A Model, Analysis, and Protocol Framework for Soft State-based Communication*

Suchitra Raman   Steven McCanne
Department of EECS, University of California, Berkeley
{suchi,mccanne}@eecs.berkeley.edu

Abstract

"Soft state" is an often cited yet vague concept in network protocol design in which two or more network entities intercommunicate in a loosely coupled, often anonymous fashion. Researchers often define this concept operationally (if at all) rather than analytically: a source of *soft state* transmits periodic "refresh messages" over a (lossy) communication channel to one or more receivers that maintain a copy of that state, which in turn "expires" if the periodic updates cease. Though a number of crucial Internet protocol building blocks are rooted in soft state-based designs — e.g., RSVP refresh messages, PIM membership updates, various routing protocol updates, RTCP control messages, directory services like SAP, and so forth — controversy is building as to whether the performance overhead of soft state refresh messages justify their qualitative benefit of enhanced system "robustness". We believe that this controversy has risen not from fundamental performance tradeoffs but rather from our lack of a comprehensive understanding of soft state. To better understand these tradeoffs, we propose herein a formal model for soft state communication based on a probabilistic delivery model with relaxed reliability. Using this model, we conduct queueing analysis and simulation to characterize the data consistency and performance tradeoffs under a range of workloads and network loss rates. We then extend our model with feedback and show, through simulation, that adding feedback dramatically improves data consistency (by up to 55%) without increasing network resource consumption. Our model not only provides a foundation for understanding soft state, but also induces a new fundamental transport protocol based on probabilistic delivery. Toward this end, we sketch our design of the "Soft State Transport Protocol" (SSTP), which enjoys the robustness of soft state while retaining the performance benefit of hard state protocols like TCP through its judicious use of feedback.

*This research was supported by DARPA contract N66001-96-C-8508, by the State of California under the MICRO program, and by NSF Contract CDA 94-01156. The views expressed here do not reflect the position or policy of the U.S. government.

1 Introduction

"Soft state" is an often cited yet vague concept in network protocol design. Even the author of this term defined it by example rather than from first principles. In his landmark retrospective paper on the Internet architecture, Clark sketched a then unconventional framework for signaling type-of-service state into the network, in which end-systems periodically transmit messages into the network to maintain "flow state" in the routers [12]. He called this concept *soft state*, and despite this induction into the network research community's vocabulary a decade ago, a more precise definition or model for soft state has not been forthcoming.

Researchers often refer to the concept of soft state implicitly, claiming a design is based on soft state because it carries certain operational properties. For example, state is called "soft" if it is maintained by some agent in the network that expires that state after a certain time interval unless it is refreshed by some update message received across the network from another agent in the system. That is, a source of *soft state* transmits periodic "refresh messages" over a (lossy) communication channel to one or more receivers that maintain a copy of that state. Associated with this state is a pending timer, which is reset upon receipt of each refresh message. If the timer expires (because the refresh messages cease), the state is deleted, and some subsequent protocol action is typically invoked. This operational framework, which is now used in a large number of Internet protocols (RSVP [53], PIM [15], RTP [45], SRM [20], SAP [21], and so forth), has been coined an *announce/listen* protocol [44] since a sender actively *announces* soft state updates and one or more receivers passively monitor and *listen* to those updates.

The reason this design principle has been so often cited and utilized is that it works well in practice. Systems built on soft state are robust. In harmony with one of the hallmark properties of the Internet, soft state systems afford "survivability in the face of failure" [12]. In Clark's "flow state" example, if a router crashes and the underlying path is recomputed, the flow state is automatically established along the new network path since periodic refresh messages from the end-system immediately begin to follow the new route.

In contrast, a hard state approach to flow state establishment would involve a specific setup and teardown protocol, e.g., Q.931 [49], ST-II [16], or Tenet [4]. A benefit to this approach is that the state is established just once with a reliable delivery protocol like TCP, thereby avoiding the bandwidth or processing overhead of the soft state refresh messages. However, when failure occurs in this environment and, say, routes are recomputed, the system would have to simultaneously detect the failure, explicitly tear down the old state in each router, and re-establish the state along the new path. This is an exceptional condition that must be explicitly engineered and leads to complex interactions among many different distributed components.

While this style of hard state design is by no means an infeasible approach (and quite arguably the best approach in certain environments), it can confound incremental engineering and deployment because every sub-component has to be foolproof before the system can be made to work as a whole. This is why, in particular, the approach does not work all that well in the Internet, where highly heterogeneous components have very mixed levels of reliability. With the soft state model, on the other hand, explicit error recovery is unnecessary as it is implied by — or "built into" — the design itself.

Another qualitative benefit of the soft state design methodology is that, once adopted, it calibrates the designer's expectations for an environment like the Internet, where consistent, homogeneous, and high-performance communication is often the exception rather than the rule. With a hard state abstraction based, say, on TCP, once a connection is successfully created, data is transferred, and the connection torn down, the application knows with complete assuredness that the other end point has successfully received its state. Thus, a designer using this primitive would be tempted to build an entire system based on a high degree of tight consistency. Then, only after the design is complete, would exceptional conditions be explored, e.g., what happens when the other side crashes mid-way through operation, or what happens if the other side is unreachable, and so forth. The outcome of this methodology is predictable: a large fraction of the resulting system becomes ingrained in error detection and recovery.

In a soft state framework, the designer is forced to presume inconsistency from the start, then think through how richer abstractions can be built on this inconsistent model. Here, consistency arises perhaps only slowly, by virtue of the periodic announce/listen update process. As a consequence, the resulting design necessarily accommodates component failure and inconsistency intrinsically and thus can continue to operate in the face of adversity. For example, a multimedia conferencing system that relies upon a centralized controller to track group membership or perform multiplexing would fail catastrophically if that controller goes down [50]. In contrast, the light-weight sessions conferencing model [28], which is based on a loosely-coupled, distributed group membership disseminated through announce/listen, gracefully accommodates end-system failures and network partitions. When a network partition occurs, for instance, the partitioned sub-sessions continue to operate and group membership knowledge that had spanned the partition eventually times out. Once the failure dissipates, the membership announcements resume their reach across the entire session and the group state quickly converges to accurately track the reformed session, all a consequence of normal protocol operation. In short, such designs are *vertically robust* in that they must accommodate inconsistency across distributed components throughout their design and as a consequence are robust against network pathologies so common to the Internet [38].

Given the attractive properties of soft state and the proliferation of the announce/listen primitive in so many Internet protocols over the past decade, one would expect that a great deal of research would exist that not only clearly articulates what soft state is but characterizes the fundamental performance tradeoffs of soft state designs. Yet such work is scant. Not only is there a dearth of analysis and refinement of soft state, but there in no well-defined communication framework, no common protocol architecture, and no application API that is based on this the soft state model. We believe this is a great misfortune because such work could help guide protocol designers and engineers to decide when and where the performance tradeoffs of soft state are worth the benefit and a common implementation and framework would provide reusable protocol building blocks for application designers. In this paper, we address this void with a formal model for soft state communication based on a probabilistic delivery model with relaxed reliability. Our contributions are as follows:

- We present a novel model for soft state protocols and probabilistically define an associated *consistency metric.*
- We theoretically analyze our model for the simple open-loop announce/listen protocol.
- We systematically characterize data consistency and performance tradeoffs of our soft state model under a range of workloads and network loss rates for the simple open-loop case and its variants,
- We extend the open-loop variant of announce/listen by adding receiver feedback to enhance data consistency and performance without increasing network resource consumption.
- Based on our model, as well as the observation that several protocols have inherently "soft" or periodically changing data, e.g., route advertisements [36, 25, 33], DNS updates [37], MBone session directories [24], stock quote or general information dissemination services [39], we propose a soft state-based transport protocol (SSTP) framework. The SSTP framework provides a parameterized spectrum of reliability semantics all derived from one framework — from simple announce/listen communication to feedback-based reliable transport. SSTP also optimally allocates bandwidth based on packet loss rates and application workload to maximize consistency. The result is a parameterized framework that can be tuned to provide one of a continuum of "reliability levels". We also incorporate ideas from application-level framing (ALF) [13] to provide an interface that allows it to be tailored to the specific application.

