acknowledgement scheme for unreliable flows. We show that our scheme achieves good performance even when the actual network conditions deviate from the designer's estimate. It works in networks having orders of magnitude difference in bandwidth and loss probability, with little variation of overhead and no change to the algorithms.**

I. Introduction

A. Motivations

An unreliable flow is one that does not require the correct receipt of all transmitted packets. The need for a loss detection algorithm for unreliable flows motivates our search for an acknowledgement scheme that is robust across a wide range of network conditions. Traditionally, loss detection serves as the basis for congestion control and error recovery (also known as reliability control). Unlike TCP flows, unreliable UDP flows have neither congestion control nor error recovery mechanisms. As UDP traffic becomes a significant portion of today's Internet traffic, people see an increasing need for UDP congestion control (e.g., [1],[4]). Furthermore, as the Internet applications become diverse, reliability requirement also becomes varied. Multimedia streaming applications and monitoring applications can benefit from selective retransmissions of some but not all lost packets, due to timing or energy constraint. For example, an MPEG video player that can tolerate some losses may decide that losing less than a certain percentage of P frames in every one-second window is acceptable, but any lost I frames should be retransmitted. (See [7] for similar cases.) As another example, a sensor that monitors the temperature of a room and sends readings to the control center may tolerate a high packet loss probability when the temperature is stable. But when a fire breaks out and the temperature changes rapidly, it may require the sensor readings to be delivered with a high reliability.

Different acknowledgement schemes have been proposed in the past. Most of them are either part of TCP's cumulative ack (CACK) [12] or extensions to it that do not work when the flow is not reliable. In any scheme that preserves cumulative ack, a lost packet prevents the cumulative ack number from increasing. We propose a bit-vector-based ack (BACK) scheme in which each ack packet contains an ack number acknowledging the packet that was just received, together with a vector of bits representing the receiving status of a set of earlier packets. BACK tells the sender both the amount and the identities of the received packets. It uses very few control bits and is robust against both forward losses and ack losses.

BACK can be used by a selective retransmission algorithm or by a window-based congestion control for unreliable flows, which serves as a robust alternative to rate-based controls for these types of traffic, as proposed in [1] and [4]. BACK can also be used by network monitoring algorithms. For instance, packet losses in a heterogeneous network can be caused by different reasons such as congestion in a wired network, congestion or fading in a wireless network, or loss of ack packets in asymmetric networks [6]. An algorithm can use the feedback information provided by BACK to estimate detailed loss statistics of the communication channel to identify the cause of the losses.

In this paper, we do not address how BACK is used in these applications. Instead, we discuss the design choices of BACK and study the performance of a few variants of BACK with respect to the metrics: accuracy and delay.

B. Related Work

An ideal feedback scheme should be able to convey to the sender the amount and the identities of successfully transmitted data. CACK compresses both types of information into one number, the cumulative ack number, i.e., 1 plus the highest byte number below which all data have been received. As a result, CACK alone is not always efficient. As pointed out in [3], TCP Reno and New Reno can only transmit at most 1 dropped packet per round-trip time while TCP Tahoe risk retransmitting packets that might have already been received. In addition, ack losses also make the estimation of the amount of outstanding packets inaccurate.

During the error recovery phase, a TCP sender may use the selective ack (SACK) option [10] to communicate the identities of several contiguous blocks of successfully received data. Its goal is to prevent unnecessary retransmissions at the sender and allow multiple retransmissions in a round-trip time. SACK can be a viable feedback scheme for unreliable flows. Compared with BACK, The main drawback of SACK is that it uses more control bits per ack packet. When the packet sequence number is 4 bytes, each SACK block requires 8 bytes to denote its boundary. As we will see later, SACK often needs to ack more than one block to achieve comparable accuracy as BACK. The precise number of blocks needed depends on the loss statistics in complicated ways.

FACK [11] uses the SACK option to improve the estimation of the amount of outstanding data. It is an algorithmic modification to TCP rather than a new feedback mechanism. TACK [14] tries to accomplish the same goal by adding a control field to the ack packet that denotes the number of additional packets received beyond the cumulative ack. TACK also uses a negative form of SACK for error recovery: each ack packet can nack one block of lost packets. SMART [5] is a proposal similar to SACK where each ack packet contains the identity of the data packet that triggers the ack packet in addition to the cumulative ack. Based on this additional information, the sender is able to infer which packets are lost. Similar to BACK, the feedback scheme proposed in [2] and [8] also use a bit map to describe correctly received or missing data in the receiver buffer. It resembles BACK with a *consecutive placement* of the ack vector. For selective retransmission of multimedia data, it is sometimes desirable to let the receiver, rather than the sender, to discover losses and specify which packets are to be retransmitted [13].

C. Problem Definition

The following is a model of the problem:

1) Sender sends an infinite number of packets to the receiver over a lossy *forward path* at a constant rate of 1 packet per second.
2) Each packet is identified by a monotonically increasing sequence number. The packet carrying sequence number i is denoted D_i.
3) Upon receiving each data packet D_i, the receiver returns an ack packet A_i that is sent over a lossy *reverse path.*
4) Each ack A_i is composed of:
    a) an ack number i
    b) an n-bit ack vector which indicates the arrival status of D_i plus $n-1$ earlier packets. The j-th bit of the ack vector $(0 \leq j \leq n-1)$ is 1 if D_{i-x_j} has been received and 0 if D_{i-x_j} has not been received. (Note: The 0-th bit is therefore always 1 and need not be present in the actual packet sent.) The x_j's define the exact ack scheme and are fixed before the flow begins. We further restrict that $x_0 = 0$ and $x_{n-1} = m-1$ where m is known as the *ack vector spread.* The $(n-1)$-st bit of an ack vector represents the receiving status of the oldest data packet being acknowledged for the last time.
5) The forward and reverse paths are both binary erasure channels with zero delay and no reordering. Their dropping behaviors are controlled by the stationary binary random sequences S and R respectively. If $S(i) = 1$, then D_i is received. If $S(i) = 0$, then D_i is lost. Similarly, the ack packet generated in response to i-th data packet is received if and only if $R(i) = 1$.
6) When a packet D_i is sent, the sender sets a timer of m seconds.[1] The sender may hear about the receiving status of D_i in zero or more of the ack packets $A_i, A_{i+x_1}, A_{i+x_2}, \ldots, A_{i+x_{n-1}}$. Subsequently, if and when the first relevant ack packet for D_i (i.e. the first of $A_i, A_{i+x_1}, A_{i+x_2}, \ldots, A_{i+x_{n-1}}$) is received, the timer is canceled. D_i is said to be acked if the k-th bit of the ack vector in A_{i+x_k} is '1'; a '0' indicates that D_i has been nacked.[2] If the sender does not hear any ack or nack before a timeout, the packet is inferred to be lost. We use $\hat{S}_i$ to denote the final estimate of $S(i)$. At time $t \geq i+m$, $\hat{S}_i$ is 1 if D_i has been acked and 0 otherwise. We define *mistake* as the event $\hat{S}_i \neq S(i)$.

We define the metrics as follows:

- *Accuracy* α is defined to be $Pr[\hat{S}_t = S(t)]$ for any time t. Intuitively, accuracy measures the likelihood that the sender's estimate of whether any packet has been received is correct. In our model, a lost data packet is always correctly detected either by a nack bit or a timeout. Hence

$$\alpha = 1 - Pr[\text{a packet is received but never acked}]$$

  We also define *mistake probability* as $1-\alpha$.
- *Ack Delay* for packet i is defined to be the length of the period after the packet is sent $(t = i)$ and just before an ack for it is first received. A corresponding *Nack Delay* is defined for each lost packet.

