A Comparison of Hard-State and Soft-State Signaling Protocols

Ping Ji, Zihui Ge, Jim Kurose, *Fellow, IEEE*, and Don Towsley, *Fellow, IEEE*

***Abstract*—One of the key infrastructure components in all telecommunication networks, ranging from the telephone network to VC-oriented data networks to the Internet, is its signaling system. Two broad approaches towards signaling can be identified: so-called hard-state and soft-state approaches. Despite the fundamental importance of signaling, our understanding of these approaches—their pros and cons and the circumstances in which they might best be employed—is mostly anecdotal (and, occasionally, religious). In this paper, we compare and contrast a variety of signaling approaches ranging from "pure" soft state to soft-state approaches augmented with explicit state removal and/or reliable signaling, to a "pure" hard state approach. We develop an analytic model that allows us to quantify state inconsistency in single- and multiple-hop signaling scenarios, and the "cost" (both in terms of signaling overhead and application-specific costs resulting from state inconsistency) associated with a given signaling approach and its parameters (e.g., state refresh and removal timers). Among the class of soft-state approaches, we find that a soft-state approach coupled with explicit removal substantially improves the degree of state consistency while introducing little additional signaling message overhead. The addition of reliable explicit setup/update/removal allows the soft-state approach to achieve comparable (and sometimes better) consistency than that of the hard-state approach.**

***Index Terms*—Communication system signaling, hard-state, performance evaluation, soft-state.**

I. Introduction

ONE of the key infrastructure components in all telecommunication networks, ranging from the telephone network to VC-oriented data networks to the Internet, is its signaling system. Two broad approaches to signaling can be identified as so-called hard-state and soft-state approaches. Between these two extremes lie approaches that borrow various mechanisms from each. Despite the fundamental importance of signaling, our understanding of these two approaches, their pros and cons and the circumstances in which they might best be employed, is still not well understood.

Broadly speaking, we associate the term "soft-state" with signaling approaches in which installed state "times out" (and is removed) unless periodically "refreshed" by the receipt of a signaling message (typically from the entity that initially installed the state) indicating that the state should continue to remain installed. Since unrefreshed state will eventually timeout, soft-state signaling requires neither explicit state removal nor a procedure to remove orphaned state should the state-installer crash. Similarly, since state installation and refresh messages will be followed by subsequent periodic refresh messages, reliable signaling is not required. The term "soft-state" was coined by Clark [3], who described the notion of periodic state refresh messages being sent by an end system, and suggested that with such refresh messages, installed state could be lost in a crash and then automatically restored by subsequent refresh messages, all transparently to the end system, and without invoking any explicit crash-recovery procedures:

> "... the state information would not be critical in maintaining the desired type of service associated with the flow. Instead, that type of service would be enforced by the end points, which would periodically send messages to ensure that the proper type of service was being associated with the flow. In this way, the state information associated with the flow could be lost in a crash without permanent disruption of the service features being used. I call this concept "soft state," and it may very well permit us to achieve our primary goals of survivability and flexibility..."

Roughly speaking, then, the essence of a soft-state approach is the use of best-effort periodic state-installation/refresh by a state-installer and state-removal-by-timeout at the state holder. Soft-state approaches form the basis of numerous Internet protocols, including RSVP [19], SRM [8], PIM [5], [6], SIP [9], and IGMP [4].

"Hard-state" signaling takes the converse approach to soft state; installed state remains installed unless explicitly removed by the receipt of a state teardown message from the state installer. Since state remains installed unless explicitly removed, hard-state signaling requires a mechanism to remove orphaned state that remains after a state installer has crashed or departed without removing state. Similarly, since state installation and removal are performed only once (and without state refresh or state timeout), it is important for the state installer to know when state has been installed or removed. Reliable (rather than best effort) signaling protocols are thus typically associated with hard-state protocols. Roughly speaking, then, the essence of a hard-state approach is the reliable and explicit installation and removal of state information. Hard-state approaches have been taken in protocols such as ST-II [13], [17] and Q.2931b [14].

Between the extremes of a pure hard-state protocol and a pure soft-state protocol lie many protocols that have adopted elements of each approach. Indeed, protocols that were initially conceived as pure soft-state protocols have adopted a number of hard-state mechanisms (often as extensions) over time. For

Manuscript received April 27, 2004; revised March 6, 2006; approved by IEEE/ACM Transactions on Networking Editor K. Almeroth.

P. Ji is with the John Jay College of Criminal Justice, City University of New York, New York, NY 10019 USA (e-mail: pji@jjay.cuny.edu).

Z. Ge is with AT&T Labs Research, Florham Park, NJ 07932 USA (e-mail: gezihui@research.att.com).

J. Kurose and D. Towsley are with the University of Massachusetts, Amherst, MA 01003 USA (e-mail: kurose@cs.umass.edu; towsley@cs.umass.edu).

Digital Object Identifier 10.1109/TNET.2007.892849

example, in IGMPv1 [4], a soft-state timeout at a router was used to detect the departure of previously registered hosts; IGMPv2/v3 [7], [2] later added an explicit leave message to allow a host to explicitly inform the state-holding router of its departure. In the original RSVP [19], PATH, and RESV state installation messages were transmitted best effort under the assumption that the loss of a signaling message would be recovered via a later refresh message; ACK-based reliable signaling was introduced as an extension to RSVP in [1] and was also suggested in [12]. RSVP has also provided for explicit (although optional) removal of filter specifications since its conception. Hard-state protocols have adopted elements of the soft-state approach as well. In the ST-II hard-state signaling protocol, periodic HELLO messages serve to inform the HELLO sender that all is well with its neighbors, and that its own state that relies on a given neighbor is still valid, an implicit refreshing of its state.

Given the blurred distinctions between hard-state and soft-state approaches and the fact that protocols that fall into one category will adopt mechanisms typically associated with the other, *we believe that the crucial issue is not whether a hard-state or a soft-state approach is "better" in some absolute sense. Instead, we believe that the more fundamental question is to understand how the mechanisms that have been included in various hard-state and soft-state signaling protocols can best be used in given situations, and why. The goal of this paper is to answer this question.*

In this paper, we thus compare and contrast a variety of signaling protocols ranging from a "pure" soft-state protocol, to soft-state approaches augmented with explicit remote state removal and/or reliable signaling, to a "pure" hard-state protocol. We define a set of generic protocols that lie along this spectrum, and develop and analyze a general model that allows us to quantify a key performance metric associated with a given signaling protocol, the fraction of time that the state of the state-installer and the state-holder are inconsistent [15]. We also quantify the "cost" (both in terms of signaling overhead, and application-specific costs resulting from state inconsistency) associated with a given signaling approach and its parameter values (e.g., state refresh and removal timeout intervals). Among soft-state protocols, we find that a soft-state protocol coupled with explicit removal substantially improves state consistency, while introducing little additional signaling message overhead. The addition of reliable explicit setup/update/removal further allows a soft-state protocol to achieve comparable (and sometimes better) consistency than the hard-state protocol.

Our work focuses on evaluating the *performance* of different signaling protocols. However, there are other nonperformance-oriented issues associated with various signaling approaches (e.g., the complexity of protocol implementation), which may be examined by other means, but these are beyond the scope of this paper.

The remainder of this paper is structured as follows. In Section II, we describe five different signaling protocols that incorporate various hard-state and soft-state mechanisms, and qualitatively discuss the factors that will influence performance. Section III presents an analytic model for examining the performance of these approaches in the single-hop case, and compares their performance. Section IV considers the multihop case. Section V discusses related work. Finally, Section VI summarizes this paper and discusses future work.

II. Soft-State, Hard-State, and Protocols in Between

In this section, we describe the operation of five signaling protocols. These protocols differ in the manner in which they install, maintain, and remove state, and whether selected signaling messages are transported best effort or reliably. We consider a single node (henceforth referred to as the "sender") that wishes to install, maintain, and eventually remove (or have removed) signaling state at a remote node (that we will refer to as the "receiver"). We consider the simple, but illustrative, example of a signaling sender having a local state value that it wishes to install at the signaling receiver. When the sender state value equals the receiver's installed state value, we will say that the values are *consistent* [15]; otherwise, the sender and receiver state values are *inconsistent*. Our goal here is not to model a specific signaling protocol such as RSVP or Q2931b, but rather to capture the essential aspects of identifiably different approaches towards signaling. After describing the protocols, we then consider the performance metrics by which these protocols can be evaluated, and qualitatively discuss the factors that will impact performance.

We consider the following five approaches.

Pure Soft-State (SS): In this approach, the sender sends a *trigger* message [1] that contains state installation or update information to the receiver, and starts a state refresh timer (with value T). When the state-refresh timer expires, the sender sends a *refresh* message [19] and resets the refresh timer. Trigger and refresh messages are sent in a best-effort manner. When a trigger or refresh message is received at the receiver, the corresponding signaling state information is recorded and a *state-timeout timer* (with value X) associated with this state is started (or restarted if it was already running). Signaling state at the receiver is removed *only* when its state-timeout timer expires; that is, state will be maintained as long as the receiver continues to receive refresh messages before the state-timeout timer expires. This timeout could occur because the sender is no longer sending refresh messages (because its local state has been removed and it thus wants the remote state to be removed at the receiver), or because refresh messages have been lost in transmission, and have resulted in a state timeout at the receiver. We will refer to the latter case as *false removal* of state, since the sender did not intend for this state to be removed.

Soft-State With Explicit Removal (SS + ER) Signaling: SS + ER is similar to the SS approach, with the addition of an explicit state-removal message. When state is removed at the signaling sender, the sender sends a best-effort (unreliable) signaling message to the receiver carrying explicit state-removal information. State refresh and trigger messages, and the state-timeout timer are all employed as in the case of SS.

