2003

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

Ping Ji
University of Massachusetts - Amherst

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

Ping Ji, Zihui Ge, Jim Kurose, and Don Towsley
Computer Science Department,
University of Massachusetts at Amherst,
{jiping,gezihui,kurose,towsley}@cs.umass.edu

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 a "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.

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 classes of signaling approaches can be identified: so-called hard-state and soft-state approaches. Between these two extremes lie signaling approaches that in practice 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, 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 the state-installer and state-removal-by-timeout at the state-holder. Soft-state approaches have been taken in numerous protocols, including RSVP [19], SRM [9], PIM [6], [5], [7], SIP[10] 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 the 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 approach and a pure soft-state approach 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 example, in IGMPv1[4], soft-state timeout at a router was used to detect the departure of previously registered hosts; IGMPv2/v3 [8], [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 from 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 evolved into being 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 approaches ranging from a "pure" soft-state approach, to soft-state approaches augmented with explicit remote state removal and/or reliable signaling, to a "pure" hard-state approach. We define a set of generic protocols that lie along this spectrum, and develop a unified parameterized analytic 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 the class of soft-state approaches, we find that a soft-state 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 soft-state approach to achieve comparable (and sometimes better) consistency than that of the hard-state approach.

Our work focuses on evaluating the *performance* of different signaling protocols. However, there are other non-performance-oriented complexity of various signaling approaches (e.g., the complexity of protocol implementation), which may be examined by other evaluation mechanisms, and is 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-A presents an analytic model for examining the performance of these approaches in the single-hop case, and compares their performance. Section III-B considers the multi-hop case. Section IV discusses related work. Finally, section V summarizes this paper and discusses future work.

II. Soft-state, hard-state and protocols in between

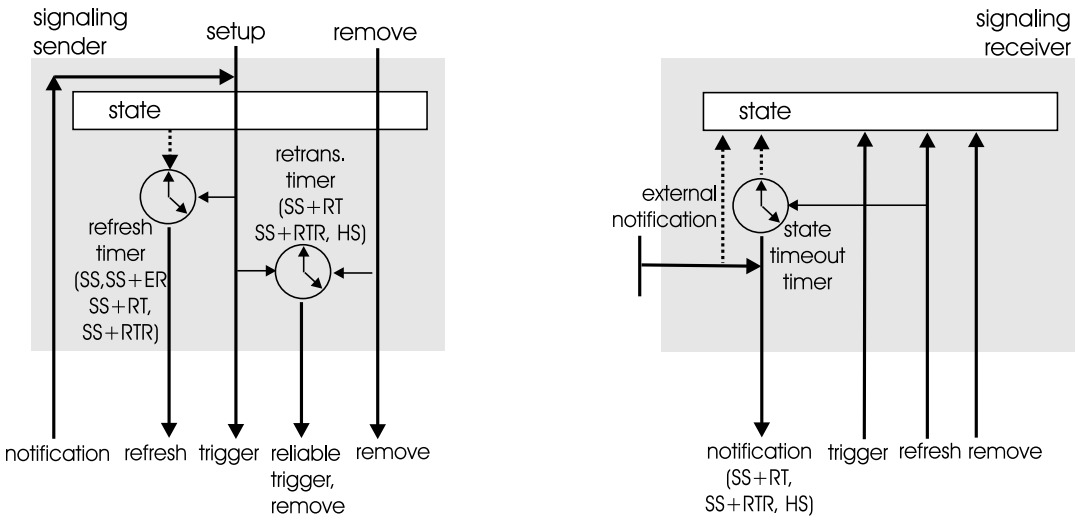

Fig. 1. Signaling sender and receiver: messages and mechanisms

In this section, we describe the operation of five different abstract signaling protocols. These protocols differ in the manner in which state is installed, maintained, and removed, and whether selected signaling messages are transported best-effort or reliably. We will consider a single node (that we will refer to as the "signaling sender") that wishes

to install, maintain, and eventually remove (or have removed) state at a remote node (that we will refer to as the "signaling 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 signaling sender state value equals the signaling 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 will consider the following five approaches:

Pure soft-state (SS): In this approach, the signaling sender sends a *trigger* message [1] that contains state installation or update information to the signaling receiver, and starts a state refresh timer (with value T). When the state-refresh timer expires, the signaling sender sends out a *refresh* message [19] containing the most up-to-date signaling state information, and resets the refresh timer. Trigger and refresh message are sent in a best-effort (unreliable) manner. When a trigger or refresh message is received at the signaling 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 signaling 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 signaling 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 signaling receiver), or because refresh messages have been lost in transmission, and have resulted in a state timeout at the signaling receiver. We will refer to the latter case as *false removal* of state, since the signaling 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 signaling receiver carrying explicit state-removal information. State refresh and trigger messages, and 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 destination not only updates signaling state, but also sends an acknowledgment to the sender. If no trigger acknowledgment is received before the retransmission timer expires, the signaling sender resends the trigger message. Secondly, SS+RT also employs a notification mechanism in which the signaling destination informs the signaling sender about state removals due to state-timeout timer expiration. This allows the signaling 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 the SS+RTR approach uses reliable messages to handle not only state setup/update but also *state removal*.

Hard-State (HS) approach: In the HS approach, reliable explicit messages are used to setup, update and remove state at the signaling 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 at the signaling receiver. Because the hard-state protocol lacks the timeout of state removal, 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 signaling sender crashes and then inform the signaling receiver of this event. Alternatively, the external signal might be generated via a notification from a lower layer protocol at the signaling receiver that had an association with a lower layer protocol at the signaling sender and hence can detect signaling sender crashes. Once such an external notification (signal) is received, the hard-state signaling approach cleans up the orphaned signaling state associated with the signaling sender. One complicating factor is that of *false notification* - the external signal may falsely detect a signaling sender crash (this would occur, for example, if a series of heartbeat messages were lost, but the signaling sender was still operational). As in the case of SS+RT, false notification can be repaired by having the signaling receiver notify the signaling 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.

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

In the following section, 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 the *cost* associated with a given signaling approach and the level of consistency it is able to achieve. One aspect of this cost will be the signaling message rate itself. A second aspect of this cost is the cost associated with being in an inconsistent state. For example, in IGMP, when an end host leaves without signaling its departure to its edge router, multicast data will continue 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 will result in other peers attempting to contact the departed peer - again, a cost. In section 3, the cost function we adopt is a weighted sum of the signaling overhead and application-specific costs (such as unwanted multicast data flows, or connection attempts to a departed peer in the examples above).

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

- **Application-specific inconsistency cost.** As noted above, these are the costs associated with the signaling sender and receiver being in inconsistent states. Clearly, when this cost is high, the signaling sender may want to incur a higher signaling overhead in order to keep the signaling sender and receiver states as consistent as possible.
- **Refresh timeout value.** As noted in [1], the smaller the value of the refresh timer, the sooner that consistent state 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. If the application-specific cost of inconsistent state is high, however, this increased signaling cost may be warranted.
- **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. However, too small a timeout value can result in false state removal.
- **Signaling message loss.** As the probability of message loss becomes higher, we expect that the fraction of time that the signaling sender and receiver states are inconsistent will 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, we expect those protocols with explicit reliable transfer to be preferable.

- **Number of hops.** In certain 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 and intuition.

III. Modeling and Analysis of Signaling Approaches

We begin our analysis by considering the simple example of a single node (the "signaling 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"). We focus here on a single 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 abstract signaling approaches described in the previous section. We assume that the signaling 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 signaling sender and receiver are the only two entities involved in the signaling protocol. As shown in Figure 2, we can think of the two entities as being connected through a single

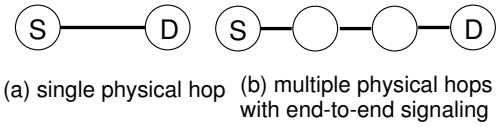

Fig. 2. single-hop signaling systems

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 certain peer-to-peer file sharing applications such as Kazaa [11], 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 the peer is in the system and serving files.

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

Signaling 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 signaling message reaches the remote receiver, enabling both sender and receiver to achieve consistent state.

Signaling 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.

Signaling state removal. At the end of the lifecycle, 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 signaling state removal. The destination may incorrectly remove state, even though the sender is still maintaining 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 is still maintaining state.