[1]For simplicity reasons, we define it such that one timer is set per packet sent. To reduce overhead in an actual implementation, a single periodic timer may be used to timeout all expired packets.

[2]Because we assume packets are not delayed or re-ordered during their transmission, a nack indicates a packet loss. In practice, we may have to hold the timer longer and wait for a more ack packets before deciding a packet is lost.

II. Optimal Ack Vector Placement for Accuracy

In this section, we analyze the optimal ack vector placement which maximizes accuracy under different channel models. More precisely, given an ack vector length n, an ack vector spread m, and the specifications of the random sequences S and R which controls the forward and reverse channels, find an ack scheme $x_0, x_1, \ldots x_{n-1}$ that maximizes the accuracy, with the constraint that $x_0 = 0$ and $x_{n-1} = m-1$.

A. Ack Vector Variants

Finding an optimal ack vector placement is in general difficult. Instead, one hopes to find a good placement for a wide range of channel models that can occur in practice. We will focus on three types of ack vector placement schemes: consecutive placement (CP), uniform placement (UP), and exponential placement (EP). CP is completely specified by the parameter n: $x_0 = 0, x_1 = 1, \ldots, x_{n-1} = n-1$. UP is specified by 2 parameters, m and n: $x_0 = 0, x_1 = (m-1)/(n-1), x_2 = 2(m-1)/(n-1), \ldots, x_{n-1} = m-1$ (Note: For analysis, we assume all x_i's are integers. In practice, we round them to nearby integers.) EP is specified by 2 parameters, m and n, as well: $x_0 = a^0 - 1, x_1 = a^1 - 1, x_2 = a^2 - 1, \ldots, x_{n-1} = a^{n-1} - 1 = m-1$, where $a = (m-1)^{\frac{1}{n-1}}$. CP is chosen because of its simplicity. UP is chosen because it spreads each ack bit as far away from each other as possible and therefore often achieves a high accuracy under a variety of network conditions. EP is a robust scheme that achieves both excellent accuracy and good average ack delay for a wide range of loss characteristics.

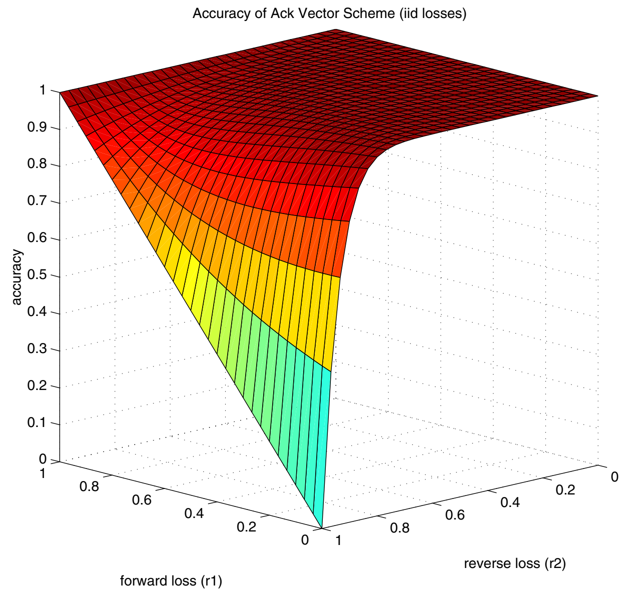


Fig. 1. Accuracy of any ack vector scheme under i.i.d. data and ack losses

B. Independent Random Losses

When both the forward and reverse channels drop packets randomly and independently of each other, any ack vector scheme achieves the same accuracy. Assuming the forward channel has a loss probability of r_1 and the reverse loss probability is r_2, there can be up to n ack packets acknowledging every data packet. Each opportunity for an ack may be missed–either due to a data packet loss in the forward path or an ack packet loss in the reverse path–with probability $r_1 + (1-r_1)r_2$. A mistake occurs when the first data packet is received, followed by the loss of the first ack and the missing of all subsequent $n-1$ acks, the probability of a mistake is therefore $(1-r_1)(r_2)(r_1+(1-r_1)r_2)^{n-1}$. Hence, accuracy is $1-(1-r_1)r_2(r_1+(1-r_1)r_2)^{n-1}$. Fig. 1 shows that the accuracy as a function of the forward and reverse loss probability. Accuracy is the lowest when the forward loss probability is low, but the reverse loss probability is high.

Although the i.i.d. loss model may not be realistic for today's networks, it provides us with an upper-bound of accuracy for any ack vector scheme under any loss model that satisfies the condition, $Pr[\bar{A}_i|\bar{A}_{i_1}, \bar{A}_{i_2}, ..., \bar{A}_{i_k}] \geq Pr[\bar{A}_i]$, for any non-negative integer k and any set $\{i_1, i_2, ..., i_k\}$, where $i_j < i$ for all $j \in \{1, 2, ..., k\}$. Here, we overload the notation A_j to denote the event that ack packet j is received. $\bar{A}_j$ denotes otherwise. If the design of an ack vector is effective in de-correlating bursty ack losses by spacing out the acks for the same data packet, the losses of different A_i's will be close to being i.i.d. even though the underlying channel may not exhibit i.i.d losses.

C. Optimality Conditions of Uniform Placement

Uniform Placement(UP) is optimal for a family of channel models. Given both the forward and reverse channel models, accuracy can be expressed in terms of a function f of the distances between different ack bits: $\alpha = 1 - f(y_1, y_2, \ldots, y_{n-1})$ where $y_j = x_j - x_{j-1}$. It is known that if f is convex and symmetric with respect to permutations of the y_j's, then f is minimized when $y_1 = y_2 = \cdots = y_{n-1} = \frac{m-1}{n-1}$. Therefore, the optimal solution is $x_i = \frac{i(m-1)}{n-1}$. For a brief discussion on this, please refer to Appendix A.

D. One-way Markov On-Off Losses

In this model, the forward channel is assumed to be perfect, while the reverse channel is controlled by a discrete-time 2-state Markov process. In the "GOOD" state (G), the ack packet is received without error. In the "BAD" state (B), the ack is lost. The number of time steps the system spends in either state is therefore geometrically distributed. We show that UP is an optimal placement that maximizes the accuracy α.

$$\begin{aligned}
\alpha &= 1 - Pr[D_i \wedge \bar{A}_i \wedge \bar{A}_{i+x_1} \wedge \cdots \wedge \bar{A}_{i+x_{n-1}}] \\
&= 1 - Pr[D_i]\, Pr[\bar{A}_i]\, Pr[\bar{A}_{i+x_1}|\bar{A}_i] \times \\
&\quad Pr[\bar{A}_{i+x_2}|\bar{A}_i \wedge \bar{A}_{i+x_1}] \times \cdots \times \\
&\quad\quad Pr[\bar{A}_{i+x_{n-1}}|\bar{A}_i \wedge \cdots \wedge \bar{A}_{i+x_{n-2}}] \\
&= 1 - Pr[D_i]\, Pr[\bar{A}_i]\, Pr[\bar{A}_{i+x_1}|\bar{A}_i] \times \\
&\quad Pr[\bar{A}_{i+x_2}|\bar{A}_{i+x_1}] \times \cdots \times Pr[\bar{A}_{i+x_{n-1}}|\bar{A}_{i+x_{n-2}}] \\
&= 1 - Pr[D_i]\, Pr[\bar{A}_i]\, P_{BB}(x_1) P_{BB}(x_2 - x_1) \cdots \\
&\quad P_{BB}(x_{n-1} - x_{n-2}) \\
&= 1 - Pr[D_i]\, Pr[\bar{A}_i] \prod_{j=1}^{n-1} P_{BB}(y_j)
\end{aligned}$$