The rest of this paper is organized as follows. In Section 2, we present a data and communication model for soft state. We then analyze the simple open-loop announce/listen case in Section 3. Based on our analysis, we propose two techniques to improve the performance of the traditional announce/listen framework: (1) a simple, two-level differentiated transmission scheme, described in Section 4, and (2) a novel application of feedback to guide the link scheduling decisions at the source, described in Section 5. Section 6 develops SSTP, a practical protocol framework for realizing and reusing the soft state communication primitive across multiple applications. In Section 7, we review past and ongoing work related to this paper and in Section 8, we present our concluding remarks.

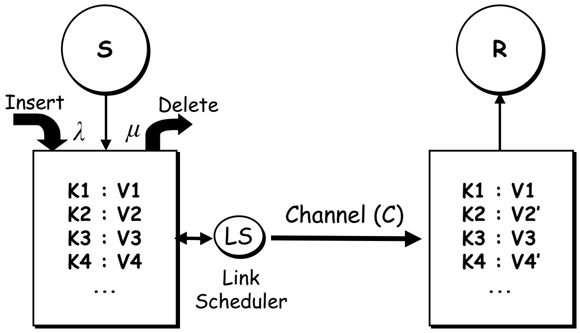

Figure 1: Soft state data model comprises an evolving table of $\{key, value\}$ pairs.

2 The Data Model

In order to evaluate the tradeoffs and performance of soft state communication, we must first carefully define a framework and model that firmly grounds our analysis and discussion.

Our model for "soft" data is a table of $\{key, value\}$ pairs at the sender, or publisher. The publisher may add, delete, or update a record at any given time. The scheduler at the source periodically announces a record chosen from its table by the scheduler on to a (lossy) channel with capacity C, according to some specified scheduling algorithm. One or more subscribers tune into the channel to receive updates from the publisher. On receiving an announcement, each subscriber that has joined the channel updates its local copy of the table. An expiration time is associated with each data item stored at the receiver. If an update is not received before the timer expires, the entry is deleted (and in practice an external notification event is generated).

The set of all data items in the sender's table at any given instant t, is termed the *live data set*, $L(t)$. An update process at the publisher adds records to its table. Each record is also associated with a lifetime after which the publisher ceases to announce it and the record is eliminated from both the sender's and receivers' tables. Figure 1 illustrates this model.

2.1 Consistency

Unlike ARQ, where receipt of an acknowledgement explicitly indicates state synchronization between sender and receiver, a soft state protocol generally provides no feedback to the sender as to what the receiver has successfully received. Instead, the end systems simply participate in the announce/listen process and the assumption is that the receiver's data store converges to a consistent state over time. Many protocols based on this premise have been described and some characterize this property as *eventual consistency* [20, 22], but a formal definition for this has not yet been proposed.

To measure the effectiveness of soft state protocols using our model, we introduce the *consistency metric*, $c(k, t)$, defined per live $\{key, value\}$ pair $\{k, val(k)\}$ for processes P and Q communicating over a loss-prone network as the probability that both processes have the same value for a given key. This is denoted as,

$$c(k,t) \quad = \quad Pr.[P.val(k) = Q.val(k)], \quad 0 \le c(k,t) \le 1$$

where $P.val(k)$ is P's value for key k.

The *instantaneous system consistency*, $c(t)$ at a given instant, t, is defined as the average consistency measured across all live data items at that instant.

$$c(t) \quad = \quad \frac{\sum_{k \in L(t)} c(k,t)}{|L(t)|}$$

The *average system consistency* is the time average of the instantaneous system consistency over the entire lifetime of a system.

$$E[c(t)] \quad = \quad \lim_{T \to \infty} \frac{\int_{t=0}^{t=T} c(t)dt}{T}$$

The definition of $E[c(t)]$ above also provides us with a method to empirically compute the consistency metric of a system — as the time average of $c(t)$ over long durations.

A protocol is said to be *eventually consistent* if this probability approaches 1 in the long run, after the item is introduced into the system, i.e.,

$$\frac{\sum_{k \in K} \lim_{t \to \infty} c(k)}{|K|} \quad \approx \quad 1$$

Another important metric of protocol performance is the average latency from the instant a new or updated $\{key, value\}$ pair is introduced into the system to the first time it is received correctly at the receiver. We call this the receive latency T_{rec}.

In the remainder of this paper, we present several analytical and simulation results showing the dependence of the consistency metric on packet loss rates, available bandwidth, and announcement workloads. This dependence is termed a *consistency profile*.

Many protocols based on soft state rely on nothing more than the announce/listen mechanism for maintaining consistency in the face of packet loss. This simple open-loop repetitive announcement process transmits state updates in a quasi-reliable manner from an announcer to a listener over a loss-prone network. For a static input at the source, announce/listen provides a simple form of reliability since eventually the receiver's state will match the sender's once all the records have been successfully transmitted.

The simple open-loop periodic retransmission scheme provides an extremely simple substrate for "quasi reliable" systems. It is an attractive alternative to ARQ-based reliable transport protocols, especially in the case of multicast, since managing receiver feedback scalably in large groups continues to remain a formidable challenge. For example, it has been successfully used in the the multicast-based session directory tools [26, 24] to disseminate MBone conference information to large groups. However, pure open-loop periodic retransmissions are redundant and do not use bandwidth efficiently. The challenge for the so called "soft state" transports, including the announce/listen protocol, is therefore to (i) maximize system consistency and minimize user-perceived latency in receiving data items, and (ii) minimize redundant transmissions. In the following sections, we evaluate several soft state-based transports and show how to optimize them for given network conditions using adaptive scheduling techniques.

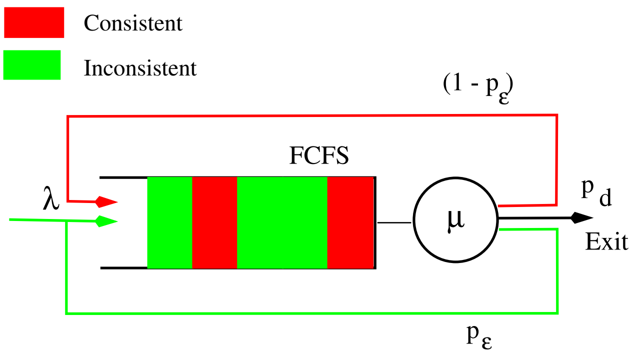

Figure 2: Queueing model for announce/listen-based transport.

3 "Open-Loop" Announce/Listen Protocol

To evaluate the performance of soft state systems that use open-loop announce/listen for data transport, we develop an analytic model based on classed queueing networks [5].