Soft-State With Reliable Trigger Messages (SS + RT): SS + RT is similar to SS with two important additions. First, trigger messages are transmitted *reliably* in SS + RT. Each time a trigger message is transmitted, the sender starts a *retransmission timer* (with value R). On receiving an explicit trigger message, the receiver not only updates signaling state, but also sends an acknowledgment to the sender. If no trigger acknowledgment

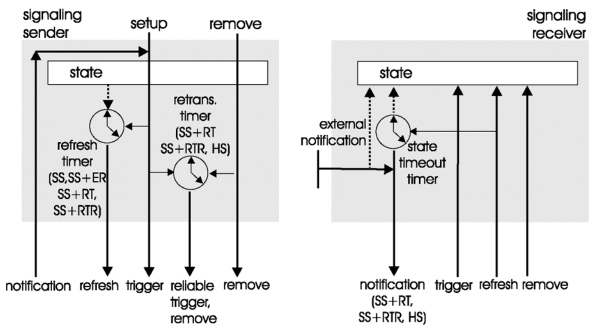

Fig. 1. Signaling sender and receiver: messages and mechanisms.

is received before the retransmission timer expires, the sender resends the trigger message. Second, SS + RT also employs a notification mechanism in which the signaling receiver informs the sender about state removals due to state-timeout timer expiration. This allows the sender to recover from false removal by sending a new trigger message.

Soft-State With Reliable Trigger/Removal Message (SS + RTR): SS + RTR is similar to the SS + RT approach, except that reliable messages are used not only for state setup/update but also for *state removal*.

Hard-State (HS) Approach: In the HS approach, reliable explicit messages are used to setup, update, and remove state at the receiver. Neither refresh messages nor soft-state timeout removal mechanisms are employed. A crucial concern with any hard-state protocol is the removal of orphaned state. Because the hard-state protocol will not remove state via timeout, it must rely on an external signal to detect that it is holding orphaned state. This signal can be generated, for example, by a separate heartbeat protocol whose job is to detect when the sender crashes and then inform the receiver of this event. Alternatively, the external signal might be generated via a notification from a lower layer protocol at the receiver that has an association with a lower layer protocol at the sender and, hence, can detect sender crashes. Once such an external notification (signal) is received, the hard-state signaling protocol cleans up the orphaned state associated with the signaling sender. One complicating factor is that of *false notification*, the external signal may falsely detect a sender crash (this would occur, for example, if a series of heartbeat messages were lost, but the sender was still operational). As in the case of SS + RT, false notification can be repaired by having the receiver notify the sender (if it exists) that its orphaned state has been removed. A signaling sender whose state has been incorrectly removed can then send a new trigger message.

Fig. 1 illustrates the messages and mechanisms used by the signaling sender and receiver in the various signaling protocols.

In Section III, we will develop a unified parameterized analytic model that allows us to quantify a key metric associated with a given signaling protocol—the fraction of time that the state of the state-installer and the state-holder are consistent (i.e., have the same value). Clearly, we would like this value to be as close to 1 as possible. In addition to quantifying consistency, we would also like to quantify a *cost* associated with a given signaling approach. One aspect of this cost is the signaling message rate itself. A second aspect of this cost is the cost associated with the sender and receiver having inconsistent state. For example, in IGMP, when an end host leaves without signaling its departure to its edge router, multicast data continues to flow towards the receiver (even though the receiving host is no longer in the multicast group), a cost. In the case of a hierarchical peer-to-peer file-sharing system in which a client uploads the names of the files it shares to a server when it joins the P2P network, but then leaves the network without signaling its departure, the inconsistent state at the server results in other peers attempting to contact the departed peer, again, a cost. In Section III, the cost function we adopt is a weighted sum of the signaling rate and application-specific costs (such as unwanted multicast data flows, or connection attempts to a departed peer in the previous examples).

Before delving into the details of the performance models, let us conclude this section with a qualitative discussion of the factors that influence performance.

- **Application-specific inconsistency cost.** As previously noted, this is a cost associated with the signaling sender and receiver being in an inconsistent state. The signaling sender may want to incur a higher signaling rate in order to keep this inconsistency cost low.
- **Refresh timeout value.** As noted in [1], the smaller the value of the refresh timer, the sooner consistent states will be installed at the state holder, and, consequently, the smaller the application-specific cost due to state inconsistency. However, this advantage comes at the cost of an increased signaling rate. This increased signaling cost may be warranted in order to reduce inconsistency cost.
- **Soft-state timeout value.** Since this timer is meant to remove state that is not refreshed, we would ideally like this value to be as small as possible in order to remove orphaned state as soon as the signaling sender departs. Too small a timeout value, however, can result in false state removal.
- **Signaling message loss.** As the probability of message loss increases, we expect the fraction of time that the signaling sender and receiver states are inconsistent to also increase, as it will take longer for either a message to be delivered reliably, or for a best-effort refresh message to be delivered. In cases of high loss and high application-specific inconsistency costs, protocols with explicit reliable transfer are expected to be preferable.
- **Number of hops.** In signaling protocols such as RSVP and AFSP [18], a signaling sender must install state at multiple nodes between itself and the ultimate signaling destination. As the number of hops increases, the fraction of time that all nodes are in an inconsistent state will also likely increase.

In the following sections, we will develop an analytic model that will allow us to quantitatively explore these issues.

III. Modeling and Analysis of Signaling Approaches in Single-Hop Systems

We begin our analysis by considering of a single node (the "signaling sender," henceforth the "sender") that can install, maintain, change, and eventually remove (or have removed) a single piece of state information at a remote node (the "signaling receiver," henceforth the "receiver"). We focus here on a single

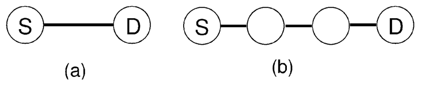

Fig. 2. Single-hop signaling systems. (a) Single physical hop. (b) Multiple physical hops with end-to-end signaling.

piece (rather than multiple pieces) of state, as it is conceptually simpler and the latter can generally be considered as multiple instantiations of the former. The installation, maintenance, change, and removal of state is accomplished using one of the five signaling protocols described in the Section II. We assume that the sender and receiver communicate over a network that can delay and lose, but not reorder, messages.

A. Signaling in a Single-Hop System

We, first, consider a single-hop system, in which the sender and receiver are the only two entities involved in the signaling protocol. As shown in Fig. 2, we can think of the two entities as being connected through a single *logical hop*, which may consist of one or more physical hops. A number of existing applications and protocols fit this simple single-hop model. For example, signaling in the IGMP protocol [4] occurs between an end system and its first-hop router. When the end system joins a multicast group, state indicating this group membership must be installed in the first-hop router; when the end host leaves the multicast group, this state should be removed from the router. In peer-to-peer file sharing applications, such as Kazaa [10], a peer registers its shared files with a server (a supernode in the case of Kazaa), which then redirects peers seeking a given file to peer nodes that have that file. A peer's registration of its files at a supernode is a single-hop signaling process, where the signaling sender is the peer, the signaling receiver is the supernode, and the signaling state contains the identities of the shared files and the fact that the peer is in the system and serving files.

B. Model Description

Before describing our system model, we first briefly discuss the events that can occur during the life cycle of a sender/receiver pair.

State setup. When the signaling session first installs (initializes) its local state, it transmits a signaling message containing the state to the receiver. After some delay, the message reaches the remote receiver, enabling both sender and receiver to achieve a consistent state.

State update. A sender may also update its local state. As in the case of state setup, the sender then installs the new state value at the receiver. When a sender updates its local state, the sender's and receiver's state will be inconsistent until the update successfully propagates to the receiver.

State removal. At the end of the life cycle, the sender will remove its state. At this point, the receiver's state should also be removed. Once the sender has removed its state, the receiver's state is "stale" (inconsistent) until it is removed. A number of protocol-dependent mechanisms (including state-timeout, and explicit removal messages) can be used to remove receiver state.

False state removal. The receiver may incorrectly remove state, even though the sender still maintains state. This can occur as a result of various protocol-dependent events. For example, in soft-state approaches, the state-timeout timer could expire at the receiver and remove state, even though the sender still maintains state.

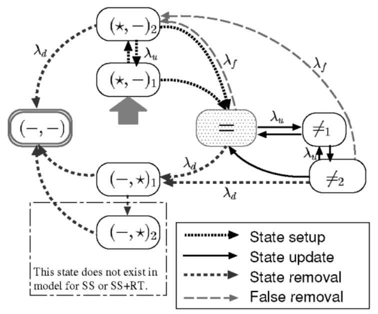

Fig. 3. Continuous time Markov model for signaling approaches in single-hop system.

The life cycle of a session is modeled by a transient Markov chain shown in Fig. 3. The Markov model's states are defined as follows. Each state consists of a pair of values (x_s, x_d), where x_s and x_d refer to the states of the sender and receiver, respectively.

- Model state $(\star, -)$ captures the initial stage of the life cycle, when a state has been installed at the sender but not at the receiver. This state is inconsistent, since the sender and receiver's state values do not match.
- Model states $(-, \star)$ correspond to cases where the sender has removed the state, but the receiver *has not*. These states are also inconsistent.
- When the sender and receiver have consistent state, the system is in state $=$.
- When the sender and the receiver have different state (i.e., both have installed state, but the state values are different), the system is in states $\neq$.
- When state is removed from both the sender and the receiver, the system enters an absorbing state represented by $(-, -)$.

Note that each of the inconsistent states, $(\star, -)$, $(-, \star)$, and $\neq$ are further divided into two separate Markov states distinguished by subscripts 1 and 2, the purpose of which is to capture protocol-dependent details that will be described shortly. In Fig. 3, a shaded arrow indicates the initial state of the Markov chain, and the double circled state $(-, -)$ is the absorbing Markov state.