$P_{BB}(y)$ is defined as the probability that the system starts in state B and returns to state B after y time steps. Assuming the state transition probability from G to B is a, and from B to G is b, then, it can be shown that

$$P_{BB}(y) = \frac{a + b(1-a-b)^y}{a+b}$$

If $a+b = 1$, then $P_{BB}(y) = \frac{a}{a+b}$. If $a+b < 1$, then $P_{BB}(y)$ decays geometrically and approaches $\frac{a}{a+b}$. If $a+b > 1$, $P_{BB}(y)$ oscillates around $\frac{a}{a+b}$ with decreasing magnitude. This rapid oscillation of probability indicates the loss of ack A_i is positively correlated with the losses of earlier acks A_{i-2},A_{i-4} ..., but negatively correlated with the losses of acks A_{i-1},A_{i-3}, and so forth. This does not model real networks and hence, for the rest of this paper, we restrict our attention to the case where $a + b \leq 1$.

Let

$$\begin{aligned}
& f(y_1, y_2, \ldots, y_{n-1}) \\
= \quad & Pr[D_i]\, Pr[\bar{A}_i] \prod_{j=1}^{n-1} \frac{a + b(1-a-b)^{y_j}}{a+b}
\end{aligned}$$

Clearly, f is symmetric with respect to permutations of the y_j's. Let us also assume the y_j's are real numbers. By taking its second derivative, it is easy to show that $\log(P_{BB}(y))$ is a a convex function of y. Therefore, the function $g_j(y_1, y_2, ..., y_{n-1}) := \log(P_{BB}(y_j))$ is convex as a function of $(y_1, y_2, ..., y_{n-1})$, for each $j \in \{1, 2, ..., n-1\}$. So is the function $\sum_{j=1}^{n-1} \log(P_{BB}(y_j))$. Next, the function e^x from $\mathbf{R}$ to $\mathbf{R}$ is convex and increasing. When it is composed with another convex function, the resulting function,

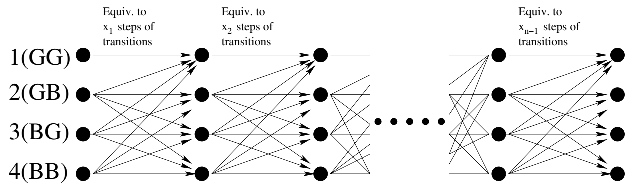

Equiv. to x_1 steps of transitions
Equiv. to x_2 steps of transitions
Equiv. to x_{n-1} steps of transitions
1(GG)
2(GB)
3(BG)
4(BB)

Fig. 2. Evolution of states in the modified system.

$\exp(\sum_{j=1}^{n-1}\log(P_{BB}(y_j)))$ is convex. Therefore, f is a convex function of $(y_1, y_2, ..., y_{n-1})$. It follows that, when $m-1$ is divisible by $n-1$, UP is an optimal placement that maximizes the accuracy α. Note that in UP, given any fixed n, $alpha$ increases monotonically as m increases.

E. Two-way Markov On-Off Losses

In this model, we assume that the forward and the reverse channels are controlled by two independent 2-state Markov chains with states marked as GOOD (G) and BAD (B). When the forward channel is in state G, the packet is received and a corresponding ack is sent. The ack is received only if the reverse channel is also in state G. The transition probabilities of the forward chain are denoted f_{GG}, f_{GB}, f_{BG}, and f_{BB}, with $f_{GG} + f_{GB} = 1$ and $f_{BG} + f_{BB} = 1$. The transition probabilities of the reverse channel are denoted r_{GG}, r_{GB}, r_{BG}, and r_{BB}.

If we view the two independent Markov chains as a single system, there are four possible states: 1(GG), 2(GB), 3(BG), and 4(BB). The first letter indicates the state of the forward chain, while the second indicates that of the reverse. The transition probabilities of the combined system can be written as a matrix $P = [p_{ij}]$ where p_{ij} is simply the product of the appropriate transition probabilities of the forward and reverse channels. Given n, m, f_{GB}, f_{BG}, r_{GB}, r_{BG}, we wish to find $x_1, x_2, \ldots, x_{n-2}$ so that the accuracy is maximized.

1) Calculating Accuracy: Note that $Pr[\bar{A}_{i+x_j}|\bar{A}_i \wedge \bar{A}_{i+x_1} \wedge \cdots \wedge \bar{A}_{i+x_{j-1}}]$ cannot be simplified to $Pr[\bar{A}_{i+x_j}|\bar{A}_{i+x_{j-1}}]$ because knowing that the ack packet $A_{i+x_{j-1}}$ is not received does not pinpoint the state of the system. Non-receipt of the i-th ack packet can be caused by either the loss of i-th data packet or the loss of the ack packet itself.

To simplify the calculation of the accuracy, we change the Markov chain slightly for each set of x_i's such that state 1(GG) becomes absorbing at times $x_1, x_2, \ldots, x_{n-1}$ as shown in Fig. 2. With this change, we can calculate the accuracy by calculating the probability that the system ends up in state 1.

Using the convention that $P^k = \left[p_{ij}^{(k)}\right]$ denotes P (the state transition matrix) raised to the power k, we define $Q^{(k)} = \left[q_{ij}^{(k)}\right]$ as the follows:

$$q_{ij}^{(k)} = \begin{cases} 1 & \text{if } i = 1 \text{ and } j = 1, \\ 0 & \text{if } i = 1 \text{ and } j \neq 1, \\ p_{ij}^k & \text{otherwise.} \end{cases}$$

The distribution of states after m time steps is

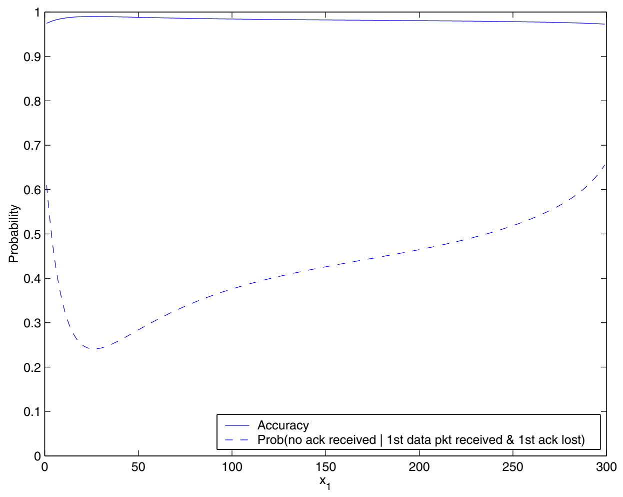


Fig. 3. Accuracy as a function of the location of the middle ack bit (x_1)

$$\mathbf{u}(m) = \mathbf{u}(0) \cdot Q^{(x_1)} \cdot Q^{(x_2 - x_1)} \; \cdots \; Q^{(x_{n-1} - x_{n-2})}.$$

$$\alpha = 1 - Pr[D_i]\, Pr[\bar{A}_i] \cdot \left([0\;1\;0\;0]\, Q^{(x_1)}\, Q^{(x_2 - x_1)} \cdots Q^{(x_{n-1} - x_{n-2})} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right)$$

2) Observations on Optimal Ack Placement: Unlike the simple case in section II-D the optimal ack vector placement in this case is not uniform. To see this, consider a model with $f_{GB} = 0.01$, $f_{BG} = 0.006$, $r_{GB} = 0.01$, and $r_{BG} = 0.08$. We wish to find an optimal ack vector placement using 3 ack bits with $x_0 = 0$ and $x_2 = 300$.