The parameters for our model are: p_ϵ, the probability that an announcement transmitted on the channel is lost by one or more subscribers, or the average per-transmission channel loss rate. Since the consistency metric treats all successful transmissions identically, regardless of their relative position in the transmitted stream, it is sufficient to specify the average packet loss rate. The metric is insensitive to the exact pattern of losses, but is only affected by the mean of the packet loss process. In contrast to other application- and media-specific metrics such as rate-distortion model [14] for multimedia signals that are sensitive to loss patterns in addition to the average drop rate, our metric is more general and applies to a wider class of systems.

In addition, we also assume that these soft state transports are ALF-based, in that individual transmissions are independent application data units (ADUs) [13]. In addition, the transport protocol does not enforce in-order delivery of packets at the receiver, which allows us to ignore the effects of packet reordering in our model, even though, in reality, though receivers suffer extra latency when individual fragments of a large ADU are reordered.

λ is the average rate of update of the publisher's table, and μ_{ch}, the available session bandwidth for this source's announcements. We model the network as a single server queue with multiple job classes or states. Each record goes through the following stages.

- Records enter the system in the "inconsistent" state, since the receiver has no knowledge of them. New records are inserted at the end of the transmission queue and the sender is assumed to have sufficient buffer space to hold all arriving announcements.
- The transmission channel acts as a server with service rate μ_{ch} and uses FIFO scheduling.
- The record changes state to "consistent" when an announcement containing it successfully reaches the receivers (with probability $(1-p_\epsilon)$).
- Each record has a bounded lifetime after which it is expired from the sender and receivers. For example, in session directories, announcements expire when the associated conference session ends. After obtaining service, an announcement exits the system with probability p_d, its death probability. The data expiration process is an inherent characteristic of the workload, and governs this death probability. In our model, we approximate the expiration process using a fixed and independent death probability per packet event though this does not take into account the possibility that an older record is more likely to expire than a new one.

	I/Exit	C/Exit	Death/Exit
I/Enter	$p_\epsilon(1-p_d)$	$(1-p_\epsilon)(1-p_d)$	p_d
C/Enter	0	$(1-p_d)$	p_d

Table 1: State change probabilities

Table 1 lists the probability of state change between consistent and inconsistent as a record leaves the server.

If $n_C(t)$ and $n_I(t)$ denote the number of consistent and inconsistent records in the system at any instant, the consistency metric for this system is the time average of the fraction $\frac{n_C(t)}{n_C(t)+n_I(t)}$. Computing the net flow $\hat{\lambda}_I$ and $\hat{\lambda}_C$ into the queue for each class, we get:

$$\begin{aligned}\hat{\lambda}_I &= \lambda + p_\epsilon(1-p_d)\hat{\lambda}_I \\ \hat{\lambda}_C &= (1-p_\epsilon)(1-p_d)\hat{\lambda}_I + (1-p_d)\hat{\lambda}_C\end{aligned}$$

Solving the above system of equations yields,

$$\begin{aligned}\hat{\lambda}_I &= \frac{\lambda}{1-p_\epsilon(1-p_d)} \\ \hat{\lambda}_C &= \frac{(1-p_\epsilon)(1-p_d)}{p_d}\hat{\lambda}_I \\ &= \frac{(1-p_\epsilon)(1-p_d)\lambda}{p_d(1-p_\epsilon(1-p_d))}\end{aligned}$$

Now,

$$\begin{aligned}\hat{\lambda} &= \hat{\lambda}_I + \hat{\lambda}_C \\ &= \frac{\lambda}{p_d}\end{aligned}$$

We first use Jackson's theorem [5] in the following steps for the single queue system with multiple job classes to compute the joint probability distributions of the number of consistent and inconsistent jobs.

$$p(n_I, n_C) = \frac{(n_I+n_C)!}{n_I! n_C!} \times \frac{\hat{\lambda}_I^{n_I}\hat{\lambda}_C^{n_C}}{\hat{\lambda}^{n_I+n_C}} \times (1-\rho)\rho^{n_I+n_C}$$

where, $\rho = \frac{\hat{\lambda}}{\mu_{ch}}$. The solution is valid only when $\rho < 1 \Rightarrow p_d > \frac{\lambda}{\mu_{ch}}$.

The *average system consistency* $E[c(t)]$ is then given by:

$$\begin{aligned}E[c(t)] &= \sum_{n_I+n_C>0} \frac{n_C}{n_I+n_C} \times p(n_I, n_C) \\ &= \sum_{k\geq 0} \frac{\hat{\lambda}_C}{\hat{\lambda}_I+\hat{\lambda}_C}(1-\rho)\rho^{k+1} \\ &= \frac{\hat{\lambda}_C}{\mu_{ch}} \\ &= \frac{(1-p_\epsilon)(1-p_d)}{1-p_\epsilon(1-p_d)} \times \frac{\lambda}{p_d\mu_{ch}}\end{aligned}$$

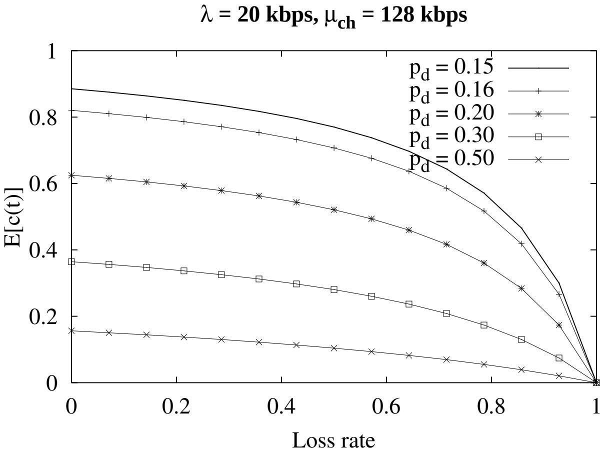

Figure 3: Consistency degrades with increasing packet loss rate and announcement death rate. A workload with a 15% death rate is 95% consistent for error rates in 1-10%.

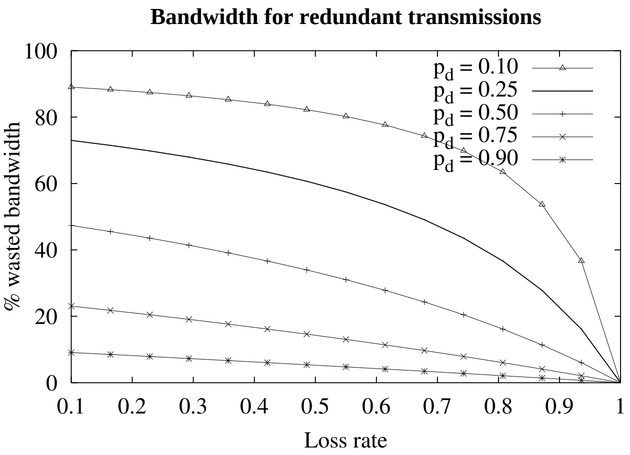

Figure 4: At loss rates between 0-20% and an announcement death rate of 10%, about 90% of the total available bandwidth is wasted.

Figure 3 shows $E[c(t)]$ graphically for different loss rates and announcement death rates. For a given death rate, as expected, we find that the system consistency goes down as the channel loss rate increases. We also observe that consistency falls sharply when the announcement death rate increases since data items are too short-lived to be propagated successfully to receivers. As seen in Figure 3 the system consistency lies between 85% and 95% for loss rates in the 1-10% range and an announcement death rate of 15%.