The transitions among the Markov states are illustrated in Fig. 3 with different line styles indicating the different events (state setup, state update, state removal, and false removal) that cause state transitions. The system parameters affecting the state transitions are as follows:

λ_u	state update rate;
λ_d	$1/\lambda_d$ is the sender's mean state lifetime;
λ_f	false state removal rate at receiver;
D	average channel delay;
p_l	channel loss rate.

In addition, we have the following previously discussed protocol specific parameters:

TABLE I
MODEL TRANSITIONS

Transition rates	SS	SS+ER	SS+RT	SS+RTR	HS
$\lambda_{(\star,-)_1,(\star,-)_2}$ and $\lambda_{\neq_1,\neq_2}$	p_l/D	p_l/D	p_l/D	p_l/D	p_l/D
$\lambda_{(\star,-)_1,=}$ and $\lambda_{\neq_1,=}$	$(1-p_l)/D$	$(1-p_l)/D$	$(1-p_l)/D$	$(1-p_l)/D$	$(1-p_l)/D$
$\lambda_{(\star,-)_2,=}$ and $\lambda_{\neq_2,=}$	$(1-p_l)/T$	$(1-p_l)/T$	$(1/T+1/R)\cdot(1-p_l)$	$(1/T+1/R)\cdot(1-p_l)$	$(1-p_l)/R$
$\lambda_{(-,\star)_1,(-,\star)_2}$	$-$	p_l/D	$-$	p_l/D	p_l/D
$\lambda_{(-,\star)_1,(-,-)}$	$1/X$	$(1-p_l)/D$	$1/X$	$(1-p_l)/D$	$(1-p_l)/D$
$\lambda_{(-,\star)_2,(-,-)}$	$-$	$1/X$	$-$	$1/X+(1-p_l)/R$	$(1-p_l)/R$
λ_f	$p_l^{\lfloor X/T\rfloor}/X$	$p_l^{\lfloor X/T\rfloor}/X$	$p_l^{\lfloor X/T\rfloor}/X$	$p_l^{\lfloor X/T\rfloor}/X$	λ_w

T	average soft-state refresh timer value;
X	average soft-state state timeout timer value;
R	average message retransmission timer value for reliable transmission.

We assume that the signaling state lifetime and the interval between signaling state updates are independent exponentially distributed random variables (with means $1/\lambda_d$ and $1/\lambda_u$, respectively), false removal is a Poisson process with rate λ_f, and message losses are independent Bernoulli trials with parameter p_l. Furthermore, we approximate the soft-state refresh intervals, state-timeout intervals, message retransmission intervals and channel delays as exponentially distributed random variables with means T, X, R, and D, respectively.

In Section II, we discussed five different signaling protocols. Each of these protocols can be modeled using the model shown in Fig. 3, with different transition rates (and, in some cases, disabled transitions) for each of the protocols. We next describe the model transitions for each of these different protocols. These transitions are shown either in the model diagram or in Table I.

SS model. The initial state of the model, $(\star,-)_1$, corresponds to the creation of new signaling state at the sender. As discussed earlier, this results in a trigger message being sent to install state at the receiver. After a channel delay, one of two events can occur. First, the trigger message can successfully reach the receiver. This event occurs with probability $(1-p_l)$, and is modeled by the transition from state $(\star,-)_1$ to state $=$ with rate $(1-p_l)/D$. The second possibility is that the trigger message is lost. This event occurs with probability p_l, and is represented by the transition from $(\star,-)_1$ to $(\star,-)_2$ with rate p_l/D. Eventually a refresh message will reach the receiver. Since the mean time between refreshes is T, and each message reaches the receiver with probability $(1-p_l)$, there is a transition from $(\star,-)_2$ to state $=$ with rate $(1-p_l)/T$.

The update process is similar to the state installation process. When the state is consistent, i.e., the Markov chain is in state $=$, a state update causes the Markov chain to transit from $=$ to state $\neq_1$ at rate λ_u. The trigger message successfully arrives at the receiver with probability $(1-p_l)$ and delay D, which corresponds to a transition to $=$ at rate $(1-p_l)/D$. While in $\neq_1$, the loss of the trigger message causes the Markov chain to transit to state $\neq_2$ at rate p_l/D_l. With rate $(1-p_l)/T$, the Markov chain transits from state $\neq_2$ to state $=$. Note that an update may also occur when the system is in state $(\star,-)_2$ or state $\neq_2$, which causes the Markov chain to transit to state $(\star,-)_1$ or state $\neq_1$, respectively, with rate λ_u. Our model serializes events in the signaling process. For example, it does not allow a state update to occur while a trigger message is on its way to the receiver. We assume that an update occurs either before a previous trigger message is sent out or after the trigger message has already reached the receiver (or has been lost).

Sender state is removed at rate λ_d, i.e., a sender has a session of mean length $1/\lambda_d$. If the state is removed at the sender before the receiver has obtained the state, the Markov chain simply transits from $(\star,-)_2$ to the absorption state $(-,-)$. However, if the receiver has already installed state either consistently or inconsistently, i.e., the system in state $\neq_2$ or state $=$, the Markov chain transits to state $(-,\star)_1$. Thereafter, the receiver must wait for the state-timeout timer to expire in order for orphaned state to be removed. We model this by letting the Markov chain transit from state $(-,\star)_1$ to state $(-,-)$ with rate $1/X$. Note that the Markov model for SS does not include the $(-,\star)_2$ state in Fig. 3.

Finally, state can be removed at the destination due to the lack of receipt of refresh messages before the state-timeout timer expires. This is modeled by a Markov chain transition from states $=,\neq_2$, to state $(\star,-)_2$ with rate λ_f. Since such false removal only occurs when all refresh messages within a state timeout timer duration have been lost, we approximate the probability of this event as $p_l^{\lfloor X/T\rfloor}$. Therefore, λ_f can be expressed as $\lambda_f = \frac{1}{X}p_l^{\lfloor X/T\rfloor}$. Note that the model does not allow a state transition from $\neq_1$ to $(\star,-)_1$, due to the serialization considerations previously noted.

SS + ER model: Recall that in SS + ER, a message carries explicit state removal information to remove state. As in SS, state can also be removed by timeout. We model this explicit removal by modifying the state removal process in the SS model as follows. When the system is in state $(-,\star)_1$ as a result of sender state removal, an explicit state removal message is sent out. With probability $(1-p_l)$ and after a channel delay, this message arrives at the destination and triggers the removal of the corresponding state. We model this through a transition from $(-,\star)_1$ to the absorbing state $(-,-)$ with rate $(1-p_l)/D$. The loss of the explicit removal message causes the system to transit from $(-,\star)_1$ to $(-,\star)_2$. From there, the system transits to the absorbing state $(-,-)$ at rate $1/X$, capturing the event that state is removed by state-timeout timer expiration.

SS + RT Model: The Markov model for SS + RT differs from that of SS in that, when a trigger message carrying state setup/update information is lost, either a successful refresh message or a successful retransmission of the trigger message can bring the system from state $\neq_2$ or state $(\star,-)_2$ to state $=$ with rate $(1/T+1/R)\cdot(1-p_l)$.

SS + RTR Model: The Markov model for SS + RTR differs from that of SS + RT in that, when an explicit removal message is lost, a state-timeout timer expiration or a successful retransmission of the removal message is needed for the system to enter state $(-,-)$. Thus, the transition rate from state $(-,\star)_2$ to state $(-,-)$ is $1/X + (1-p_l)/R$.

HS Model: The HS model is similar to the SS + RTR model, except that the transition rates associated with refresh messages and state-timeout timers are excluded. In addition, as discussed in Section II, the HS approach must rely on an external signal to recover from sender failure. Accounting for the related cost of such an external signal is difficult, since it depends on the underlying scheme performing the failure detection for the HS approach. For instance, a link-layer sensing mechanism provides failure detection to HS signaling without introducing extra signals, whereas failure-detection relying on an underlying "heartbeat" mechanism may have an *overall* overhead comparable to that of SS + RTR. Nonetheless, we consider failure detection as a separate component and exclude it from the analysis. However, we assume that an external signal can be falsely generated with rate λ_w, resulting in the *faulty removal* of state in the HS approach.

We summarize the protocol-specific state transitions of the Markov chain for different signaling approaches in Table I, where λ_{ij} denotes the state transition rate from state i to j.

C. Model Solution and Performance Calculations

Using this model, we can now study the performance of the signaling approaches discussed in Section II. We are interested in the following metrics: the *inconsistency ratio*, δ, defined as the fraction of time that the signaling sender and receiver do not have identical state values; and the normalized average signaling message rate, γ^*, defined as $\gamma^* = \lambda_d \Gamma$, where Γ is the total number of signaling messages required during the lifetime of a signaling session (i.e., time from when the signaling state is initiated until it is removed from the system), and $1/\lambda_d$ is the expected lifetime of the sender's signaling session. Since the lifetime of the signaling session at the receiver varies with the signaling approach while $1/\lambda_d$ is invariant, the normalization provides a fair comparison between different signaling approaches.

To obtain the inconsistency ratio δ, we need to know the fraction of time that the system spends in states other than state $(=)$, before it eventually transits to the absorbing state $(-,-)$. Let π_i be the stationary probability of the system being in state i in the transient Markov model. This can be derived from the average time that the system spends on a particular state, as $\pi_i = \frac{w_{*,i}}{w_*}$, where w_* is the mean time to absorption from starting state $*$ and $w_{*,i}$ is the mean sojourn time in state i before absorption, when system starts in starting state $*$.[1]

Thus, we have the following expression for δ:

$$\delta = \pi_{(\star,-)_1} + \pi_{(\star,-)_2} + \pi_{\neq_1} + \pi_{\neq_2} + \pi_{(-,\star)_1} + \pi_{(-,\star)_2}.$$

[1]Given a transient Markov model with state space S, absorption state $a \in S$, state transition rate $\lambda_{ij}, \forall i,j \in S$, the mean time to absorption can be computed through $w_i = \begin{cases} \frac{1}{\lambda_i} + \sum_k \frac{\lambda_{ik}}{\lambda_i} w_k, & i \neq a \\ 0, & i = a \end{cases}$ and w_{ij} can be evaluated by $w_{ij} = \begin{cases} 1(i=j)\frac{1}{\lambda_i} + \sum_k \frac{\lambda_{ik}}{\lambda_i} w_{kj}, & i \neq a \\ 0, & i = a \end{cases}$, where $\lambda_i = \sum_j \lambda_{ij}$ and $1(\cdot)$ equals to one when the predicate $(\cdot)$ is true, and is zero, otherwise.