Figure 3 shows the accuracy achieved as a function of the placement of the middle ack bit x_1. Assuming the system starts in state 2 (GB) (i.e. the first data packet is received, but first ack is lost), the broken line shows the conditional probability that the system does not end up in state GG (i.e. no ack ever received). Accuracy is maximized when the middle ack bit is chosen to be 26, not 150, packets away from the first; hence uniform placement is *not* optimal for general two-way on-off loss Models.

III. Evaluation of Different Ack Vectors Placement Schemes

A. Introduction

In this section, we evaluate the accuracy of a few ack vector placement schemes under a variety of network loss models. The design of an ack scheme is influenced by three conflicting goals. The first goal is to achieve a certain required accuracy with as few bits as possible. The second is to allow ack information to be returned to the sender as early as possible. The third goal is to maintain good performance in a wide range of network conditions.

Today's networks often lose packets due to congestion or transmission errors (e.g. wireless losses) that may be bursty. One way to combat bursty ack losses is to separate the different acks for any particular packet as far as possible from each other.

However, this has the undesirable side effect of increasing the ack delay. A good ack vector scheme should strike a balance between a high accuracy and a low ack delay.

The third goal of an ack scheme is robustness. The performance of an ack scheme must be insensitive to the loss pattern of the underlying network. If the accuracy or the average ack delay of an ack scheme is too sensitive to the packet loss probability or the loss pattern, it cannot be used in a heterogeneous network in which the loss behavior is often unknown at design time, and may even vary over time and locations.

Toward these goals, we studied the performance of three different ack schemes using simulation under three network models with a wide range of parameters. The first set of simulations is designed to test how well each ack scheme achieves the first goal by measuring its ability to handle long bursts of errors with different patterns using a limited number of bits. The second set of simulations is designed to evaluate the effectiveness of each ack scheme in achieving a low average ack delay. The third set tests the robustness of each ack scheme with respect to uncertainty in the network bandwidth and loss pattern.

B. Network Loss Models

We simulated the behavior of different ack schemes under three different network models. In the first model, the data packets go through a path which is controlled by a discrete-time 2-state Markov process. In the GOOD state, the data packet is received correctly; in the BAD state, the packet is lost. The ack packets travel over a reverse path governed by another independent discrete-time 2-state Markov process. As a result, the length of loss bursts and successful packet bursts follow a geometric distribution. We call it the Markov On-Off loss model.

In the second model, the forward path and the reverse path are controlled by two independent discrete-time 2-state Markov processes that can either be in the HI state or the LO state. While the system is in the HI state, the packets are dropped independently with probability p_{hi}. When the system is in the LO state, the packets are dropped with probability p_{lo} independently. We called this model the Markov-modulated loss model.

The third model is called the Pareto model. In this model, the forward channel experiences Markov On-Off losses. The reverse channel experiences periods of lossy and lossless periods. The lossy periods follow a Pareto distribution during which all packets are lost. The lossless periods follow a geometric distribution. This model is intended to emulate networks that occasionally experience very long bursts of losses.

C. Accuracy

The first suite of simulation tests the accuracy of various ack schemes in different network loss models. Our simulation will focus on fairly high forward loss rates and heavy reverse losses. The reasons for this choice is two fold. First, when the network experience little or no losses, even a simple positive ack scheme will perform as well as any other BACK or SACK scheme on average. Due to space constraints, we will omit such discussion. Second, an ack scheme that is robust against reverse losses can be exploited to reduce the frequency of ack packets in order to conserve bandwidth or energy. This is extremely useful for two classes of networks: networks with asymmetric links where the relative cost of sending an ack is higher and wireless sensor networks that have stringent power constraints. If every other ack packet is skipped, the effective reverse loss rate is at least 50%.

Unless otherwise stated, the forward channel experiences 20% losses with an average of 200 packets in each loss burst across all distributions. The reverse channel experiences different levels of loss with the same average loss burst length of 1000. The ack vector length varies from 2 to 64 bits for UP and EP, and from 2 to 2048 bits for CP. The ack vector spread (m) is fixed at 65536. (Note: To make the graphs more readable, within each graph, the labels of the curves are listed in a top down order corresponding to the relative positions of the curves in it.)

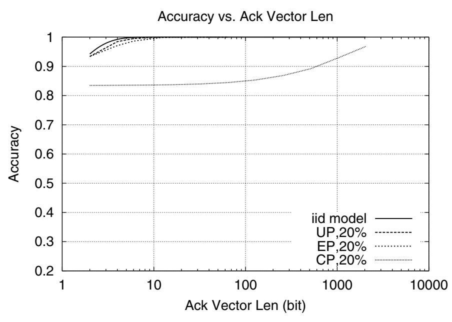
In Fig. 4, the x-axis shows the number of bits in each ack vector, the y-axis shows the accuracy achieved. In addition to the 3 ack schemes being tested, we also include the theoretical accuracy achieved if both the forward and the reverse channels experienced i.i.d. losses instead, as described in section II-B.

Under the Markov On-Off model, UP achieves close to the theoretical maximum regardless of the loss probability. This shows that even though UP may not always achieve the optimum as demonstrated in section II-E.2, it achieves a very high accuracy in practice. As expected, CP does not handle long loss bursts well when the ack vectors are short. To achieve a high accuracy (>90%), the number of bits in each ack vector must be on the order of the number of packets in a loss burst (in this case, 1000). This is in sharp contrast to UP or EP, which requires only tens of bits to achieve the same accuracy even under severe losses. The performance of EP trails behind UP closely across all loss probabilities, especially when the loss probability is below 50%. Overall, EP requires less than twice the number of bits as UP to achieve the same accuracy. The reason why EP does not achieve as high an accuracy as UP is that EP concentrates most of its bits at the beginning which means their losses are likely to be positively correlated. UP achieves a higher accuracy at the expense of higher ack and nack delay which will be quantified later.

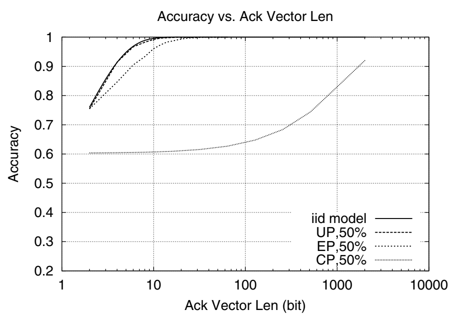
Under the Markov-modulated model, the performance of both UP and EP are very close to the theoretical maximum with UP slightly better than EP as shown in Fig. 5(a). CP also performs very well under this model, requiring roughly twice the number of bits as UP to achieve similar accuracy. The reason for the good performance of CP is that, under this loss model, a fraction of packets can still be successfully transmitted during the high loss periods. Therefore, the probability of a large number of consecutive acks being lost in a row is much smaller than in other models. In the extreme case when the system spends most of its time in the high loss state, and hence the losses are almost i.i.d., CP achieves almost the same accuracy as any other ack vector placement scheme.