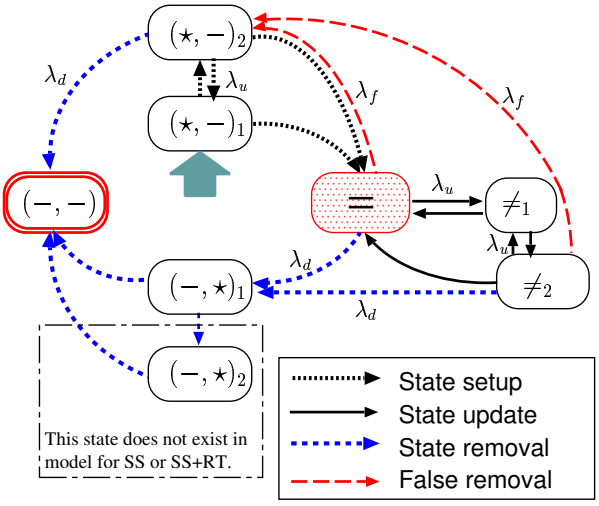

Fig. 3. A continuous time Markov model for signaling approaches in single-hop system

Given these events in the lifecycle of a signaling sender and receiver, we can develop a Markov model, shown in Figure 3, to capture this behavior. 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 signaling sender and receiver, respectively:

- Markov state $(\star,-)$ captures the initial stage of the lifecycle, when signaling state has been installed at the sender but not at the receiver. This is an inconsistent state, since the sender and receiver's state values do not match.
- Markov 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 signaling state, the state of the Markov chain is $=$.
- When the sender and the destination have different signaling state (i.e., both have installed state, but the state values are different), the Markov chain is in states $\neq$.
- When the signaling state is removed from both the sender and the receiver, the system enters an absorbing state represented by Markov state $(-,-)$.

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 we will describe shortly. In Figure 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 Figure 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 considered in the state transitions are:

λ_u: signaling state update rate

λ_d: $1/\lambda_d$ is the sender's mean signaling state lifetime

λ_f: false state removal rate at receiver

D: signaling channel delay

p_l: signaling channel loss rate

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

T: soft-state refresh timer value

X: soft-state state timeout timer value.

R: message retransmission timer value for reliable transmission

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

In Section 2, we discussed five different approaches towards signaling. Each of these approaches can be modeled using the model shown in Figure 3, with different transition rates (and in some cases disabled transitions) for each of the approaches. We next describe the model transitions for each of these different signaling approaches. These transitions are shown either in the model diagram or in Table 1.

Soft-State (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 destination. 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 destination. Since refreshes are sent periodically with interval T, and each message reaches the destination 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 setup 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 average delay D, which corresponds to a transition back 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 $\neq_2$ back 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 while a trigger message is on its way to the receiver. We assume that an update happens either before a previous trigger message is sent out or after the trigger message has already reached the receiver (or has been lost).

Sender signaling state is removed at rate λ_d, i.e., a sender has a session of mean length $1/\lambda_d$. If the signaling 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 information 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 to remove the orphaned state. 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 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 happens 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 noted above.

Soft-State with Explicit Removal (SS+ER) model: Recall that in SS+ER, a signaling message carries explicit state removal information (in addition to the state-timeout mechanism) to remove signaling state. We model this explicit removal by modifying the state removal process in the SS model as follows. When the Markov chain enters 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 by letting the Markov chain transit from $(-,\star)_1$ to the absorbing state $(-,-)$ with rate $(1-p_l)/D$. The loss of the explicit removal message causes the Markov chain to transit from $(-,\star)_1$ to $(-,\star)_2$. From there, the system transits to the absorbing state $(-,-)$ at rate $1/X$, capturing the state removal caused by the state-timeout timer expiration.

Soft-State with Reliable Trigger messages (SS+RT) model: The Markov model for SS+RT differs from the model for 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 Markov chain from state $\neq_2$ or state $(\star,-)_2$ to state $=$ with rate $(1/T + 1/R)\cdot(1-p_l)$.

Soft-State with Reliable Trigger/Removal message (SS+RTR) model: The Markov model for SS+RTR differs from the model for SS+RT in that, when an explicit removal message is lost, the system waits for the state-timeout timer to expire or a successful retransmission of the removal message to go into state $(-,-)$. Thus the transition rate from state $(-,\star)_2$ to state $(-,-)$ is $1/X + (1-p_l)/R$.