The open-loop announce/listen protocol analyzed above treats all data items — old and new — alike, retransmitting data items that may have already been received by the members of the group. From our model, we find that the fraction of bandwidth consumed by redundant transmissions is given by:

$$\begin{aligned} w &= \frac{\hat{\lambda}_C}{\hat{\lambda}} \\ &= \frac{(1-p_\epsilon)(1-p_d)}{1-p_\epsilon(1-p_d)} \end{aligned}$$

Figure 4 shows this result graphically. At loss rates of up to 50% and a death rate of 10%, over 90% of the total bandwidth is wasted on redundant retransmissions.

In reality, periodic source-based retransmissions are not entirely wasteful and benefit late joiners in an ongoing multicast session by reducing the delay such receivers experience in "catching" up with the rest of the session. Even in the case of unicast transmission, periodic source announcements allow the receiver to reconstruct the data store following a crash. Several techniques can be applied to the basic open-loop protocol to improve its consistency. In Sections 4 and 5, we show that maintaining multiple transmission queues at the sender and adding receiver feedback in a controlled manner allow for better bandwidth management at the sender.

4 Multiple Transmission Queues

Since redundant transmissions of previously consistent data items do not contribute to system consistency, one way to improve the performance of the basic open-loop announce/listen protocol is to reduce the fraction of bandwidth for repeated retransmissions. We do this by differentiating new and old data items. Several policies exist for ageing data items, but we consider one simple ageing scheme with two transmission queues for our analysis. We refer to the two transmission queues as the "hot" (or foreground) queue for new data items, and the "cold" (or background) queue for data items that have been transmitted at least once from the sender. The available data bandwidth is shared between the two queues proportionally (e.g., using a randomized lottery scheduler [51], weighted fair queueing [17] or stride scheduling [52]). Proportional sharing is preferred over strict priority scheduling since it prevents starvation of cold data items in the background transmission queue. Bandwidth allocated to foreground transmissions directly increases the likelihood that a new data item is successfully delivered, and hence contributes to system consistency. Unused excess hot bandwidth is consumed by transmissions from the cold queue.

We evaluate the consistency of this scheme using simulations [1]. Our simulations comprise a single sender and single receiver with a lossy communication channel. Having two transmission queues raises the important issue of allocating the total data bandwidth μ_{data} for the hot (μ_{hot}) and cold (μ_{cold}) queues and our simulations quantify the impact of this bandwidth allocation policy on system consistency.

Figure 5 shows the impact of increasing μ_{hot}, the bandwidth allocated to foreground transmissions, when μ_{data}, the total data bandwidth is held fixed. The results show that increasing μ_{hot} has a positive effect on the average system consistency, but only while $\mu_{hot} > \lambda$ (upto about 40%, in this experiment). The sender must allocate sufficient bandwidth to new data arriving at rate λ, to prevent the hot queue from growing indefinitely. The optimal consistency level is reached for $\mu_{hot} \geq \lambda$. However, as we see from

[1] Unfortunately, this extended model with two-level scheduling is not analytically tractable using Jackson's theorem.

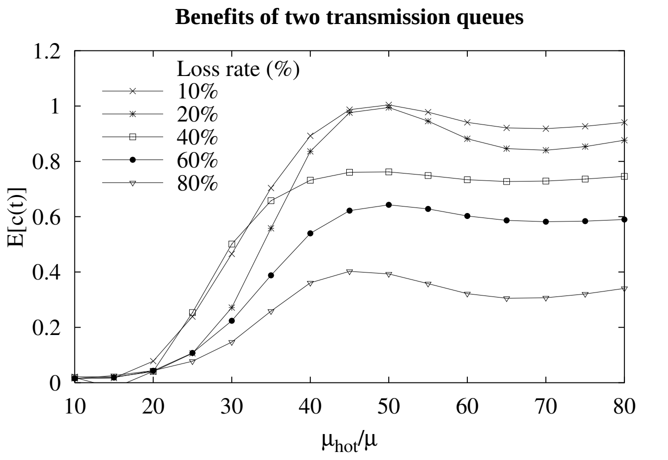

Figure 5: Two-level scheduling improves consistency by 10% to 40%. $\mu_{data} = 45kbps$, $\lambda = 15kbps$. Consistency is maximum when $\mu_{hot} > \lambda$.

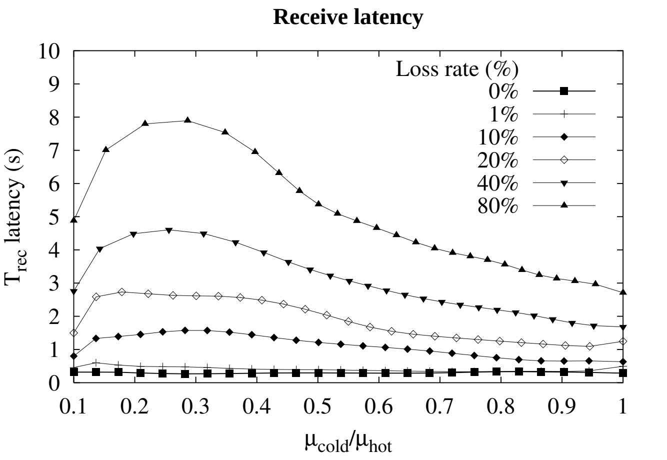

Figure 6: Increasing the cold bandwidth reduces queueing delay. When $\mu_{cold} >> \mu_{hot}$, no data item is retransmitted resulting in low latencies for successful transmissions.

Figure 5, consistency does not significantly change as we increase the bandwidth for the hot queue beyond λ.

The benefits of cold retransmissions are in the form of reduced average receive latency. We study the effect of increasing μ_{cold} (and hence mu_{data}), while maintaining μ_{hot} at its optimal level, just higher than the arrival rate. From Figure 6, we find that the receive latency T_{rec} initially increases, but drops as more bandwidth is added for background transmissions. This is due to two competing effects:

(i) At low values of μ_{cold}, all successful transmissions experience very small latency. When $\mu_{cold} \approx 0$, data items are never retransmitted and all successfully delivered items thus experience low delay. Since the average T_{rec} is measured only over all successful transmissions, our measurement excludes data items that take indefinitely long to reach the receiver, and hence the apparently low latency. The 300 ms latency for $\mu_{cold}/\mu_{data} \leq 1\%$ is explained by approximating the system as a single-server single-queue system with bandwidth $\mu_{hot} \approx \mu_{data}$. With exponential interarrivals and service times, the average latency is $E[w] = \frac{\rho}{\mu_{data}(1-\rho)}$. However, without retransmissions ($\mu_{cold} \approx 0$), and in the face of high loss rates, a significant fraction of data is never successfully transmitted, resulting in a low average consistency. Hence, turning off background transmissions is detrimental to system performance especially in the face of high loss rates.

(ii) Increasing the cold bandwidth increases the likelihood of a successful retransmission and, therefore, reduces T_{rec}, as shown in Figure 6.

5 Impact of Receiver Feedback

The inefficiency of the open-loop protocols discussed in Section 4 stems from the source's incomplete knowledge of receiver state. Adding receiver feedback attempts to remedy this by communicating receiver status back in order to improve bandwidth management at the sender. In this section, we discuss our simulation results quantifying the impact of adding receiver feedback in the form of negative acknowledgments (NACKs) to the original announce/listen framework. Once again, our simulations have one sender and one receiver. We find that adding feedback can improve consistency by 10% to 50% for loss rates between 5% and 40%.