Under the heavily tailed Pareto model ($\beta = 1.4$) [3], when the losses are moderate ($\leq$ 20%), both UP and EP are capable of achieving an accuracy very close to 1 using only about 10 bits. However, with heavier losses ($\geq$50%), the number of error

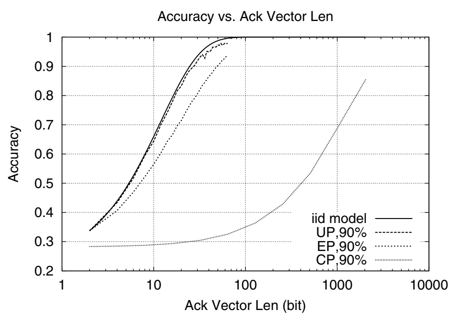
[3]Pareto c.d.f. is $F(x) = 1 - (\frac{K}{x})^{\beta}$ for some $K > 0$ determined by the mean.



(a) 20% reverse loss

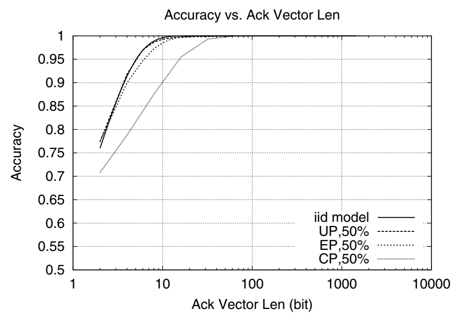


(b) 50% reverse loss

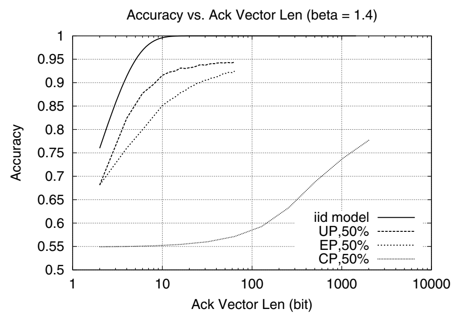


(c) 90% reverse loss

Fig. 4. Effects of number of ack bits on accuracy under Markov On-Off losses

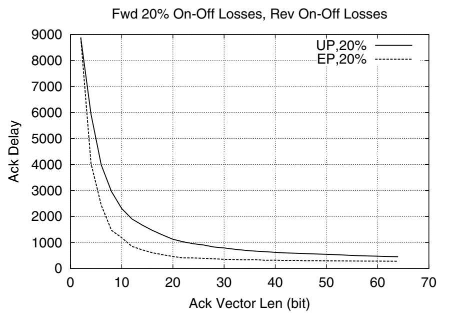


(a) 50% reverse loss, Markov-Modulated

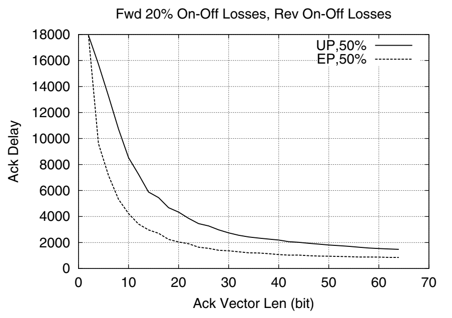


(b) 50% reverse loss, Pareto

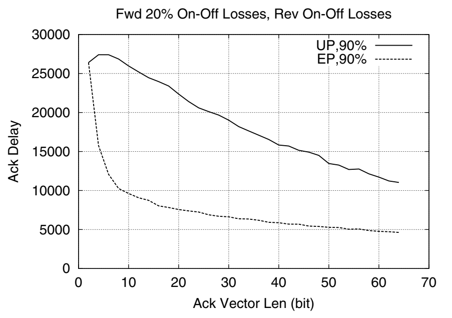
Fig. 5. Effects of number of ack bits on accuracy under Markov-modulated and Pareto losses (Note: the y-axis is scaled differently for readability.)



(a) 20% reverse loss

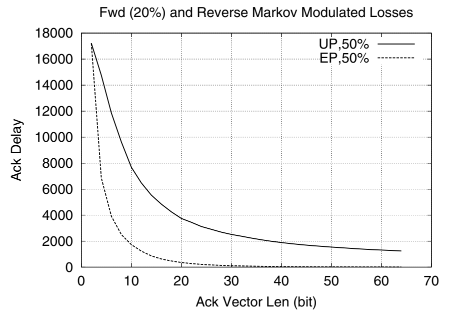


(b) 50% reverse loss

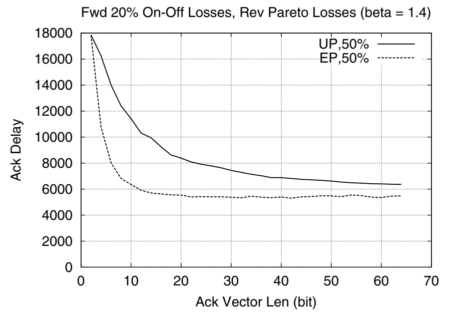


(c) 90% reverse loss

Fig. 6. Effects of number of ack bits on ack delay under Markov on-off losses



(a) 50% reverse loss, Markov-Modulated



(b) 50% reverse loss, Pareto

Fig. 7. Effects of number of ack bits on ack delay under Markov-modulated and Pareto losses

bursts longer than the ack vector spread (i.e. 65536) becomes significant. As a result, the accuracy of both EP and UP do not approach 1 as shown in Fig. 5(b). Even so, under 20% forward and 50% reverse loss, UP requires 10 bits while EP requires 28 bits to achieve an accuracy of 90%, which are both quite reasonable. At even heavier reverse losses (90%), UP requires 24 bits while EP requires 44 bits to achieve an accuracy of 80%. In general, EP requires a small constant factor more bits (roughly twice) than UP to achieve the same accuracy, which is still several orders of magnitude less than what CP requires.

We can conclude that the accuracy of UP and EP are much higher than CP in the different network conditions we tested. In general, to achieve high accuracy, the ack vector length of CP must be about the same size as the average length of error bursts. In contrast, the accuracy of UP is often comparable to the theoretical maximum predicted by the i.i.d. model presented in section II-B. The performance of EP closely trails behind UP. To achieve the same accuracy as UP, EP usually requires no more than twice the number of bits.

D. Ack Delay

The performance of an ack scheme is not only about high accuracy. The time it takes for an ack to be returned to the sender is also very important. In this section, we compare the performance of UP and EP in terms of the ack delay under different network models. In particular, we want to find out how well each ack scheme makes use of ack bits in reducing ack delay in addition to increasing accuracy. The network loss models and the parameters for the simulations used in this section are identical to those in the previous. Again, the forward loss (Markov on-off) is fixed at 20%.

Fig. 6 shows the performance of EP and UP under different levels of reverse loss under the assumptions of Markov on-off losses. The ack delays shown are in units of packet intervals, which is one second. The upper curve shows the average ack delay of UP as a function of the number of ack bits. The lower curve is that of EP. For EP, the average delay drops exponentially with the addition of more bits. For UP, it drops roughly proportional to $1/n$. At extremely high loss probability, the ack delay decreases very slowly for UP, while the ack delay still drops rapidly when more ack bits are used in EP. It shows that EP is more effective than UP in utilizing extra ack bits to reduce ack delay in low and high loss situations alike. The relative performance of EP and UP in terms of ack delay is very consistent regardless of the loss model. The general trend of ack delay for the Markov-modulated and Pareto models can be seen in Fig. 7.