Hard-State (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 that performs the failure-detection for the hard-state 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 "heart-beat" exchanging mechanism may have an *overall* overhead comparable to that of SS+RTR. Nonetheless, we consider the failure-detection as a separate component in the system architecture with the signaling mechanism. Therefore, in our paper, we exclude this part from the analysis. However, we assume that the external signal can be falsely generated with rate λ_w, which causes a *faulty removal* of a signaling state in the HS approach.

We summarize the protocol-specific state transition of the Markov chain for different signaling approaches in Table 1, where λ_{ij} denotes the state transition rate from Markov state i to j.

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

TABLE I

MODEL TRANSITIONS

2) 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 consistent 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 outside state $(=)$, before it eventually transits to the absorbing state $(-,-)$. This is equivalent to evaluating the sum of the stationary probabilities of the inconsistent states in the recurrent Markov model where the absorption state $(-,-)$ and the starting state $(\star,-)_1$ are merged. Let π_i be the stationary probability of the recurrent Markov model in state i. We have the following expression for δ:

$$\begin{aligned}\delta &= \pi_{(\star,-)_1} + \pi_{(\star,-)_2} + \pi_{\neq_1} + \pi_{\neq_2} + \pi_{(-,\star)_1} + \pi_{(-,\star)_2} \\ &= 1 - \pi_{=} \end{aligned} \tag{1}$$

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:

$$\Gamma = \mathcal{T} \cdot \gamma \tag{2}$$

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 signaling 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 Markov chain 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 mean message rate for explicit triggers, γ_{et}, is,

$$\begin{aligned}\gamma_{et} &= \pi_{(\star,-)_1}\lambda_{(\star,-)_1,=} + \pi_{\neq_1}\lambda_{\neq_1,=} + \\ &\quad \pi_{(\star,-)_1}\lambda_{(\star,-)_1,(\star,-)_2} + \pi_{\neq_1}\lambda_{\neq_1,\neq_2} \end{aligned} \tag{3}$$

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

$$\gamma_{er} = \pi_{(-,\star)_1}\lambda_{(-,\star)_1,(-,-)} + \pi_{(-,\star)_1}\lambda_{(-,\star)_1,(-,\star)_2} \tag{4}$$

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,

$$\gamma_r = \frac{1}{T}\cdot(\pi_{(\star,-)_2} + \pi_{=} + \pi_{\neq_2}) \tag{5}$$

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 mean message rate for reliable triggers, γ_{rt}, can be computed by,

$$\gamma_{rt} = \frac{1}{R}(\pi_{(\star,-)_2} + \pi_{\neq_2}) + \sum_i \pi_i\lambda_{i,=} + \lambda_f(\pi_{=} + \pi_{\neq_2}) \tag{6}$$

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 mean message rate γ_{rr} is:

$$\gamma_{rr} = \frac{1}{R}\pi_{(-,\star)_2} + \pi_{(-,\star)_1}\lambda_{(-,\star)_1,(-,-)} + \pi_{(-,\star)_2}\lambda_{(-,\star)_2,(-,-)} \tag{7}$$

In summary, the overall mean message rate for different signaling protocols are as follows:

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

3) Model Evaluation: We now compare and contrast the performance of the five different signaling approaches using our modeling framework. In order to use representative parameter values, let us consider as an example, the signaling process between a Kazaa regular peer (hereafter, simply referred to as a peer) and its supernode (as described in the beginning of Section III-A). Unless otherwise noted, we use the following default parameters: $p_l = 0.02$, $D = 30ms$, $1/\lambda_u = 20s$, $1/\lambda_d = 1800s$, $T = 5s$, $X = 3T$, $R = 4D$, and $\lambda_w = 0.0001$. These parameter values are chosen to capture the behavior of a Kazaa session: a signaling state is added when the 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 peer-state maintained by the supernode should

be deleted. If this state is not removed at the supernode, an inconsistent state will occur. As a result, the supernode may respond to other peers incorrectly (e.g., directing them to an already departed peer; these other peers may then fruitlessly contact the departed peer, decreasing the usability of the application).