An alternative approach to compute π_i in our model is by merging the absorption state $(-,-)$ and the starting state $(\star,-)_1$, and computing the stationary probability of each state.

To obtain the total signaling message overhead, Γ, we need to compute the average lifetime of a signaling state, $\mathcal{T}$, and the mean signaling message rate γ

$$\Gamma = \mathcal{T} \cdot \gamma. \tag{1}$$

Here, $\mathcal{T}$ is derived from calculating the mean time to absorption for state $(\star,-)_1$ in the transient Markov model, and γ is obtained by considering in which state and with what rate each of the messages, explicit trigger and removal messages, soft-state refresh messages, retransmission and acknowledgment messages, are generated during the signaling process. We proceed as follows.

With a successfully transmitted trigger message, the system transits from state $(\star,-)_1$ or $\neq_1$ to state $=$, and if a trigger message is lost, the Markov chain transits from state $(\star,-)_1$ to $(\star,-)_2$ or from $\neq_1$ to $\neq_2$. Thus, the message rate for explicit triggers, γ_{et}, is

$$\gamma_{et} = \pi_{(\star,-)_1}\lambda_{(\star,-)_1,=} + \pi_{\neq_1}\lambda_{\neq_1,=} + \pi_{(\star,-)_1}\lambda_{(\star,-)_1,(\star,-)_2} + \pi_{\neq_1}\lambda_{\neq_1,\neq_2}. \tag{2}$$

Similarly, the message rate for explicit removal, γ_{er}, is

$$\gamma_{\mathrm{er}} = \pi_{(-,\star)_1}\lambda_{(-,\star)_1,(-,-)} + \pi_{(-,\star)_1}\lambda_{(-,\star)_1,(-,\star)_2}. \tag{3}$$

Soft-state refresh messages are generated at mean rate $1/T$ when the Markov chain is in states $(\star,-)_2, =$, or $\neq_2$. Therefore, the mean message rate for refresh messages, γ_r, can be expressed as

$$\gamma_r = \frac{1}{T} \cdot \left(\pi_{(\star,-)_2} + \pi_{=} + \pi_{\neq_2}\right). \tag{4}$$

If trigger messages are transmitted reliably, retransmissions will be generated at rate $1/R$ when the chain is in states $(\star,-)_2$ and $\neq_2$, and acknowledgment messages will be generated for every transition to state $=$. Therefore, the message rate for reliable triggers, γ_{rt} is

$$\gamma_{\mathrm{rt}} = \frac{1}{R}\left(\pi_{(\star,-)_2} + \pi_{\neq_2}\right) + \sum_i \pi_i \lambda_{i,=} + \lambda_f(\pi_{=} + \pi_{\neq_2}). \tag{5}$$

The third term of γ_{rt} is caused by false removal, since a reliable trigger scheme requires the signaling destination to send a message to the signaling sender notifying it of the removal. Similarly, for reliable removal, the message rate γ_{rr} is

$$\gamma_{\mathrm{rr}} = \frac{1}{R}\pi_{(-,\star)_2} + \pi_{(-,\star)_1}\lambda_{(-,\star)_1,(-,-)} + \pi_{(-,\star)_2}\lambda_{(-,\star)_2,(-,-)}. \tag{6}$$

In summary, the overall message rate for the different signaling protocols are

$$\begin{aligned} \text{SS:}\gamma &= \gamma_{\mathrm{et}} + \gamma_r \\ \text{SS+ER:}\gamma &= \gamma_{\mathrm{et}} + \gamma_r + \gamma_{\mathrm{er}} \\ \text{SS+RT}: \gamma &= \gamma_{\mathrm{et}} + \gamma_r + \gamma_{\mathrm{rt}} \\ \text{SS+RTR:}\gamma &= \gamma_{\mathrm{et}} + \gamma_r + \gamma_{\mathrm{er}} + \gamma_{\mathrm{rt}} + \gamma_{\mathrm{rr}} \\ \text{HS:}\gamma &= \gamma_{\mathrm{et}} + \gamma_{\mathrm{rt}} + \gamma_{\mathrm{rr}} + \gamma_{\mathrm{er}}. \end{aligned}$$

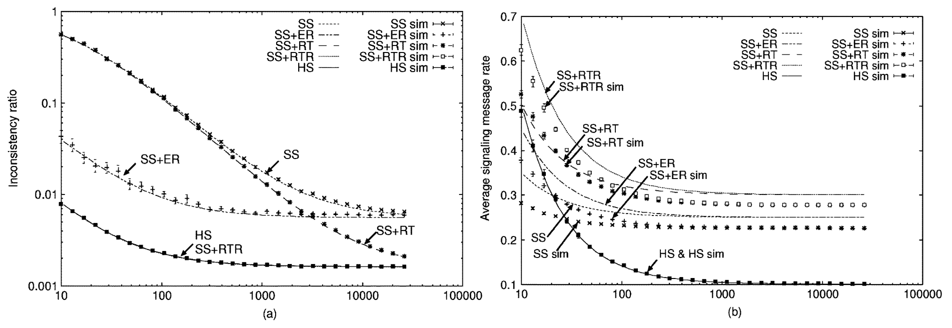

Fig. 4. Comparison against the time that a signaling state exists at the sender (i.e., session length), $1/\lambda_d$. (a) Mean lifetime of a signaling state at signaling sender (seconds): $1/\lambda_d$. (b) Mean lifetime of a signaling state at signaling sender (seconds): $1/\lambda_d$.

D. *Model Evaluation*

We now compare and contrast the performance of the five different signaling approaches using our modeling framework. Unless otherwise noted, we use the following default parameters: $p_l = 0.02$, $D = 30$ ms, $1/\lambda_u = 20$ s, $1/\lambda_d = 1800$ s, $T = 5$ s, $X = 3T$, $R = 4D$, and $\lambda_w = 0.0001$. These parameter values are chosen to capture the behavior of a Kazaa session: state is added when a Kazaa regular peer (hereafter, simply referred to as a peer) starts the Kazaa application, and is updated when the peer changes its collection of shared files (e.g., a new file is downloaded into its shared directory). When the peer exits the Kazaa application, the state maintained by the supernode (as described in the beginning of Section II) for that peer should be deleted.

Impact of Session Length $(1/\lambda_d)$*:* We first study the performance of different protocols as a function of the expected amount of time that state is installed at the sender, $(1/\lambda_d)$. In our Kazaa example, this corresponds to a peer's average session length. In Fig. 4(a), we plot the inconsistency ratio δ, and in Fig. 4(b), we plot the normalized average signaling message rate γ^* for the different signaling approaches. Since our model assumes exponentially distributed timer values, while in practice signaling protocols usually use deterministic timers, we simulate the system with deterministic timers and show the corresponding simulation results in Fig. 4 as dotted curves with 95% confidence intervals. We find a good match between the analytical and the simulation results with deterministic timers, indicating that the impact of assuming exponential timer values in our model is limited.

Fig. 4 provides the following number of insights into the single-hop signaling system.

- When the expected session length increases, both the inconsistency ratio and the average signaling message rate decrease for all signaling approaches. In the context of our Kazaa example, this implies that if Kazaa is used mostly by peers who tend to turn themselves off shortly after starting, as opposed to remaining on for long periods, (e.g., peers use Kazaa for 5 min every hour versus 2 hr every day), the system is likely to incur greater signaling overhead, with supernodes responding to queries based on stale information.
- Comparing SS + ER to SS, we note that the improvement of SS + ER over SS (using the inconsistency ratio as the performance metric) becomes more significant as the average session length decreases. Even when the average session length is on the order of thousands of seconds, the benefit of adding explicit removal is still non-negligible. This is due to the fact that removing orphaned state requires a relatively long wait for the timeout timer to expire, in the absence of explicit removal. More importantly, considering the average message rate in Fig. 4, we find that when the average session length is on the order of thousands of seconds, the addition of explicit removal introduces negligible signaling message overhead compared to the SS approach. While the cost of including this capability is so low, our model indicates that it is very useful to include explicit removal in SS signaling in such circumstances, since the "penalty" of not using explicit removal is so high.
- Fig. 4(a) indicates that the performance gain (in terms of a reduced inconsistency ratio) achieved by introducing reliable triggers becomes significant when the peers' average session length is long. This is evidenced by the fact that when the average session length is long, the five approaches are differentiated according to whether or not they provide reliable triggers. Conversely, when the average session length is short [towards the left of Fig. 4(a)] the five approaches are grouped on the basis of how state removal is performed: those without explicit removal (SS, SS + RT) have the highest inconsistency ratio, those with explicit removal (SS + ER) have a reduced inconsistency ratio, and those with reliable removal (SS + RTR, HS) have the lowest inconsistency ratio. We note that for long sessions, when the differences in trigger message reliability is most pronounced, the inconsistency ratio is relatively low for all approaches. Also, as shown in Fig. 4(b), the limited benefit of SS + RT over SS comes with nontrivial additional signaling overhead. Thus, for these application parameters, providing reliable trigger messages does not appear to be crucial.
- SS + RTR provides essentially the same inconsistency ratio as HS. This suggests that beyond explicit removal and reliable transmission, any enhancements to SS refresh mechanism can provide only modest gains (if any) in the inconsistency ratio. Indeed, in some cases SS + RTR already performs slightly better than HS.
- Comparing HS and SS approaches, HS has relatively low inconsistency ratio and message overhead once the mean

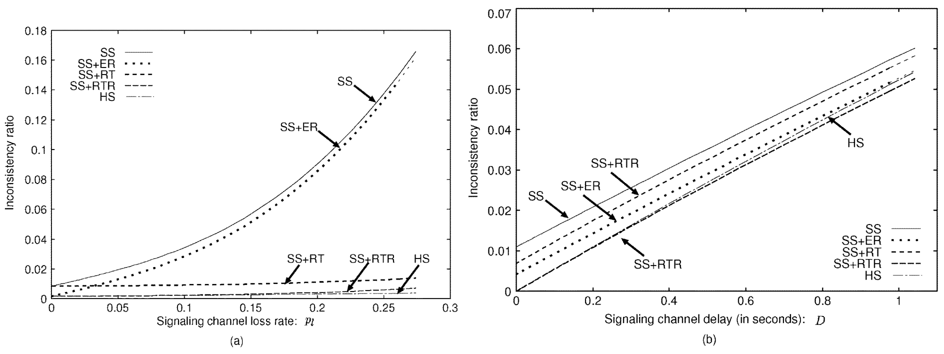

Fig. 5. Comparison against link loss rate p_l and link delay D.