Under the Markov-modulated model, the differences between EP and UP become more pronounced. Using only 10 bits, EP and UP achieve a similar accuracy as shown in Fig. 7(b), but their ack and nack delay differ significantly, as shown in the following table:

	Ack Delay		Nack Delay	
Fwd/Rev Loss	UP	EP	UP	EP
20%/10%	962.5	16.8	10032.1	85.6
20%/20%	1963.1	146.1	11000.8	299.1
20%/50%	7685.5	1734.6	16508.8	2449.7
20%/90%	26226.0	6875.8	32182.7	8566.9

The advantage of EP is that first few ack bits follow each other very closely. Therefore, if the lossy periods have ‘holes’ (i.e. not all packets within a loss burst are corrupted or lost), EP is able to exploit the characteristics of this type of channel to achieve a low ack delay without reducing the accuracy. Another advantage of EP is that it incurs far smaller nack delay than UP in all the network conditions we have tested. The reason is that when a data packet is lost, no corresponding ack is sent. In UP, the first nack is therefore sent $\frac{m-1}{n-1}$ packets after the original data packet, which can be very large when m is large. However, in EP, the first nack is sent almost immediately after the data packet is lost. In short, we expect EP to achieve a similar accuracy as UP but a much lower ack and nack delay.

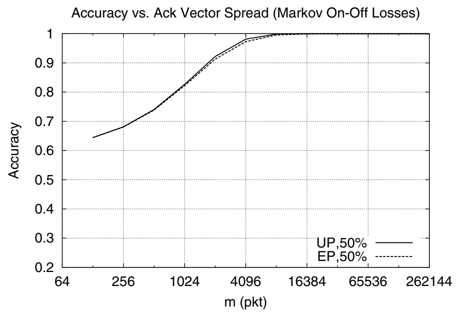
Under the Pareto model, when the ack loss probability is no more than 20%, EP quickly approaches the best possible ack delay with tens of bits. UP also perform reasonably well. At 50% ack loss probability (Fig. 7(b)), EP and UP begin to diverge. The average ack delay for EP decreases very rapidly, but the decrease for UP is slower. Under severe ack losses (90%), the ack delay for EP still decreases exponentially fast as more ack bits are added, but UP does not perform as well. The decrease in ack delay is roughly linear in the number of ack bits.

E. Robustness Effects of Uncertainty in m

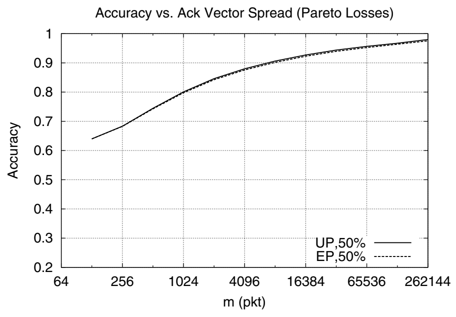
1) Accuracy: So far, we have only considered the case where the ack vector spread m is fixed. In practice, m is a design parameter that needs to be chosen based on the estimated characteristics of the network and the maximum acceptable ack / nack delay. When the average length of the loss burst does not change, m will be roughly proportional to the bandwidth. In order to achieve the best possible performance, it may be necessary to fine-tune the parameter m for each particular network. However, as stated in the design goals, we wish to design an ack scheme that works across diverse network conditions. Therefore, we hope to find an ack scheme whose performance is insensitive to the choice of m and hence we can eliminate the need for manual fine-tuning. In this section, we investigate the effects of varying m on the accuracy and the ack delay.

In the following set of simulations, we use a fixed length ack vector of 64 bits, but the spread (m) varies from $2^7 = 128$ to $2^{18} = 262144$. The forward and reverse channel follows the Markov On-Off, Markov-modulated, or Pareto models. Forward loss probability is always 20% with an average burst loss length of 200 while the reverse loss probability can be 20%, 50%, or 90% with an average burst loss length of 1000. These loss parameters are the same as before.

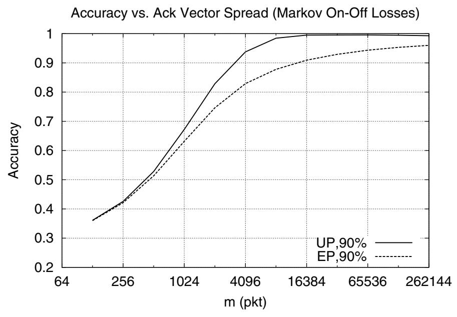
Simulation results show that, except under severe ack losses ($\geq$90%), there is virtually no difference in terms of the accuracy achieved by UP and EP given any ack vector spread m. For example, Fig. 8(a) and 8(b) shows the accuracy achieved by EP and UP under Markov on-off and Pareto loss models are virtually indistinguishable. Under the Markov-modulated model, both EP and UP achieve an accuracy of close to 1 at all loss probabilities. Under severe losses, UP shows somewhat higher accuracy than EP under Markov on-off losses, as shown in Fig. 8(c). Under Markov-modulated and Pareto losses, the difference in accuracy between EP and UP is smaller.



(a) 50% reverse on-off losses

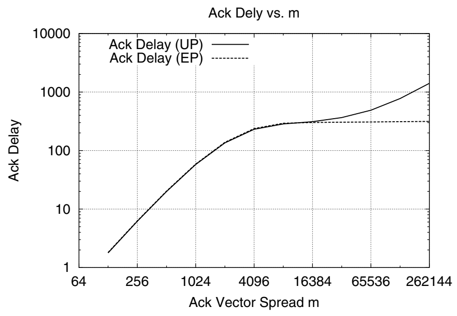


(b) 50% reverse Pareto losses

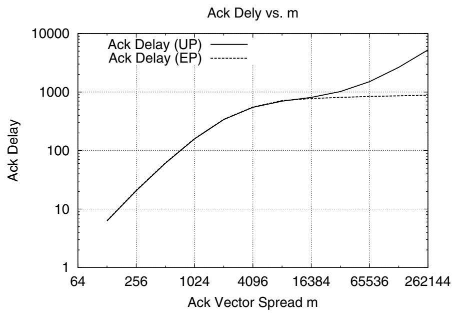


(c) 90% reverse on-off losses

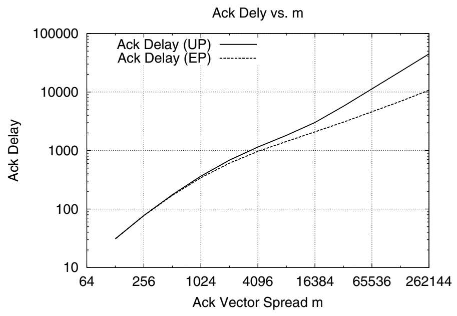
Fig. 8. Effects of ack vector spread on accuracy



(a) 20% reverse loss

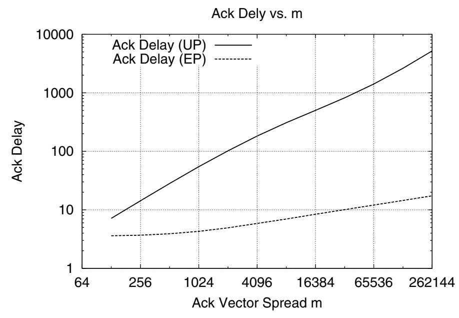


(b) 50% reverse loss

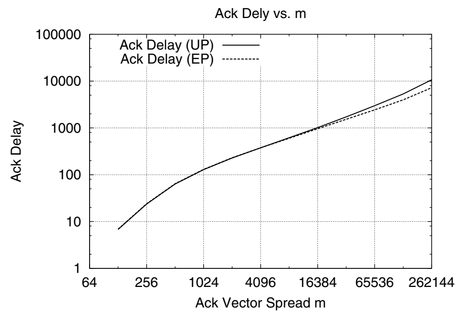


(c) 90% reverse loss

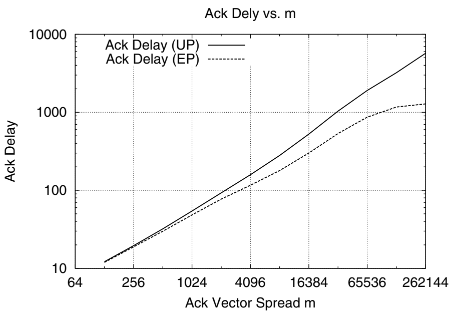
Fig. 9. Effects of ack vector spread on ack delay under Markov on-off losses