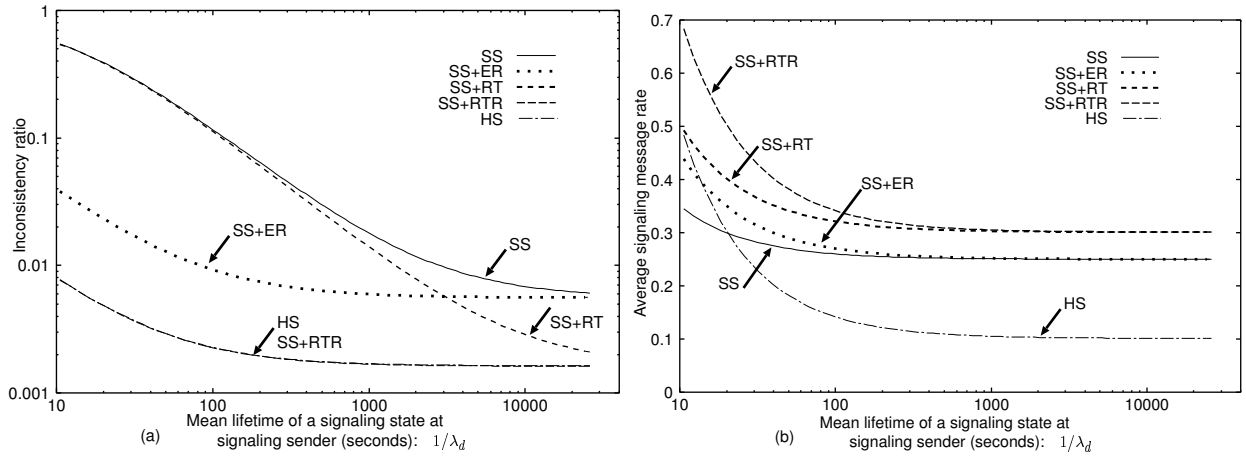

Fig. 4. Comparison against the time that a singling state exists at the sender (i.e., session length), $1/\lambda_d$

Impact of session length ($1/\lambda_d$). We first study the performance of different signaling approaches as a function of the expected amount of time that signaling state is installed at the signaling sender, $(1/\lambda_d)$. In our Kazaa example, this corresponds to a peer's average session length. In Figure 4 (a), we plot the inconsistency ratio δ, and in Figure 4 (b), we plot the normalized average signaling message rate γ^* for the different signaling approaches. Figure 4 provides a 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 minutes every hour versus 2 hours every day), the system is likely to incur more 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 Figure 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 soft-state signaling in such circumstances. This is because that the "penalty" of not using explicit removal is so high.
- Figure 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 shorter (towards the left of Figure 4(a))

the five approaches are grouped on the basis of how state removal is performed: those without explicit removal (SS, SS+RT), those with explicit removal (SS+ER) and those with reliable removal (SS+RTR, HS). 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 non-trivial additional signaling overhead. Thus, for these application parameters, providing reliable trigger messages does not appear to be very crucial.

- SS+RTR provides essentially the same inconsistency ratio as HS. This suggests that beyond explicit removal and reliable transmission, any enhancements to soft-state 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.

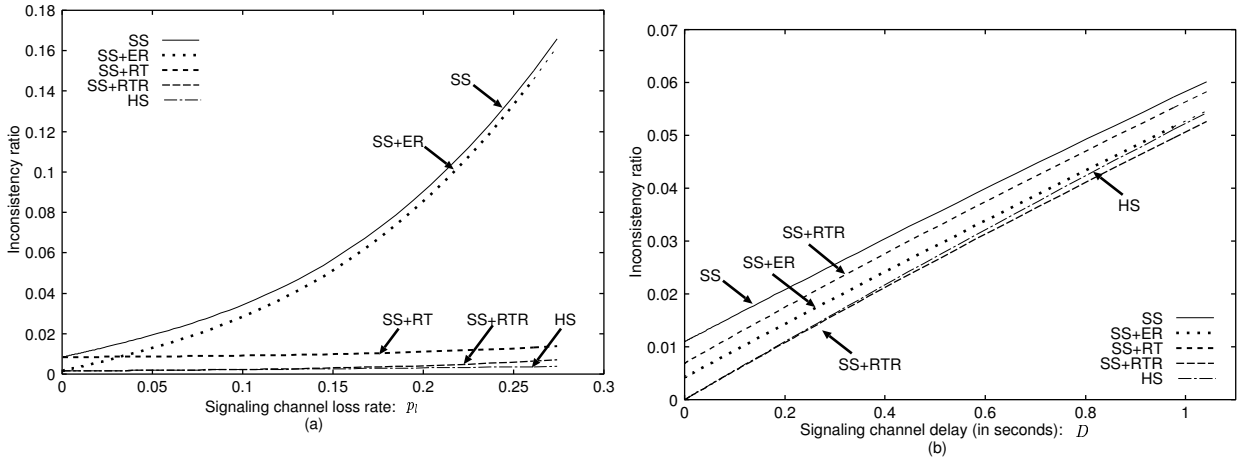

Fig. 5. Comparison against link loss rate p_l and link delay D

Impact of message loss and delay.