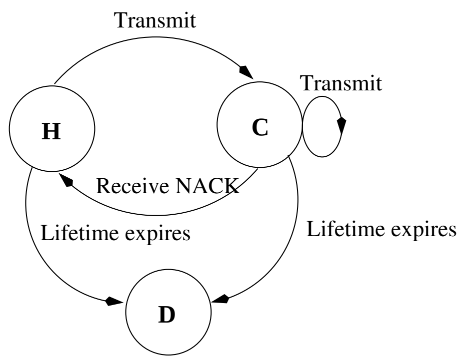

Figure 7: State transitions between "hot" (H), "cold" (C), and "dead" or invalid (D) states.

The sender maintains two transmission queues — (i) a hot (or foreground) queue that contains data that is thought to be inconsistent, and (ii) a cold (or background) queue for repeated retransmissions. As in the previous cases, *late joiners* who need to catch up with the current state of an ongoing session benefit from repeated retransmissions. Data items get scheduled for transmission as follows: a new data item is transmitted through the foreground queue, and subsequently moved to the background queue, as shown in Figure 7. The two queues share the available data bandwidth proportionally, and we control this allocation in our experiments. The receiver generates a NACK upon detecting a loss. In response to the NACK, the sender schedules a retransmission of the requested data item, by moving it from the cold queue to the tail of the hot queue. Hence, hot band-

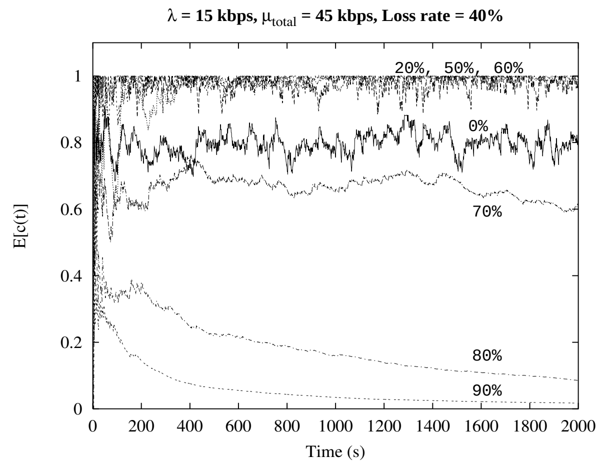

Figure 8: In open-loop ($\mu_{fb}/\mu_{tot} = 0$), consistency is about 80%. When $\mu_{fb}/\mu_{tot} = 20 - 60\%$, consistency reaches 99%. At higher values, when insufficient bandwidth is available for data, consistency collapses.

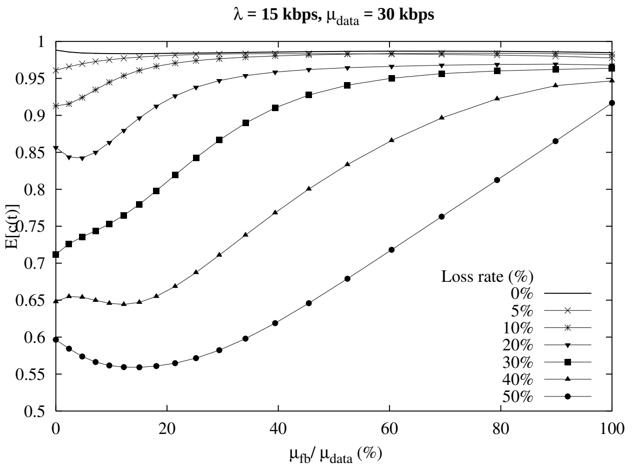

Figure 9: Consistency is improved by allocating sufficient bandwidth for feedback. At loss rates over 50%, allocating additional feedback bandwidth feedback reduces consistency.

width is allocated to new data items and retransmissions requested by the receiver, while cold bandwidth is used for background retransmissions of previously transmitted data.

- **Data vs. feedback.** We simulate the effect of increasing the fraction of total bandwidth allocated for feedback and find that adding feedback improves system consistency from 60% to almost 98% at a loss rate of 40%. Figure 8 shows these results. Allocating a small fraction of the total available bandwidth for feedback messages significantly improves system consistency. Consistency is maximum (at close to 100%, in this example) when sufficient bandwidth is available to transmit NACKs generated in response to data loss. Beyond this threshold level, consistency drops rapidly as the feedback bandwidth grows at the expense of the sender's data bandwidth. For example, in Figure 8, when feedback receives 70% of the total bandwidth, the system consistency collapses rapidly.

  We also study the impact of adding feedback bandwidth, while maintaining μ_{data} fixed and find that the average system consistency increases by about 10% when the loss rate is about 10% and by 50% for even higher loss rates ($\geq 50\%$). This is shown in Figure 9. Consistency reaches a maximum between 90% and 100% depending on the loss rate, indicating that increasing the feedback bandwidth beyond this threshold level does not significantly affect consistency.

  Since the packet loss rate also affects the optimal data vs. feedback allocation, the protocol must monitor loss rates via receiver reports and use this information to adaptively reallocate bandwidth to maintain this "optimal" consistency level.

- **Hot vs. cold bandwidth.** To manage the available data bandwidth at the sender, we study the impact of allocating bandwidth to hot and cold data queues. In Figure 10, we find that the consistency metric remains close to 5% as long as the arrival rate exceeds μ_{hot}. When μ_{hot} is increased beyond λ, the consistency sharply rises to almost 100%. Increasing μ_{hot} beyond λ does not have a significant impact on the consistency metric. Hence, $\lambda \leq \mu_{hot}$ is the optimal region beyond which the marginal benefit from additional bandwidth to the "hot" queue is limited and below which system consistency shows marked degradation. If the system's consistency metric is to be maximized, the application must adhere to its allowed maximum level. Later, in Section 6, we show how our transport framework uses this to notify the application to refrain from injecting new records if system consistency is to be maximized.

- **Effect of loss rate.** Since the channel loss rate indirectly affects the number of NACKs and hence the number of retransmissions, we study the impact of loss rate on the consistency metric, varying the sender's bandwidth allocation between its two transmission queues. From Figure 11, we see that the loss rate limits the maximum consistency that can be attained with a given amount of total bandwidth, regardless of how it is scheduled between the hot and cold transmissions. However, the relative proportion of hot vs. cold bandwidth does not significantly affect consistency, once sufficient bandwidth is available to absorb new arrivals.

  The consistency profiles discussed here influence bandwidth management. In Section 6 we elaborate on a profile-driven allocation scheme that aims to utilize the available bandwidth optimally.

6 A Soft State Transport Framework

Conventional reliable transport protocols like TCP are built on "hard" protocol state at the end points and export a single restrictive interface to the application — that of a sequenced, in-order, byte-stream. While some extensions to relax the constraints of TCP have been proposed, the underlying abstraction provided by TCP is rigid and does not lend itself to arbitrary application customization. For example, extensions to TCP byte sequence numbers to support

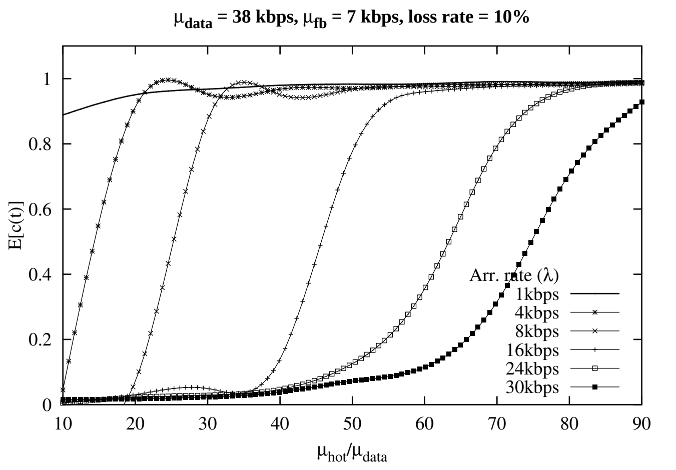

Figure 10: $\lambda \leq \mu_{hot}$ is the optimal region beyond which the marginal benefit from additional bandwidth to the "hot" queue is limited and below which system consistency shows marked degradation.