(a) 50% reverse loss, Markov-modulated



(b) 50% reverse loss, Pareto, avg. loss bursts of 1000 packets



(c) 50% reverse loss, Pareto, avg. loss bursts of 50 packets

Fig. 10. Effects of ack vector spread on ack delay under Markov-modulated and Pareto losses

One important point to note is that the ack vector spread does have a noticeable effect on the accuracy of both EP and UP when the errors occur in long bursts. As a rule of thumb, m should be much larger than the average length of a loss burst (e.g. 10 times or more), especially when the distribution of the length of the loss bursts has a heavy tail.

2) Ack Delay: We noted in the previous section that choosing a large enough m is important for achieving a high accuracy. In this section, we investigate the effects of varying m on the average ack delay.

Fig. 9 shows the average ack delay incurred by EP and UP under different levels of Markov on-off losses. Note the plots use a log-log scale. When m is smaller than the average burst loss length (i.e. 1000), the behaviors of UP and EP are similar. However, when m is large, the ack delay of UP grows roughly proportional to m. The ack delay of EP, on the contrary, remains fairly insensitive to the choice of m, even when the reverse loss probability is as high as 50%. The reason for this behavior is as follows. Consider an ack vector with length n and spread m. The position of the $(n-1)$-st bit is $x_{n-1} = a^{n-1} - 1 = m - 1$. Therefore, $a = e^{\frac{\ln m}{n-1}}$. The position of the j-th bit is $x_j = a^j - 1 = e^{\frac{j \ln m}{n-1}} - 1$. Now, consider another ack vector with the same number of bits, but the spread is increased to km. The position of the last bit is now at $x'_{n-1} = a'^{n-1} - 1 = km - 1$. Hence, $a' = e^{\frac{\ln km}{n-1}}$. The position of the j-th bit is now $x'_j = (a')^j - 1 = e^{\frac{j \ln (km)}{n-1}} - 1$. The ratio of the new position to the old one is $\frac{x'_j}{x_j} = (e^{\frac{j \ln (km)}{n-1}} - 1)/(e^{\frac{j \ln m}{n-1}} - 1) \approx (e^{\frac{j \ln (km)}{n-1}})/(e^{\frac{j \ln m}{n-1}}) = e^{\frac{j \ln k}{n-1}} = k^{\frac{j}{n-1}}$. Therefore, for the first few bits where j is much smaller than $n-1$, the new position does not change very much. Under moderate losses, most packets are acked within the first few attempts, and hence an increase in m results in very little increase in the average ack delay.

Under extreme losses, the ack delay continues to increase linearly for EP as we increase the ack vector spread. This increase in delay is expected because EP has not reached its maximum accuracy given the limited spread under such heavy losses. Even under such extreme conditions, the average delay incurred by EP increases at a much slower rate compared to UP. As a result, the delay incurred by EP is much lower when compared to UP.

Under the Markov-modulated loss model and for low (10%) to medium (20%) losses, EP achieves the best possible delay regardless of the choice for m. At higher losses (50%), the ack delay incurred by EP starts to increase as m increases, but it increases at a much slower rate than UP as shown in Fig. 10(a). When m reaches 262144 (i.e. 256 times the average loss burst length), the ack delay is still only about 20. Under extreme losses, the ack delay for both EP and UP goes up as m increases, though the delay for EP rises at a much slower rate.

Finally, we note that under a heavy-tailed loss model (Fig. 10(b)), the ack delay increases roughly linearly when m is much larger than the actual burst length. This is expected because under heavy-tailed losses, not only are the first few ack bits of an ack vector used, the entire ack vector may be used. When the last few bits of an ack vector are used, the delay is very sensitive to the choice of m. Unfortunately, this is unavoidable because the occasional extremely long bursts lead to long ack delays.

It is important to note that the behavior of EP is *not* fundamentally different under different loss models as Fig. 9(b),10(a), and 10(b) might suggest otherwise. In fact, regardless of the loss model, when the ack vector spread is too small to achieve an accuracy close to 1, the ack delay of both EP and UP grows at roughly the same rate with respect to the increase in m. Once the maximum accuracy is reached, any further increase in m results in a roughly linear increase in the ack delay for UP, but very little increase in the case of EP. The difference among various loss models is in the m required to achieve the maximum accuracy. Under the Markov-modulated model, the maximum accuracy is achieved when m is very small and hence Fig.10(a) resembles the right portion of Fig. 9(b) which is mostly flat. In contrast, under the heavy-tailed Pareto loss model, m must be larger than 26144 in order to reach the maximum accuracy. Hence 10(b) is similar to the left portion of Fig. 9(b). If the average burst size of the Pareto model is reduced from 1000 to 50, the maximum accuracy can now be achieved with a smaller m. Fig. 10(c) shows that the ack delay of EP flattens as soon as the maximum accuracy is reached. It shows that the general trend of the delay incurred by EP is not so much affected by the loss pattern (Markov on-off, Markov-modulated, or Pareto).

F. Comparison with SACK

In this section, we compare the SACK to BACK with exponential placement. The SACK used here is not the same as TCP-SACK which cannot be used directly for unreliable flows because it uses a cumulative ACK number. The SACK being compared here acknowledges the most recent received packet in addition to a few blocks of most recently received packets.

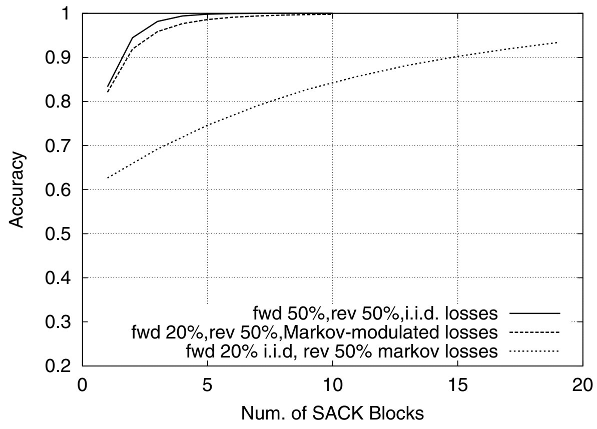


Fig. 11. Accuracy of SACK under different loss models

When the loss bursts are long and consecutive, SACK can efficiently encode the ack information in a few ranges of packets.

The ack delay incurred is also relatively low. However, the number of SACK blocks required to achieve a certain accuracy is very sensitive to the packet loss pattern. When the forward channel losses are fragmented or when the reverse channel losses are bursty, more SACK blocks are required. Fig. 11 shows the accuracy of SACK as a function of the number of SACK blocks under three loss models. The top curve shows the accuracy of SACK under 50% i.i.d. losses, both forward and reverse. The middle curve shows the accuracy under Markov-modulated losses. The forward loss is at 20% with an average burst length of 200; the reverse loss is at 50% with an average burst length of 1000. The bottom curve shows the accuracy under 20% forward i.i.d. losses, and 50% reverse Markov on-off losses with a mean burst loss length of 50 packets. For comparison, BACK with EP ($m = 65536$) uses no more than 13 bits to achieve an accuracy of 0.99 in all three situations. SACK requires 4, 6, and 39 blocks respectively. In the third situation, if the average burst length of the reverse channel is longer, even more blocks are required. The per packet overhead of BACK is $s + n$, where s is the size of the sequence number (in bits) and n is the ack vector length. SACK requires $2sb$ where b is the number of SACK blocks. (For example, TCP uses 32 bits for each sequence number, so s would be 32.)