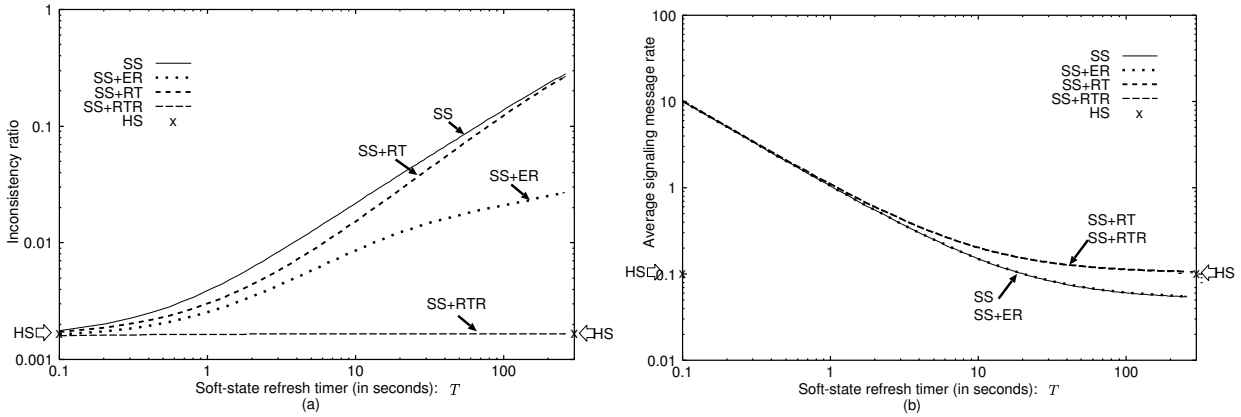

Fig. 6. Comparison against soft-state refresh timer (T), single hop case

Figure 5 plots the inconsistency ratio for different signaling approaches for various loss rates (a) and delays (b). Figure 5(a) indicates that even for modest loss rates (e.g., 5%), reliable transmission significantly improves the performance of soft-state protocols. Figure 5(b) plots the inconsistency ratio versus the one-way sender-to-receiver delay. We observe an approximately linear increase in the inconsistency ratio under all signaling approaches. However, signaling approaches with reliable transmission exhibit a slightly larger slope. 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 suffers longer latencies in an environment with longer transmission delays, while soft-state approaches that only rely on a refresh mechanism do not.

Impact of timer configuration. There are three different timers used in the five signaling approaches we consider: the soft-state refresh timer, the soft-state state-timeout timer and the retransmission timer. Figure 6 explores the performance of different soft-state signaling approaches under different soft-state refresh timer settings. Since HS does not employ a refresh mechanism, it is shown as a '$\times$' on the y-axis. When the refresh timer value changes, we set the state-timeout timer to be 3 times the value of the refresh timer. Figure 6 reveals an interesting tradeoff between having a short refresh timer (to reduce the inconsistency ratio) and a long refresh timer (to keep signaling message overhead low).

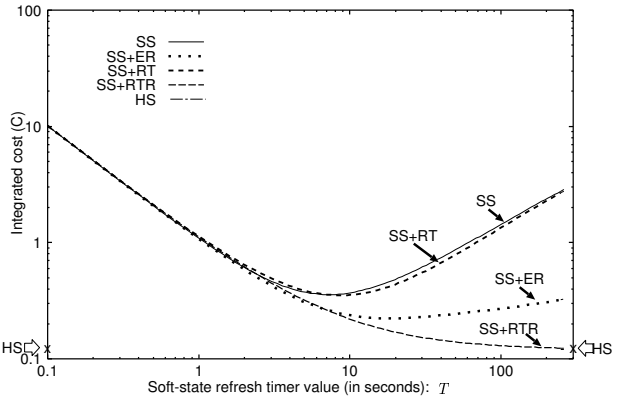

Fig. 7. Soft-state refresh timer (T)

Overall Cost. As discussed earlier, there are two components of overall cost: signaling 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 was the transmission of unwanted multicast data; in the case of Kazaa, this latter cost was 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

$$C = c \cdot \delta + \gamma^* \tag{8}$$

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 the following, we set c to be 10 (msg/sec).