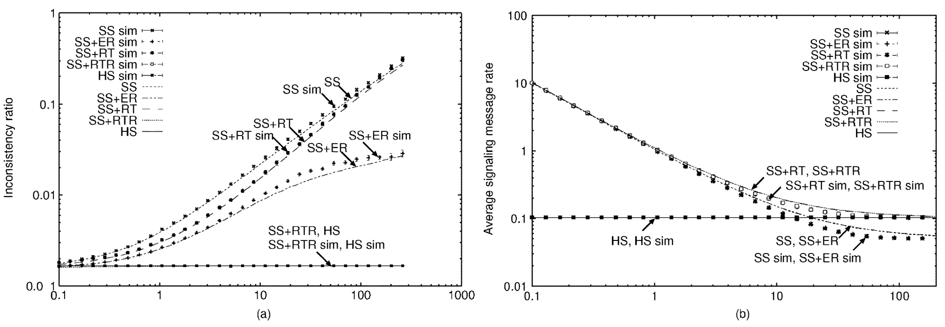

Fig. 6. Comparison against SS refresh timer (T), single hop case. (a) SS refresh timer (in seconds): T. (b) SS refresh timer (in seconds): T.

lifetime of signaling state is past a threshold value. The lower message overhead of HS is due to the absence of periodical refresh messages as in SS approaches. Such advantage of HS is more pronounced when the mean state lifetime is longer. In addition, the lack of refresh mechanism in HS avoids the accidental removal phenomenon of SS approaches. However, for the same reason, HS has to rely on an external notification system for detecting orphaned states. Thus, the performance of HS depends on the accuracy of the external notification system. In Fig. 4, we have assumed a low "false alarm" rate ($\lambda_w = 0.0001$), which contributes to the relatively good performance of HS.

Note that we have assumed exponentially distributed session lengths in Fig. 4. We have also explored other distribution functions, including lognormal distribution and Pareto distribution, for the lifetime of signaling states in our simulation. The results are quantitatively similar to those with exponential distributed state lifetimes.

Impact of Message Loss and Delay: Fig. 5(a) and (b) plot the inconsistency ratios of different signaling approaches for various loss rates and delays. Fig. 5(a) indicates that even for modest loss rates (e.g., 0.05), reliable transmission significantly improves the performance of SS protocols. Fig. 5(b) plots the inconsistency ratio versus the one-way sender-to-receiver delay. We observe an approximately linear increase in the inconsistency ratio as a function of the one-way channel delay for all signaling protocols. However, signaling protocols with reliable transmission exhibit a slightly larger gradient. This is because the value of the retransmission timer is generally proportional to the channel delay. Thus, to recover from loss, approaches with reliable transmission suffer longer latencies in an environment with longer transmission delays, while SS approaches relying only on refresh mechanisms do not.

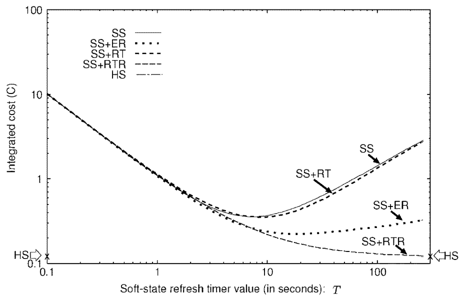

Fig. 7. SS refresh timer (T).

Impact of Timer Configuration: There are three different timers used in the five signaling approaches we consider: the SS refresh timer, the SS state-timeout timer, and the retransmission timer. Fig. 6 explores the performance of different SS signaling approaches under different SS refresh timer settings. When the refresh timer value changes, we set the state-timeout timer to be three times the value of the refresh timer. Fig. 6 reveals an interesting tradeoff between having a short refresh timer (to reduce the inconsistency ratio) and a long refresh timer (to maintain a low message overhead). Note that Fig. 6 also includes the simulation results where the refresh timer, the state-timeout timer and the retransmission timer are deterministic (as opposed to exponentially distributed timers in the analytical model). We observe only a small performance difference (less than 3%) between the use of deterministic timers and exponential timers.

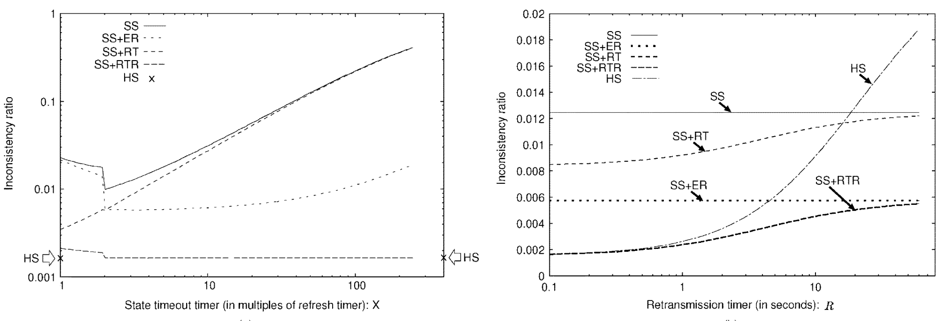

Fig. 8. State timeout timer (X) for SS-based protocols and retransmission timer R for reliable transmissions.

Overall Cost: As discussed earlier, the overall cost is the sum of two components: message cost and application-specific costs arising from inconsistent state in the sender and receiver. For example, we saw earlier that for IGMP, this latter cost is due to the transmission of unwanted multicast data; in the case of Kazaa, this latter cost is the additional overhead caused by the supernode providing peers with pointers to already departed peers. To evaluate the cost of both signaling overhead and application-specific costs resulting from state inconsistency, we define an *integrated cost* (C) as

$$C = c \cdot \delta + \gamma^* \tag{7}$$

where c indicates the relative weight of application-specific cost due to inconsistent signaling state. In Kazaa, for example, c might be interpreted as the number of signaling messages associated with fruitless queries that are caused by inconsistent file-sharing state at the supernode. In other application scenarios, different cost functions may be better suited for integrating the two cost components. Here, we choose the weighted linear function for the benefit of simplicity. In the following, we set c to be 10 (msg/s).

In Fig. 7, we plot the integrated cost associated with different signaling protocols versus the mean SS refresh timer value, (T). From this figure, we observe that there exist relatively sensitive optimal operating points for SS and SS + RT, above which the inconsistency cost increases substantially and below which the message signaling cost increases significantly. Such an optimal operating point also exists for SS + ER, although the integrated cost is not very sensitive to large refresh timer values. Last, for SS + RTR, a large timer value is preferred, and when the timer is large enough (on the order of hundreds of seconds), it provides comparable performance to the HS approach.

Fig. 8(a) explores the impact of the average state timeout timer value on the inconsistency ratio of SS approaches. Here, we fix the mean state refresh timer to be 5 s and vary the mean state-timeout timer. The x-axis is in the units of multiples of state refresh timer value. We observe that different approaches behave very differently: SS + RTR does well with large timeout values, since the larger the timeout timer, the less likely it is that a state is falsely removed due to loss of refresh messages. SS and SS + ER do best when the state-timeout timer is approximately twice the size of the refresh timer, so that the probability of false removal is reduced, in another word, the system is resilient to random loss of refresh message. However, since larger timeout timers add larger delays to remove orphaned state, SS and SS + ER also require the state timeout timer to be small enough to avoid such problems. Recall that SS + RT employs a notification mechanism in which the signaling receiver informs the signaling sender about state removals and the signaling sender recovers from a false removal by sending another trigger message. Since SS + RT is the most sensitive to the process of removing orphaned state and its notification mechanism reduces the penalty of false removal, it works best with a timeout timer value that is just slightly larger than that of the state-refresh timer.

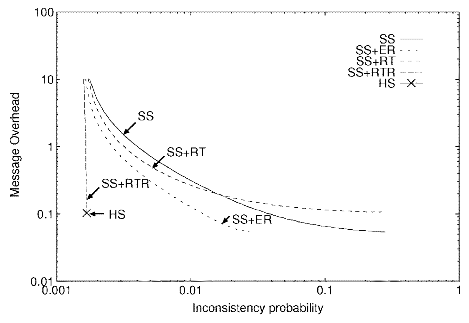

Fig. 9. Tradeoff between inconsistency ratio and average signaling message rate, derived by varying T.

Fig. 8(b) explores the impact of average retransmission timer value on the inconsistency ratio of the five signaling approaches. Here, we set the state-timeout timer as 15 s, which is three times of the refresh timer. Since HS depends only on explicit reliable transmissions for state setup/update/removal, it is the most sensitive to changes in the retransmission timer R.