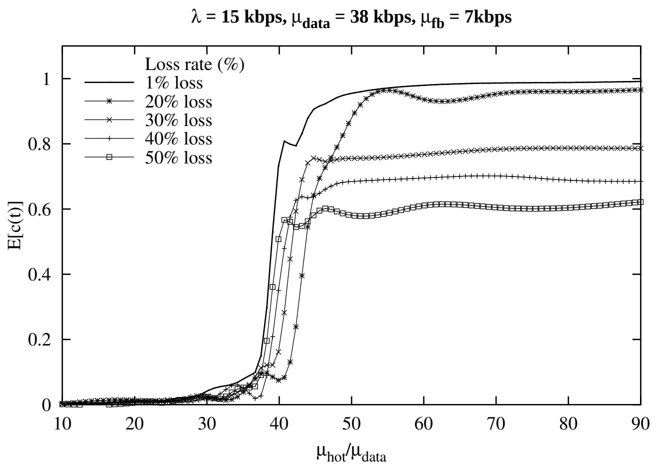

Figure 11: The system consistency shows a "knee", beyond which the marginal benefit from additional bandwidth to the "hot" queue is limited and below which system consistency shows marked degradation.

application-defined records is not straightforward. (See [18] for an interesting discussion on this.) Motivated by this, we propose a new framework for reliable transport protocols whose behavior, e.g., degree of reliability, message ordering semantics, bandwidth allocation policies, can be customized by the application. Using our formalism of soft state, we propose the *soft state transport protocol* framework (SSTP) for reliable data transport. To the best of our knowledge, this is the first soft state transport protocol whose properties can be predicted using a model.

In contrast to the conventional approach to transport design, the SSTP architecture is guided by ALF and exposes a powerful, yet simple programming interface allowing it to be tailored to the needs of the user application. SSTP aims to provide the necessary interface and mechanisms for an application to control the degree of reliability and message delivery semantics. An SSTP sender transmits original application data as well as periodic soft state announcements summarizing all previously transmitted data. SSTP receivers use NACKs to report lost data items to the sender, which in response performs the appropriate retransmissions. SSTP may be applied to multicast as well as unicast transport. In the case of multicast, a scalable mechanism such as slotting and damping [11, 20] may be used in managing feedback traffic.

The following two salient features of SSTP are described in this section: (i) application-controlled bandwidth management, and (ii) a hierarchical data model to efficiently support large data stores.

6.1 Application-controlled Bandwidth Allocation

SSTP provides a parameterized framework to schedule available bandwidth between data and feedback messages appropriately to achieve consistency levels desired by the application. Based on the amount of bandwidth allocated to data and announcements (or, "cold" data), a continuum of consistency levels is provided. SSTP uses measured packet loss rates using RTCP-style receiver reports and empirically derived *consistency profiles* to carefully control bandwidth allocation.

SSTP does not attempt to perform congestion control nor determine the total available data rate to the session member, but rather, relies on a congestion management module, such as the CM [3], to obtain this information. SSTP merely decides how this available bandwidth is to be used by the application and transport protocol. While most existing research addresses the issue of detecting network congestion and reacting to it by lowering the transmission rate (or reception rate, as in layered multicast) [27, 32, 31, 9, 48, 8, 34, 42], the issue of how best to utilize available bandwidth in reliable transport has received far less attention. Even though this decision is generally application-specific, we can use the consistency metric for a large class of applications that fit the data model described previously in Section 2.

Rather than treat all data as equal, SSTP allows the application to reflect its priorities into the data transport protocol. Using a hierarchical scheduler (e.g., CBQ [19] or H-FSC [47]), the application flexibly controls the amount of bandwidth allocated to its different data classes. Figire 12 shows an example of such an allocation hierarchy. An application can experience the maximum possible consistency under given network loss rates by scheduling its available session bandwidth based on consistency profiles derived from our model. Consistency profiles predict system consistency for given network loss conditions and announcement characteristics.

Using stored consistency profiles similar to Figure 9, the bandwidth allocator outputs values $\{\mu_{data}, \mu_{feedback}\}$. The share of bandwidth for the different transmission queues is obtained from the T_{rec} profile, similar to Figure 6. The allocator also notifies the application if it detects that rate of arrival of new data from the application exceeds the bandwidth available for it, i.e., μ_{hot}. This dictates the maximum rate at which the application can send to maintain the requested level of consistency. This notification from SSTP gives the application an opportunity to adapt to the rate constraint in the best possible application-specific manner.

SSTP uses the following information in making band-

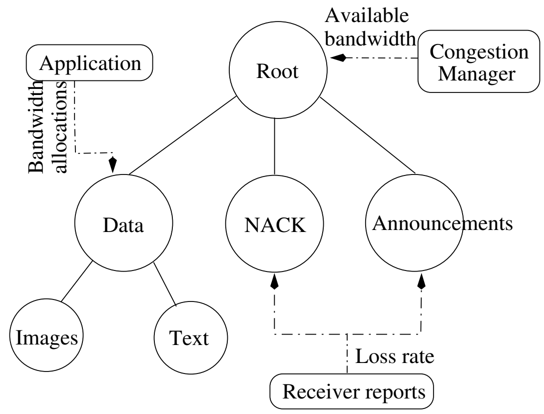

Figure 12: Profile-driven scheduler for application adaptation.

width allocation decisions:

- The average packet loss rate, periodically obtained from RTCP-like receiver reports;
- The application's consistency target (e.g., 90% consistency), and optionally a "soft" delay requirement[2] for individual data items;
- The total available session bandwidth, either configured manually as in most non-TCP applications today (e.g., the MBone video conferencing applications [35, 30, 29] and the Real Audio^{TM} player [41]) or available from a congestion control algorithm.

6.2 Hierarchical Data Model

Our simple data model presented in Section 2 fits a number of existing systems such as routing updates and the current session directory protocol. However, if such soft state systems are to scale to extremely large systems, the "table of key-value pairs" model needs to be refined. In order to scale announcement-driven loss recovery to applications with large data sets, SSTP supports hierarchical namespaces. The SSTP hierarchy provides a good summarization structure for soft state announcements. Such a hierarchy maps to logically independent objects within an application and allows such objects to be treated independently during loss recovery. Since the sturcture of application data is exposed to SSTP, this eliminates the undesirable coupling induced by a TCP-like in-order byte-stream abstraction.

An SSTP namespace is a hierarchical index structure over the set of application data units generated by a sender. Each namespace node, n, is associated with a fixed-length summary or digest of the subtree rooted at it, and is computed recursively using the one-way hash function h (e.g., MD5 [43]) as shown below:

$$s(n) = \begin{cases} \text{right\_edge}(n), \text{ if } n \text{ is a leaf-level ADU} \\ h(s(c_1), s(c_2), \ldots, s(c_k)), \text{ otherwise} \end{cases}$$

[2] While SSTP does not guarantee end-to-end delay, it uses delay information as hints to determine the operating region in the T_{rec} profile.