SACK has an advantage over BACK in terms of ack delay. For example, in the third situation, SACK achieves an average delay of 28.7 using 39 blocks. Using a fixed m of 65536, BACK-EP achieves a delay of 426.9, 83.9, 43.5, and 35.6 using 16, 32,64,128 bits respectively. It shows that while SACK achieves better delay than BACK-EP when their accuracy is comparable, BACK-EP achieves a delay within a factor of 2 of SACK using far fewer bits even when the spread m is orders of magnitude larger than the actual average loss burst length of 50.

IV. Conclusion

Current loss detection methods either do not work correctly for unreliable flows or do not adapt well to high-speed networks with highly fragmented losses. Granted, when both the forward and reverse channels are nearly perfect, any ack scheme such as a simple positive ack scheme should work well. However, a feedback scheme should not suddenly break down when the losses probability is very high or when the losses are very bursty. In general, the performance of an ack scheme should degrade gracefully as the operating conditions of the network deviate from the original conditions anticipated by the protocol designer. In particular, our study shows that an ack vector with Exponential Placement (EP) has several advantages: it is simple, robust, scalable, and incurs low delay.

An EP ack vector is simple because it is fully specified using only two parameters: m and n. To even further simplify its implementation, we can specify m to be a power of 2 plus 1 which further reduces the number of parameters to one. In addition, the ack bit positions are all powers of 2 which can be easily calculated in computers by shifting bits, thereby avoiding any complicated and slow floating-point arithmetic. EP is also robust because it achieves a high accuracy even under severe losses. When the channel loss probability is low, it incurs very low ack and nack delay.

As seen in section III-E.1, to achieve a high accuracy the ack vector spread has to be much larger than the actual burst loss length. Unfortunately, the properties of a network such as the loss rate and the average length of a loss burst are often unknown to the protocol designer. These properties often depend on the workload and they may change over time. Consequently, a designer has to choose a value for m based on the estimated burst length and the maximum tolerable ack delay. Choosing a large m may incur unnecessary delay because acks are spread too far apart; picking a small m is risky if the losses turn out to be more severe than expected. The major benefit of EP is that the ack delay incurred is *not* sensitive to the choice of m once the maximum accuracy is reached. Therefore, we recommend the use of EP with m chosen to be at least 10 times larger than the expected burst loss size. This configuration achieves both a high accuracy and a low ack delay.

EP is also scalable in that it can accommodate a network with very long bursts of errors using a few bytes worth of ack bits. For example, if the link speed is B kbps, an average packet is S bits long, and a lossy period is expected to last no more than T seconds, then m should be at least $10BT/S$. For a 10 Gbps optical link with an average packet size on the order of 10Kb, and an lossy period of 100ms, m should be at least 10^6 which corresponds to using about 20 ack bits per packet (assuming that m is chosen to be a power to 2 plus 1.) In a sensor network with radio links of about 10Kbps, an average packet size on the order of 200 bits, and a lossy period of up to 10s, m should ideally be at least 5000 which corresponds to about 13 bits.[4] In either case, the number of per packet overhead changes by only a few bits even though the networks have orders of magnitude difference in throughput and delay.

Appendix A: Relevant Majorization Results

This brief discussion is derived from [9]. For any $x = (x_1, ..., x_n) \in \mathbf{R}^n$, let $x_{[1]} \geq ... \geq x_{[n]}$.

[4]In reality, a sensor node might not have enough memory to remember so many packets. In such case, m should be reduced accordingly.

Definition 1: For $x, y \in \mathbf{R}^n$, x is said to be majorized by y, denoted by $x \prec y$, if the following two conditions both hold.

$$\begin{aligned} \sum_{i=1}^{k} x_{[i]} &\leq \sum_{i=1}^{k} y_{[i]}, \qquad k = 1, ..., n-1 \\ \sum_{i=1}^{n} x_{[i]} &= \sum_{i=1}^{n} y_{[i]} \end{aligned}$$

It is a fact that for all $(y_1, ..., y_n) \in \mathbf{R}^n$,

$$(\bar{y}, ..., \bar{y}) \prec (y_1, ..., y_n)$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

Theorem 1: If $\phi : \mathbf{R}^n \to \mathbf{R}$ is symmetric and convex, then $x \prec y$ implies $\phi(x) \leq \phi(y)$.

By the above fact and theorem, UP is optimal whenever the mistake probability is symmetric and convex.

References

[1] Deepak Bansal and Hari Balakrishnan. Binomial Congestion Control Algorithms. In *Proc. INFOCOM 2001*, Anchorage, AK, April 2001.
[2] B.T. Doshi, P.K. Johri, A.N. Netravali, and K.K. Sabnani. Error and flow control performance of a high speed protocol. *IEEE Transactions on Communications*, 41:707–720, May 1993.
[3] Kevin Fall and Sally Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *Computer Communication Review*, 26(3):5–21, 1996.
[4] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer. Equation-Based Congestion Control for Unicast Applications. In *ACM Proceeding Sigcomm 2000*, Stockholm, Sweden, August 2000.
[5] S. Keshav and S.P. Morgan. SMART: Retransmission: Performance with Random Losses and Overload. In *Proceedings of the IEEE Infocom 1997*, Kobe, Japan, April 1997.
[6] T.V. Lakshman, U. Madhow, and B. Suter. Window-Based Error Recovery and Flow Control with a Slow Acknowledgement Channel: A Study of TCP/IP Performance. In *Proceedings of the IEEE Infocom 1997*, Kobe, Japan, April 1997.
[7] J. R. Li, S. Ha, and V. Bharghavan. A Transport Protocol For Heterogeneous Packet Flows. In *Proceedings of the IEEE Infocom 1999*, New York, NY, March 1999.
[8] John C.-H. Lin and Sanjoy Paul. RMTP: A reliable multicast transport protocol. In *Proceedings of the IEEE Infocom 1996*, pages 1414–1424, San Francisco, California, March 1996.
[9] Albert W. Marshall and Ingram Olkin. *Inequalities: Theory of Majorization and Its Applications*. Academic Press, 1979.
[10] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. *TCP Selective Acknowledgement Options, RFC 2018*. IETF, October 1996.
[11] Matthew Mathis and Jamshid Mahdavi. Forward Acknowledgement: Refining TCP Congestion Control. In *Proc. ACM SIGCOMM '96*, Palo Alto, CA, August 1996.
[12] W. Richard Stevens. *TCP/IP Illustrated, Vol. 1: The Protocols*. Addison-Wesley, 1994.
[13] Hari Balakrishnan Suchitra Raman and Murari Srinivasan. An Image Transport Protocol for the Internet. In *Proc. International Conference on Network Protocols (ICNP) 2000*, Osaka, Japan, November 2000.
[14] J. Waldby, U. Madhow, and T. V. Lakshman. Total Acknowledgements: A Robust Feedback Mechanism for End-to-End Congestion Control. In *Proceedings of ACM Sigmetrics'98*, Madison , Wisconsin, June 1998.