Tradeoff Between Inconsistency Ratio and Signaling Message Rate: By varying the SS refresh timer, we study the tradeoff between the inconsistency ratio and the signaling message rate of different signaling protocols, and show the results in Fig. 9. Since HS does not use the refresh timer, neither the inconsistency ratio nor the average signaling message rate vary with T; in Fig. 9, HS is shown as a single point $\times$. Fig. 9 also indicates that the inconsistency of SS + RTR is not sensitive to SS refresh rate (which is determined by the refresh timer), whereas the inconsistency of the other SS protocols change with the signaling overhead. We also examined the tradeoffs between inconsistency ratio and signaling overhead based on other system or design parameters. Fig. 10 shows the tradeoff between the inconsistency ratio and the message overhead by varying the

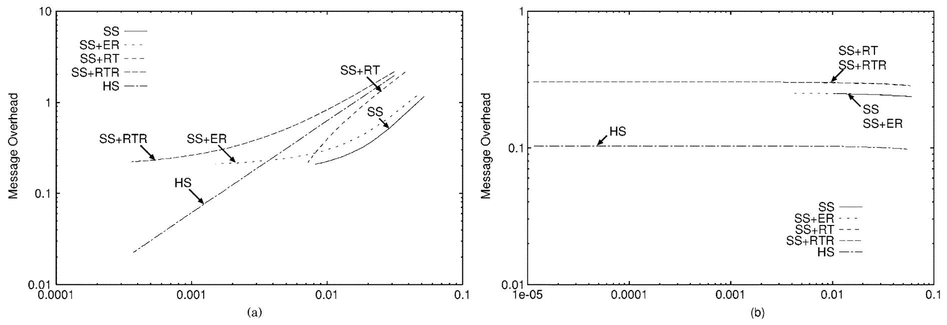

Fig. 10. Tradeoff between inconsistency ratio and average signaling message rate: (a) derived by varying $1/\lambda_u$ and (b) derived by varying D.

$$\begin{tikzcd} S \arrow[r, dash] & D' \arrow[r, dash] & D' \arrow[r, dash] & D' \arrow[r, dash] & D \end{tikzcd}$$

Fig. 11. Sample of a multihop system.

$$\begin{tikzcd} S \arrow[r, dash, "="] & D' \arrow[r, dash, "="] & D' \arrow[r, dash, "="] & D' \arrow[r, dash, "\neq"] & D' \arrow[r, dash, "\neq"] & D \end{tikzcd}$$

Fig. 12. Five-hops system with three consistent hops.

signaling state update rate λ_u Fig. 10(a) and by varying the average signaling channel delay D Fig. 10(b). From Fig. 10(a), we observe that for a large inconsistency probability (> 0.01), to achieve the same consistency quality, SS uses the least amount of signaling messages compared to all other signaling protocols. On the other hand, to achieve small inconsistency probabilities (< 0.004), HS should be used to reduce signaling overhead. From Fig. 10(b), we observe that the tradeoff curves are not sensitive to the mean signaling channel delay.

IV. Modeling and Analysis of Signaling Approaches in Multihop Systems

In this section, we consider the case in which not only the end systems (the signaling sender and receiver) but also intermediate routers or relay nodes (henceforth referred as nodes) must maintain signaling state. This would occur, for example, in the case that the intermediate nodes need to maintain a bandwidth reservation for communication between the end systems. Fig. 11 illustrates the abstract model of a multihop signaling system that involving a signaling sender, a receiver, and a chain of intermediate signaling nodes. State generated at the sender is required to be maintained at the receiver and at all nodes along the path between sender and receiver.

A. *Model for Multihop Systems*

In Section III, we identified various factors that influence the performance of signaling protocols in the single-hop scenario. Many of the results are directly applicable here as well. In this section, we focus on the unique issues that arise in the case that state is maintained at a chain of multiple hops. We focus on the process in a multihop signaling system, in which state is updated at the signaling sender, and changes must be propagated to all receivers. To simplify our model, we consider the lifetime of a state to be infinity ($\lambda_d \to \infty$), and model state updates as a Poisson process with rate λ_u.

Our models for the multihop system are extensions of the single-hop models. Let N denote the number of hops in a multihop system and n ($n \leq N$) the number of consistent hops (links). We assume that these hops are identical, i.e., they have identical channel loss rate (p_l) and mean channel delay (D), and further assume that channel losses are independent. We define a *consistent hop* to be a hop (link) whose two end points have consistent state. Fig. 12 illustrates a five-hop ($N = 5$) system with three consistent hops ($n = 3$).

We model state update as a Markov process with the state space as $\mathcal{S} = \{(n, s)\}$, where $0 \leq n \leq N$ is the number of consistent hops, and s is a variable taking on values of 0 or 1 to indicate whether the Markov chain is in a *fast-path* state ($s = 0$) or a *slow-path* ($s = 1$) state. The distinction between a *fast-path* Markov state and a *slow-path* Markov state will be made clear later. In addition to the (n, s) states, there is a special state for HS signaling, Δ, that models the interval during which the HS system recovers from a false removal.

We, next, describe the model transitions for the pure soft-state protocol (SS), the soft-state protocol enhanced with hop-by-hop reliable transmission (SS+RT), and the hard-state protocol (HS).

Modeling SS Protocol Transitions: Under SS, state updates are carried by trigger messages sent to the receiver(s). A trigger message may be lost at any of the hops. If a trigger message is lost, a refresh message carrying identical information will eventually reach the signaling receiver(s) and make the receiver(s) state consistent. A state is removed if the timeout timer expires due to losses of all refresh messages sent during the timeout interval. Note that if a timeout occurs at one node, all nodes beyond this node in the linear topology also time out, because they too will not receive the refresh messages. The behavior of this SS protocol is modeled by the Markov chain illustrated in Fig. 13.

Consider the state update process. When signaling state is updated at the sender, the system transits to state $(0, 0)$, which represents the case in which no hop is consistent ($n = 0$) and a trigger message is on its way to the next hop (i.e., the model is in the so-called *fast-path* state, $s = 0$).

After a one-hop channel delay, two things may happen to the trigger message. First, it may successfully reach the next receiver, making this hop consistent. Thus, the system transits from $(0, 0)$ to $(1, 0)$, or more generally from state $(i, 0)$ to state $(i + 1, 0)$, with rate $(1 - p_l)/D$. The second possibility is that the trigger is lost, in which case the model transitions from $(0, 0)$ to $(0, 1)$, or more generally from state $(i, 0)$ to state $(i, 1)$, with rate p_l/D and waits for a subsequent refresh message (i.e., the model is in a *slow-path* state, $s = 1$).

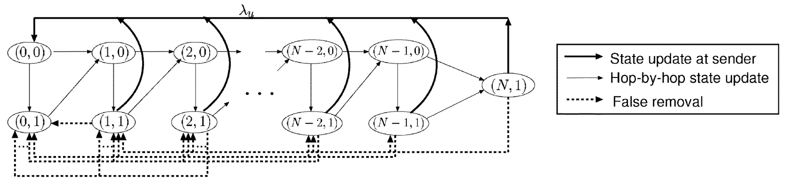

Fig. 13. Markov model for end-to-end SS approach.

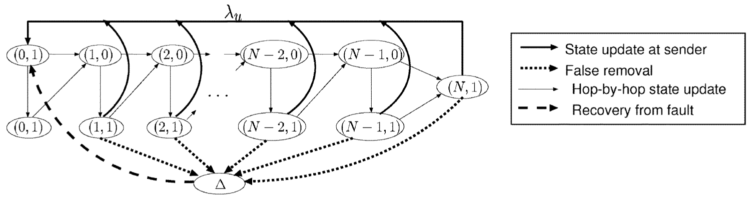

Fig. 14. Markov model for HS signaling approach.

Since we assume that refresh intervals are exponentially distributed with mean T, and the probability that a refresh message generated at the sender reaches across the ith hop is $(1-p_l)^i$, the transition rate from state $(i-1,1)$ to state $(i,0)$ is $(1-p_l)^i/T$. Eventually, the system can transit to state $(N,1)$, either via fast path state $(N-1,0)$ or via slow path state $(N-1,1)$.

In addition, the state-timeout timers at signaling destinations may expire due to lost refresh messages. Assume hop j is the first hop in the chain where a state timeout occurs, (in this case, the timer of the corresponding states at the $j+1$th to the Nth hops will also expire). When this happens, we let the Markov chain transit to state $(j,1)$ with rate $\lambda^{(e)}_{(i,1),(j,1)}$. Transition rate $\lambda^{(e)}_{(i,1),(j,1)}$ is given by

$$\lambda_{(i,1),(j,1)} = \begin{cases} \frac{1}{X}\cdot[1-(1-p_l)^{j+1}]^{\lfloor \frac{X}{T}\rfloor} \\ -\frac{1}{X}\cdot[1-(1-p_l)^{j}]^{\lfloor \frac{X}{T}\rfloor}, & 1\le j< i\le N \\ 0, & \text{otherwise.}\end{cases} \tag{8}$$

The expression $(1-(1-p_l)^{j+1})^{\lfloor \frac{X}{T}\rfloor} - (1-(1-p_l)^{j})^{\lfloor \frac{X}{T}\rfloor}$ approximates the probability that a timeout occurs at the $(j+1)$th signaling receiver, but not at any preceding hop.