If n is a leaf-level ADU, right_edge(n) is the number of bytes transmitted from ADU n by the sender. If n is an internal node, $c_1, c_2, \ldots, c_k$ are immediate children of n in the namespace hierarchy.

We briefly describe our scalable announcement protocol that uses the namespace summaries to reduce the number of messages in detecting and recovering from losses. A sender transmits new data upon arrival from the application. In addition, the sender also carries out "cold" transmissions of the root summary. Upon receiving a summary announcement, if a receiver detects a mismatch at the root namespace node, a feedback message requesting further namespace repair is scheduled for transmission. In response to such a feedback query from the receiver, the sender (or any participant in a multicast session), responds with a set of next level signatures. In this manner, loss recovery proceeds recursively down the namespace hierarchy.

An additional advantage of the recursive descent procedure is that a receiver may refrain from requesting further repair along a branch if there is no application-level "interest" for data items belonging to it. For example, a PDA browser may not wish to repair high resolution image data types. The sender communicates such hints to the receivers using application-level meta-data tags associated with the namespace nodes. Receiver-driven reliability using such application-level data names is described in detail in [40].

Our hierarchical data model for SSTP simultaneously solves the namespace scaling problem and provides a rich naming structure that is amenable to ALF. By controlling the bandwidth allocated to original data transmissions and summary announcements, we can control the level of consistency and latency to recover lost items.

7 Related Work

Chandy et al. [10] formally define soft state probabilistically and use it as a primitive for exchanging state information for distributed resource management. However, their soft state model is restricted to continuous state variables and their main innovation is in the application of estimation techniques to infer state values between state updates. Their work does not relate the model to existing soft state-based protocols such as announce/listen.

Sharma et al. [46] study the general problem of scalable timers in soft state protocols and present an adaptive algorithm for (i) dynamically adjusting the sender's refresh rate, and (ii) estimating the sender's transmission rate at the receiver to determine timeouts for ageing out state.

In [20], Floyd et al. describe the Scalable Reliable Multicast protocol as being eventually consistent. The authors propose an SRM framework, in which data is expired using application hints, analogous to our death process. Handley et al. [23] list eventual consistency as one of the goals of the shared state in the network text editor, NTE. However, neither paper provides an evaluation of system consistency. In [22], Handley demonstrates that adding feedback in the form of "address clash reports" to detect and correct address clashes in an announce/listen-based address allocation protocol can greatly increase its scalability to larger groups. Even though this is a specific case, it motivates us to study the more general case of adding feedback for reliability. In our work, we study the impact of adding receiver feedback in the more general context of soft state transport protocols.

The notion of "probabilistic reliability" was also proposed by Birman *et al.* [7] in their work on *bimodal mul-*

ticast, in which receivers recover a lost stream of items in reverse order. This scheduling choice makes the protocol more stable in large groups, and provides bimodal delivery guarantees. — i.e., almost all or very few members receive each transmission (a probabilistic version of the "all or none" atomic broadcast [6]). Our work differs from this in that it is not restricted to multicast transport. In our framework, the delivery of a given piece of data is probabilistic — there is a predictable and tunable likelihood of reception.

Amir et al. [1] present SCUBA, a consensus-based bandwidth allocation strategy for multicast video, where sources gather receiver votes in a scalable fashion to adjust transmission rates. Our work also addresses the issue of receiver-based bandwidth allocation, however, we focus on reliable multicast transports with hierarchically structured data stores. In [2] Amir et al. also introduce the notion of soft state gateways and multiple transmission queues for the scalable exchange of RTCP-like control traffic between islands of high network high bandwidth bridged by low bandwidth links. However, their work does not analyze the performance nor investigate the tradeoffs between different allocation policies. This scheme is a specific instantiation of our more general parameterized SSTP framework.

8 Concluding Remarks

In this paper, we have presented a model based on Jackson queueing networks that formalizes the notion of soft state. Based on this model, we have introduced a new consistency metric, a probabilistic measure of the effectiveness of different protocol variants, from "open-loop" announce/listen-style communication to feedback-based reliable transport. We show that consistency improves by 10-40% by appropriately aging data items and allocating progressively lower bandwidths for older data. This technique of distinguishing new from old data in combination with receiver feedback in the form of negative acknowledgments improves consistency by 12-50%. In each of these cases, we have shown the optimal bandwidth allocation for which the available bandwidth is best utilized in terms of the consistency metric.

Using the consistency metric as our basis, we apply these results to the design of an adaptive framework for soft state transport protocols (SSTP). SSTP provides a continuum of reliability "levels" that can be customized by the application. It also includes a profile-driven allocation algorithm that uses measurements of network loss rates to adapt to the optimal bandwidth allocation for the required consistency. While SSTP does not solve the problem of determining the available bandwidth, it uses consistency profiles derived from our soft state model to best utilize this bandwidth. In addition, SSTP incorporates application-level framing principles to provide a flexible and powerful primitive for applications to reflect their performance preferences into the protocol machinery.

References

[1] Amir, E., McCanne, S., and Katz, R. Receiver-driven Bandwidth Adaptation for Light-weight Sessions. In *Proceedings of ACM Multimedia '97* (Nov. 1997), ACM.

[2] Amir, E., McCanne, S., and Katz, R. An Active Service Framework and its Application to Real-time Multimedia Transcoding. In *Proceedings of SIGCOMM 1998* (Vancouver, Canada, Sep 1998), ACM.

[3] Balakrishnan, H., Rahul, H. S., and Seshan, S. An Integrated Congestion Management Architecture for Internet Hosts. In *Proceedings of SIGCOMM 1999* (Cambridge, MA, Sep 1999), ACM.

[4] Banerjea, A., Ferrari, D., Mah, B., Moran, M., Verma, D., and Zhang, H. The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences. *IEEE/ACM Transactions on Networking* (1995).

[5] Baskett, F., Chandy, M., Muntz, R., and Palacios, F. Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. *Journal of the Association for Computing Machinery 22*, 2 (1975), 248–260.

[6] Birman, K., Chiper, A., and Stephenson, P. Lightweight Causal and Atomic Group Multicast. *ACM Transactions on Computer Systems 9*, 3 (Aug. 1991), 272–314.

[7] Birman, K., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M., and Minsky, Y. Bimodal Multicast. Tech. Rep. TR98-1683, Cornell University, Ithaca, NY, May 1998.

[8] Bolot, J.-C., and Turletti, T. A Rate Control Mechanism for Packet Video in the Internet. In *Proceedings IEEE Infocom '94* (Toronto, Canada, June 1994), ACM.

[9] Bolot, J.-C., Turletti, T., and Wakeman, I. Scalable Feedback Control for Multicast Video Distribution in the Internet. In *Proceedings of SIGCOMM '94* (University College London, London, U.K., Sept. 1994), ACM.

[10] Chandy, K. M., Rifkin, A., and Schooler, E. Using Announce-Listen with Global Events to Develop Distributed Control Systems. *Concurrency: Practice and Experience* (1998), 1021–1027.

[11] Chesson, G. XTP/Protocol Engine Design. In *Proceedings of the IFIP WG6.1/6.4 Workshop* (Rüschlikon, May 1989).