In Figure 7, we plot the integrated cost associated with different signaling approaches versus the soft-state refresh timer value, (T). From this figure, we observe that there exists 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 rather longer refresh timer values. Last, for SS+RTR, a longer timer value is preferred, and when the timer is large enough (on the order of 100s of seconds), it provides comparable performance to the hard-state approach.

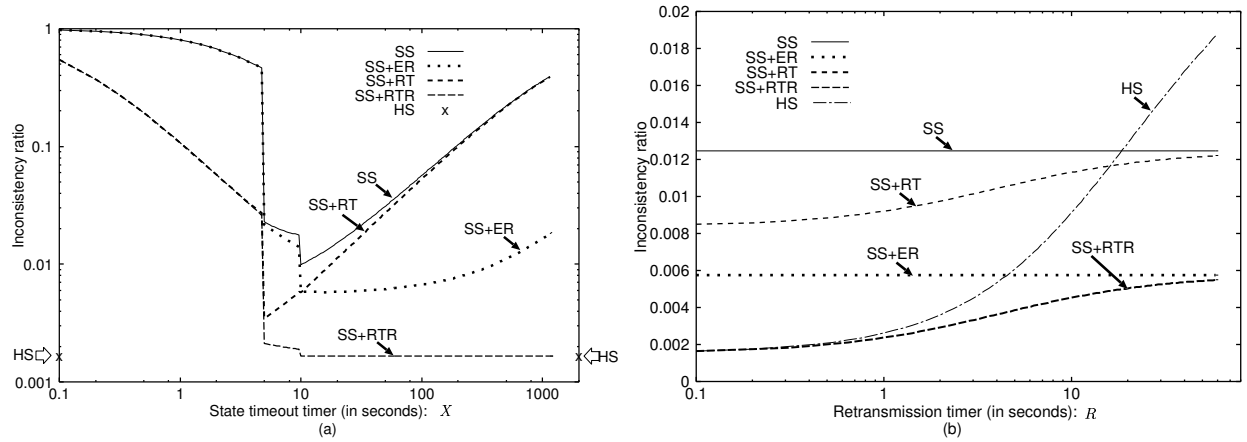

Fig. 8. state timeout timer (X) for soft-state based approaches and retransmission timer R for reliable transmissions

Figure 8 (a) explores the impact of different state timeout timer values on the inconsistency ratio of soft-state approaches. Here we fix the state refresh timer to be 5 seconds and vary the state-timeout timer. The results indicate that, when the state-timeout timer is shorter than the refresh timer, all soft-state based approaches perform poorly, since refresh messages arrive too late to "keep alive" the signaling state at the signaling receiver. Once the state-timeout timer value is greater than the refresh timer value, the different approaches behave very differently: SS+RTR does well with long timeout values, since the longer 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 length of the refresh timer, so that the probability of false removal is reduced. However, since longer timeout timers add larger delays to remove orphaned state, SS and SS+ER also require the state timeout timer to be short 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.

Figure 8 (b) explores the impact of different retransmission timer values on the inconsistency ratio of the five signaling approaches. 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 average signaling message rate. By varying the soft-state refresh timer, we study the tradeoff between the inconsistency ratio and the average signaling message rate of different signaling approaches, and show the results in Figure 9. Since hard-state does not use the refresh timer, neither the inconsistency ratio nor the average signaling message rate vary with T; in Figure 9, hard-state is shown as a single point $\times$. Figure 9 also indicates that the consistency quality of SS+RTR is not sensitive to soft-state refresh rate (which is determined by the refresh timer), whereas the consistency qualities of the other soft-state approaches change with the signaling overhead. We also examined the tradeoffs between inconsistency ratio and signaling overhead based on other system or design parameters. Figure 10 shows the tradeoff between the inconsistency ratio and the message overhead by varying the signaling state update rate λ_u (Figure 10 (a)) and by varying the signaling channel delay D (Figure 10 (b)). From Figure 10 (a), we observe that for large inconsistency probability (> 0.01), to achieve the same consistency

quality, SS uses the least amount of signaling messages comparing to all other signaling approaches; On the other hand, to achieve small inconsistency probability (< 0.004), HS should be used to save the signaling overhead. From Figure 10 (b), we observe that the tradeoff curves are not sensitive to changing signaling channel delays.