Model Transitions for SS + RT Protocol: Under SS + RT, when the system is in a slow path state, both a successful retransmission of the trigger message and a successful refresh message can make the corresponding hop consistent. This is because reliable transmission is used. Therefore, in SS + RT, the transition rate from state $(i-1,1)$ to state $(i,0)$ becomes

$$\lambda_{(i-1,1),(i,0)} = \frac{1}{T}(1-p_l)^i + \frac{1}{R}(1-p_l). \tag{9}$$

Model Transitions for HS Protocol: In HS, reliable trigger messages (propagated reliably hop-by-hop) are used to update state along the signaling path. Neither refresh messages nor SS timeout removal are employed. Thus, a state transition from state $(i-1,1)$ to state $(i,0)$ is achieved via retransmission, and the transition rate is

$$\lambda_{(i-1,1),(i,0)} = \frac{1}{R}(1-p_l). \tag{10}$$

As in the case of the single-hop system, we model false removals at each receiver as an independent Poisson process with rate λ_w. Thus, the system transits from a slow path state to the recovery state, Δ, with rate $N\lambda_w$. As discussed in Section II, a receiver is notified by an external signal when any failure (e.g., link failure) occurs. This receiver then sends messages to inform other receivers and the sender of the failure. On receipt of such a message, other receivers remove their associated state(s). If the sender receives such a message, it sends a trigger signaling message to reinstall state. We model this by letting the system transit from the Δ state to the $(0,0)$ state with rate $\frac{2}{ND}$, an approximation that captures the expected latency for the sender to initiate the recovery process. We show the modified multihop Markov model for HS approach in Fig. 14.

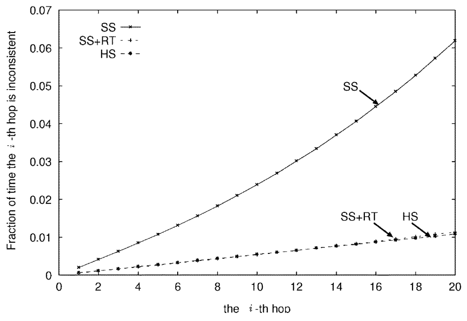

Fig. 15. Fraction of time that the ith hop is inconsistent, where $1\le i\le N$ and $N=20$.

B. Multihop Model Solution and Results

The solution of the multihop model is similar to that of the single-hop model. Let π_i be the stationary probability of the system being in state i. The inconsistency ratio δ for the multihop system is

$$\delta = 1-\pi_{N,1}. \tag{11}$$

The mean signaling message rate γ for the SS approach is

$$\gamma_{\text{ss}} = \sum_{k=1}^{N-1}\pi_{k,0}\cdot\frac{1}{D} + \sum_{k=0}^{N}\pi_{k,1}\cdot\frac{1}{T}\cdot\bar{E} \tag{12}$$

where $\bar{E}$ is the expected number of messages

$$\bar{E} = E[k\,|\,\text{loss}]\cdot P[\text{loss}\,|\,\text{msg}] + P[\text{succ}\,|\,\text{msg}]\cdot(k+1) \tag{13}$$

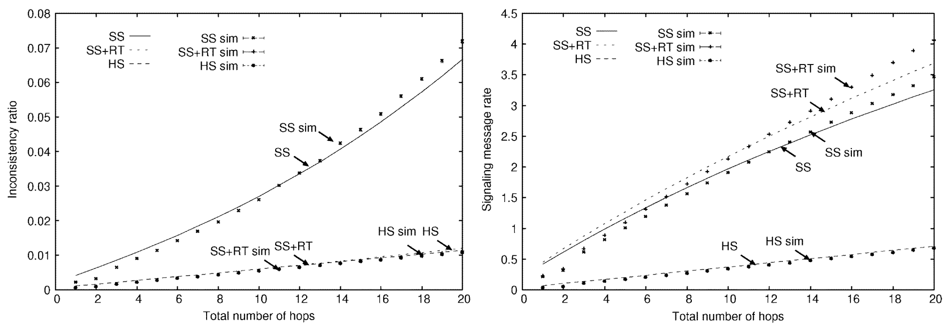

Fig. 16. Inconsistency ratio and signaling message rate against total number of hops.

and $E[k \mid \text{loss}]$ is the expected number of hops that a packet is lost before it arrives at the kth hop

$$\begin{aligned} E[k \mid \text{loss}] &= \sum_{i=1}^{k} i \cdot (1-p_l)^{i-1} \cdot p_l \\ &= -p_l \left(\frac{(1-p_l)(1-(1-p_l)^k)}{p_l} \right)' \\ &= \frac{1-(1-p_l)^k}{p_l} - k(1-p_l)^k. \end{aligned} \tag{14}$$

The mean signaling message rate for SS + RT approach is

$$\gamma_{\text{ss+rt}} = \sum_{k=1}^{N-1} \pi_{k,0} \cdot \frac{1}{D} + \sum_{k=0}^{N} \pi_{k,1} \cdot \left(\frac{1}{R} + \frac{1}{T}\bar{E} \right) \tag{15}$$

and the mean signaling message rate for HS is

$$\gamma_{\text{hs}} = \sum_{k=1}^{N-1} \pi_{k,0} \cdot \frac{1}{D} + \sum_{k=0}^{N} \pi_{k,1} \cdot \frac{1}{R} + \pi_{\Delta} \cdot \frac{2}{D}. \tag{16}$$

We examine the inconsistency ratio and signaling message overhead of the three multihop signaling approaches. In choosing model parameters, we consider the process of reserving bandwidth along a multihop path as an example. Unless otherwise specified, we use the following default parameters: $N = 20$, $p_l = 0.02$, and $D = 30$ ms at each hop, $1/\lambda_u = 60$ s, $T = 5$ s, $X = 3T$, $R = 4D$, and $\lambda_w = p_l^3$. We focus on the impact of multiple hops on performance.

In Fig. 15, we plot the fraction of time that the ith hop is inconsistent, where the total number of hops is 20. We observe that the inconsistency associated with a hop on the signaling path increases, with increasing distance from the signaling sender. The inconsistency increases in an approximately linear manner for all signaling approaches. Combining hop-by-hop reliable triggers with the end-to-end SS approach significantly improves consistency at all nodes, with the consistency of SS + RT being comparable to that of the HS approach. In our evaluation, the HS approach exhibits slightly higher consistency than SS + RT. This is due to the effect of a state being falsely removed upon the expiration of a state timeout timer in the SS + RT approach, and since the SS refresh messages are generated from the sender only, state timeout is more likely to happen at the receivers far (more hops away) from the sender.

Fig. 16 plots both the inconsistency rate (on the left) and the signaling message rate (on the right) as a function of the number of hops in the multihop system. In the same figure, we also plot the results derived from simulations with deterministic timers. The simulation results are shown as dotted curves with 95% confidence intervals. From Fig. 16, we observe that both inconsistency and signaling message overhead monotonically increase as the number of hops increase. From Fig. 16, comparing the HS approach and the SS approach with reliable trigger approach (SS + RT) indicates that the consistency of the pure SS approach is more sensitive to an increase in the number of hops. Fig. 16(b) suggests that adding a reliable trigger to the end-to-end SS approach, while significantly improving consistency, introduces relatively little additional signaling overhead.

We also evaluated the impact of system and design parameters on the performance of the multihop signaling approaches. Fig. 17 shows the impact of the mean SS refresh timer T on the performance of signaling approaches, SS, SS + RT and HS for a multihop system. From Fig. 17(a), we observe that when the SS refresh timer is relatively small (< 0.9 s), the inconsistency ratio of SS decreases as T increases; however, when the SS refresh timer increases (> 0.9 s), the inconsistency ratio of SS significantly increases as T increases. On the other hand, the inconsistency ratio of SS + RT decreases as the refresh timer value increases, until it reaches an optimal value at the point of $T = 10$ s. From Fig. 17(b), we observe that the average signaling message rate for both SS and SS + RT decreases as the SS refresh timer increases.

V. Related Work

The most closely related work to our present work is [15], the first effort to develop analytic models of SS protocols, and also the first (and only) that has sought to develop a more principled understanding of SS protocols. The model in [15] considered link loss and a state deletion probability, and introduced the metric of inconsistency that we have adopted here. There are a number of important differences in our works. The two protocols considered in [15] correspond closely to our SS and SS + RT protocols. A single server queuing network is used to model the *system consistency* for a single-hop SS signaling system. However, there are several counter intuitive observations that result from their model. Based on their model, the length of time that a signaling state is maintained in the system depends on the rate of new states being generated and the service capacity of the signaling channel (as a concrete example, this would correspond to a session's holding time being dependent on the bandwidth of the signaling channel, an assumption that is unrealistic). In our study, we assume that the time that a signaling state exists in the system is application dependent and independent of signaling channel bandwidth. In addition, due to a mistake in evaluating the average system consistency $E[c(t)]$

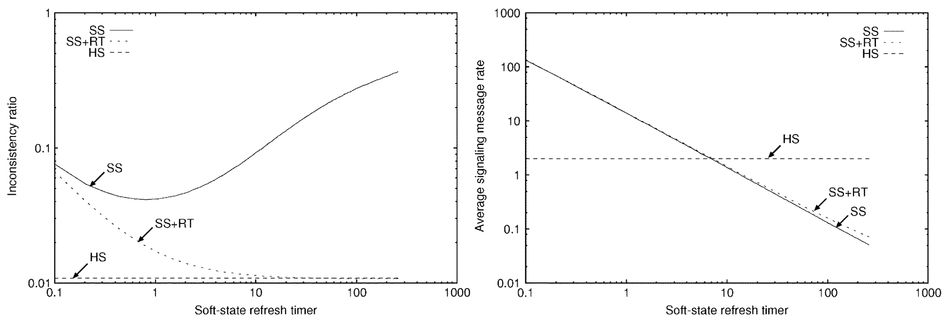

Fig. 17. Comparison against SS refresh timer T, multihop case.

in [15, Sec. 3] (where $P(n_I, n_C)$ should be replaced by conditional probability $P(n_I, n_C \mid n_I + n_C > 0)$), the results in [15] indicated that as the capacity of the signaling channel increases, the degree of inconsistency also *increases*. Our model, by creating an absorbing state, avoids such a rather counter intuitive result. More importantly, while [15] considers only state setup, we consider both state installation, state updates, and state removal as integral parts of a spectrum of signaling protocols.