[12] Clark, D. D. The Design Philosophy of the DARPA Internet Protocols. In *Proceedings of SIGCOMM '88* (Stanford, CA, Aug. 1988), ACM.

[13] Clark, D. D., and Tennenhouse, D. L. Architectural Considerations for a New Generation of Protocols. In *Proceedings of SIGCOMM '90* (Philadelphia, PA, Sept. 1990), ACM.

[14] Cover, T. M., and Thomas, J. A. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[15] Deering, S., Estrin, D., Farinacci, D., and Jacobson, V. An Architecture for Wide-Area Multicast Routing. In *Proceedings of SIGCOMM '94* (University College London, London, U.K., Sept. 1994), ACM.

[16] Delgrossi, L., Herrtwich, R. G., , Hoffmann, F. O., and Schaller, S. Receiver-initiated Communication with ST-II. *Multimedia Systems 2*, 4 (Oct. 1994), 141–149.

[17] Demers, A., Keshav, S., and Shenker, S. Analysis and Simulation of a Fair Queueing Algorithm. In *Proceedings of SIGCOMM '89* (Sept. 1989), ACM.

[18] Falk, A., and Paxson, V. Minutes of the "RUTS" IETF BOF, Dec. 1998. ftp://ftp.ee.lbl.gov/ietf/ruts-98-minutes.

[19] Floyd, S., and Jacobson, V. Link-Sharing and Resource Management Models for Packet Networks. *IEEE/ACM Transactions on Networking 3*, 4 (Aug. 1995), 365–386.

[20] Floyd, S., Jacobson, V., McCanne, S., Liu, C.-G., and Zhang, L. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *Proceedings of SIGCOMM '95* (Boston, MA, Sept. 1995), ACM.

[21] Handley, M. *SAP: Session Announcement Protocol*. Internet Draft, Nov 19, 1996.

[22] Handley, M. Session Directories and Internet Multicast Address Allocation. In *Proceedings of SIGCOMM 1998* (Vancouver, Canada, Sep 1998), ACM.

[23] Handley, M., and Crowcroft, J. Network Text Editor (NTE): A Scalable Shared Text Editor for the MBone. In *Proceedings of SIGCOMM 1997* (Cannes, France, Sep 1997), ACM.

[24] HANDLEY, M., AND JACOBSON, V. *sdr — A Multicast Session Directory*. University College London.

[25] HEDRICK, C. *Routing Information Protocol*. Rutgers University, June 1988. RFC-1058.

[26] JACOBSON, V. *Session Directory*. Lawrence Berkeley Laboratory. ftp://ftp.ee.lbl.gov/conferencing/sd.

[27] JACOBSON, V. Congestion Avoidance and Control. In *Proceedings of SIGCOMM '88* (Stanford, CA, Aug. 1988).

[28] JACOBSON, V. SIGCOMM '94 Tutorial: Multimedia conferencing on the Internet, Aug. 1994.

[29] JACOBSON, V., AND MCCANNE, S. *LBL Whiteboard*. Lawrence Berkeley Laboratory ftp://ftp.ee.lbl.gov/conferencing/wb.

[30] JACOBSON, V., AND MCCANNE, S. *Visual Audio Tool*. Lawrence Berkeley Laboratory. ftp://ftp.ee.lbl.gov/conferencing/vat.

[31] JAIN, R. Congestion Control in Computer Networks: Issues and Trends. *IEEE Network Magazine* (May 1990), 24–30.

[32] JAIN, R., RAMAKRISHNAN, K., AND CHIU, D.-M. Congestion Avoidance in Computer Networks With a Connectionless Network Layer. Tech. Rep. DEC-TR-506, Digital Equipment Corporation, Aug. 1987.

[33] LOUGHEED, K., AND REKHTER, Y. *A Border Gateway Protocol (BGP)*. Cisco Systems and T. J. Watson Research Center, IBM Corp., June 1989. RFC-1105.

[34] MCCANNE, S. *Receiver-driven Layered Multicast*. PhD thesis, University of California, Berkeley, Dec. 1996.

[35] MCCANNE, S., AND JACOBSON, V. *vic:* A Flexible Framework for Packet Video. In *Proceedings of ACM Multimedia '95* (Nov. 1995), ACM.

[36] MCQUILLAN, J., ET AL. A New Routing Algorithm for the ARPANET. *IEEE Transcations on Networking* (May 1980).

[37] MOCKAPETRIS, P. *Domain Names – Implementation and Specification*. SRI International, Menlo Park, CA, Nov. 1987. RFC-1035.

[38] PAXSON, V. End-to-End Routing Behavior in the Internet. In *Proc. ACM SIGCOMM '96* (Aug. 1996).

[39] POINTCAST INC. *PointCast Home Page*. http://www.pointcast.com.

[40] RAMAN, S., AND MCCANNE, S. Scalable Data Naming for Application Level Framing in Reliable Multicast. In *Proceedings of ACM Multimedia '98* (Bristol, UK, Sept. 1998), ACM.

[41] REALNETWORKS, INC. *RealPlayer*. http://www.real.com/.

[42] REJAIE, R., HANDLEY, M., AND ESTRIN, D. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. *IEEE Infocom* (1999).

[43] RIVEST, R. *The MD5 Message-Digest Algorithm*. MIT Laboratory for Computer Science and RSA Data Security, Inc., 1992. RFC-1321.

[44] SCHOOLER, E. M. A Multicast User Directory Service for Synchronous Rendezvous. Tech. rep., California Institute of Technology, Pasadena, CA, August 1996.

[45] SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. *RTP: A Transport Protocol for Real-Time Applications*. Internet Engineering Task Force, Audio-Video Transport Working Group, Jan. 1996. RFC-1889.

[46] SHARMA, P., ESTRIN, D., FLOYD, S., AND JACOBSON, V. Scalable Timers for Soft State Protocols. In *Proceedings IEEE Infocom '97* (Kobe, Japan, 1997).

[47] STOICA, I., ZHANG, H., AND NG, T. S. E. A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time and Priority Service. In *Proceedings of SIGCOMM 1997* (Cannes, France, Sep 1997), ACM.

[48] TURLETTI, T., AND BOLOT, J.-C. Issues with Multicast Video Distribution in Heterogeneous Packet Networks. In *Proceedings of the Sixth International Workshop on Packet Video* (Portland, OR, Sept. 1994).

[49] Digital Subscriber Signalling System No. 1 (DSS 1) – ISDN User-Network Interface Layer 3 Specification for Basic Call Control, 1993. ITU-T Recommendation Q.931.

[50] Procedures for Establishing Communication between Three or more Audiovisual Terminals using Digital Channels up to 1920 kbit/s, 1997. ITU-T Recommendation H.243.

[51] WALDSPURGER, C. A., AND WEIHL, W. E. Lottery Scheduling: Flexible Proportional-Share Resource Management. In *First Symposium on Operating Systems Design and Implementation (OSDI)* (1995), USENIX Association, pp. 1–11.

[52] WALDSPURGER, C. A., AND WEIHL, W. E. Stride Scheduling: Deterministic Proportional-Share Resource. Tech. Rep. MIT/LCS/TM-528, MIT Laboratory for Computer Science, Cambridge, MA, June 1995.

[53] ZHANG, L., DEERING, S., ESTRIN, D., SHENKER, S., AND ZAPPALA, D. RSVP: A New Resource ReSerVation Protocol. *IEEE Network Magazine* (Sept. 1993), 8–18.