In [15], the authors proposed a modification for the SS signaling protocol, where a NACK message is sent by the receiver when a signaling message is detected to be lost, and assume that when message loss happens, the receiver learns of this loss instantaneously. A priority queue is used at the sender to transmit signaling messages, where messages corresponding to new states and messages that are detected to be lost are sent with a higher priority. This protocol is essentially the SS + RT signaling protocol that we propose in our work. However, in our work, retransmissions of lost messages are controlled by a separate state retransmission timer (R), while in Raman and McCane's work, such retransmission timer depends on channel capacity and other parameters. In addition, Raman and McCane use simulation to evaluate the parameters that affect the consistency performance of their modified SS approach, while in our study, our generic model can be used to analyze the performance of both SS and SS + RT signaling systems.

Other than SS and SS + RT signaling protocols, in our work, we also consider a broader range of protocols, including those that adopt a number of HS features (including the SS + ER and HS protocols). We built a generic Markovian model to analyze the performance of all the signaling systems that we have identified. More generally our aim is not just to understand SS protocols but to compare and contrast a variety of signaling protocols and their mechanisms, ranging from a "pure" SS, to SS approaches augmented with explicit state removal and/or reliable signaling, to a "pure" HS protocol.

In a recent work presented by Lui *et al.* [11], analytic models have been proposed to evaluate the *robustness* of SS and HS protocols under three "extreme" network scenarios, denial of service attacks, correlated lossy feedback channel, and implosion under multicast services. While generally refering to "robustness" as the condition that *performance is above some acceptable threshold and need not be optimal*, in various network scenarios, the authors use different analytic models to evaluate different system metrics as measures of robustness. However, one of the most importnat performance metrics, the *state consistency*, of signaling protocols is not considered in their study.

Two works that have addressed limited aspects of SS protocol operation include [16], which investigated techniques to dynamically set SS timer values, and [12], which investigated a scheme to use different SS timers for trigger and refresh messages.

VI. Conclusion

In this paper, we have compared and contrasted the performance of a variety of signaling approaches ranging from a "pure" SS approach, to SS approaches augmented with explicit state removal and/or reliable signaling, to a "pure" HS approach. Our goal in doing so was not to argue whether a HS or a SS approach was "better" in some absolute sense. Instead, noting protocols that fall into one category will adopt mechanisms typically associated with the other, we sought to understand how the mechanisms that have evolved into being included in various HS and SS signaling protocols can best be used in given situations, and why. We defined a set of generic protocols that lay at various points along the HS/SS spectrum and developed a unified parameterized analytic model that allows us study their performance. Indeed the fact that a single model can capture a range of signaling protocols (from pure SS to HS, and variations in between) suggests that the protocols are not as different as one might think. Our results indicate that among the class of SS approaches, an SS approach coupled with explicit removal substantially improves state consistency, while introducing little additional signaling message overhead. The addition of reliable explicit setup/update/removal further allows the SS approach to achieve comparable (and sometimes better) consistency than that of the HS approach.

Our focus on this paper has been primarily quantitative and performance oriented. We are currently exploring various ways to quantify the nonperformance oriented complexity of various signaling approaches. Here, architectural issues such as the coupling of signaling with other system components (e.g., the fact that HS protocols require an external notification mechanism, or an addition internal heartbeat mechanism), will be important.

References

[1] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, and S. Molendini, "RSVP refresh overhead reduction extensions," RFC 2961, 2001.
[2] B. Cain, S. Deering, B. Fenner, and A. Thyagarajan, "Internet group management protocol, version 3," RFC 3376, 2002.
[3] D. D. Clark, "The design philosophy of the DARPA internet protocols," presented at the SIGCOMM, Stanford, CA, 1988.
[4] S. Deering, "Host extensions for IP multicasting," RFC 1112, 1989.
[5] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "The PIM architecture for wide-area multicast routing," *IEEE/ACM Trans. Netw.*, vol. 4, no. 2, pp. 153–162, Apr. 1996.

[6] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol independent multicast-sparse mode (PIM-SM): Protocol specification," RFC 2362, Jun. 1998 [Online]. Available: http://www.faqs.org/rfcs/rfc2362.html

[7] W. Fenner, "Internet group management protocol, version 2," RFC 2236, Nov. 1997.

[8] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 784–803, Dec. 1997.

[9] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "Sip: Session initiation protocol," RFC 2543, 1999.

[10] "Kazaa file sharing network," 2002 [Online]. Available: http://www.cazaa.com/

[11] J. C. Lui, V. Misra, and D. Rubenstein, "On the robustness of soft-state protocols," *IEEE ICNP*, pp. 50–60, 2004.

[12] P. Pan and H. Schulzrinne, "Staged refresh timers for RSVP," presented at the 2nd Global Internet Conf., Phoenix, AZ, 1997.

[13] C. Partridge and S. Pink, "An implementation of the revised internet stream protocol (ST-2)," *J. Internetw.: Res. Exp.*, vol. 3, no. 1, Mar. 1992.

[14] Q2931. ITU-T Recommendation.

[15] S. Raman and S. McCanne, "A model, analysis, and protocol framework for soft state-based communication," presented at the SIGCOMM, Boston, MA, 1999.

[16] P. Sharma, D. Estrin, S. Floyd, and V. Jacobson, "Scalable timers for soft state protocols," presented at the IEEE INFOCOM, Kobe, Japan, 1997.

[17] C. Topolcic, "Experimental internet stream protocol: Version 2 (ST-II)," Internet RFC 1190, Oct. 1990.

[18] S. Zabele, M. Dorsch, Z. Ge, P. Ji, M. Keaton, J. Kurose, J. Shapiro, and D. Towsley, "Sands: Specialized active networking for distributed simulation," presented at the DARPA Active Networks Conference and Exposition (DANCE), San Francisco, CA, 2002.

[19] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource reservation protocol," *IEEE Netw.*, vol. 7, no. 5, pp. 8–18, Sep. 1993.

Ping Ji received the B.A. degree in computer science and technology from Tsinghua University, Beijing, China, in 1998, and the Ph.D. degree in computer science from the University of Massachusetts, Amherst, in 2003.

Since 2003, she has been a faculty member of the Master's Program in Forensics Computing at the Mathematics and Computer Science Department, John Jay College of Criminal Justice, New York, NY, and a faculty member of the Ph.D. program in Computer Science of the Graduate Center, City University of New York (CUNY), Flushing, NY. Her research interests include protocol design, performance analysis, and signaling for advanced networks services, network security, wireless and sensor networks, and network measurements and performance modeling.

Dr. Ji has been a recipient of the PSC-CUNY Research Award for three consecutive years.

Zihui Ge received the B.A. degree in computer science and technology from Tsinghua University, Beijing, China, in 1998, the M.S. degree in computer science from Boston University, Boston, MA, in 2000, and the Ph.D. degree in computer science from the University of Massachusetts, Amherst, in 2003.

Since 2004, he has been working as a Researcher in the Networking Management and Performance Department, Internet and Networking Systems Research Center, AT&T Research Labs, Florham Park, NJ. His research interests include Internet technologies and applications, peer–peer systems and overlay networks, network traffic measurement and characterization, network traffic management and anomaly detection, network security, distributed publish/subscribe architecture and applications, and multimedia content distribution systems.

Jim Kurose (F'97) received the B.A. degree in physics from Wesleyan University, Middletown, CT, and the Ph.D. degree in computer science from Columbia University, New York, NY.

Currently, he is a Distinguished Professor (and past chairman) in the Department of Computer Science, University of Massachusetts, Amherst. He has been a Visiting Scientist at IBM Research, INRIA, and Institute EURECOM. His research interests include network protocols and architecture, network measurement, multimedia communication, network and operating system support for servers, and modeling and performance evaluation.

Dr. Kurose is the past Editor-in-Chief of the IEEE Transactions on Communications and was the founding Editor-in-Chief of the IEEE/ACM Transactions on Networking. He has been active in the program committees for IEEE Incomer, ACM SIGCOMM, and ACM SIGMETRICS Conferences for a number of years, and has served as Technical Program Co-Chair for these conferences. He was a recipient of a number of awards for his educational activities, including the IEEE Taylor Booth Education Medal. He is a Fellow of the ACM.

Don Towsley (F'95) received the B.A. degree in physics and the Ph.D. degree in computer science from the University of Texas, Austin, in 1971 and 1975, respectively.

Currently, he is a Distinguished Professor at the Department of Computer Science, University of Massachusetts, Amherst. From 1976 to 1985, he was a member of the faculty of the Department of Electrical and Computer Engineering, University of Massachusetts. He has held visiting positions at IBM T. J. Watson Research Center, Yorktown Heights, NY (1982–1983); Laboratoire MASI, Paris, France (1989–1990); INRIA, Sophia-Antipolis, France (1996); and AT&T Labs–Research, Florham Park, NJ (1997). His research interests include networks and performance evaluation.

Dr. Towsley currently serves as the Editor-in-Chief of the IEEE/ACM Transactions on Networking, and has previously served on several editorial boards including those of the *Journal of the ACM* and IEEE Transactions on Communications. He was a Program Co-chair of the joint ACM SIGMETRICS and PERFORMANCE'92 Conferences and the Performance 2002 Conference. He is a member of ACM and ORSA, and Chair of IFIP Working Group 7.3. He has been elected a Fellow of ACM. He was a recipient of the 1998 IEEE Communications Society William Bennett Paper Award and three Best Conference Paper Awards from ACM SIGMETRICS. He has also been named the recipient of the 2007 IEEE Kobayashi Computers and Communications Award.