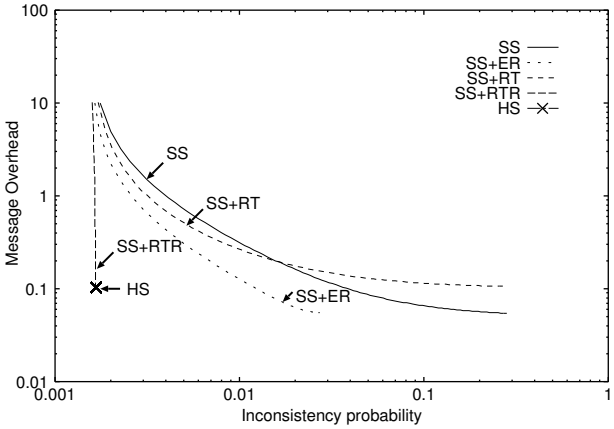

Fig. 9. Tradeoff between inconsistency ratio and average signaling message rate, derived by varying T

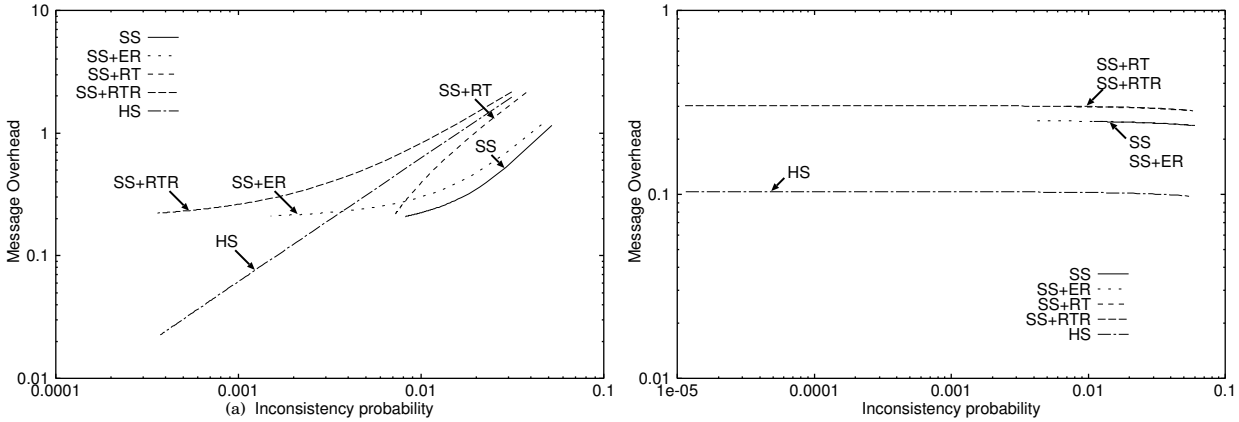

Fig. 10. Tradeoff between inconsistency ratio and average signaling message rate: **(a)** derived by varying $1/\lambda_u$, **(b)** derived by varying D.

While our model assumes exponentially distributed timer values, in practice, signaling protocols usually use deterministic timers. To investigate the impact of our exponential assumption on the timer values, we build simulations that use deterministic timers under the same system settings. Our simulation results indicate that using deterministic timers does not affect our observations and conclusions. For example, in comparison to the evaluation results shown in Figure 4, the inconsistency ratio slightly differs ($< 1\%$) between the analytical and the simulation results with deterministic timers. For average signaling message rate, the difference between the analytical and the simulation results is between 5% to 15%, while the qualitative relative behavior for different signaling protocols remains unchanged. Figure 11 depicts the results, where the simulation results are shown as dotted curves with 95% confidence interval. Simulation results, in comparison to the evaluation results shown in Figure 6, are shown in Figure 12, where we also observe small performance difference (less than 3%) between the schemes of using deterministic timers and using exponential timers.

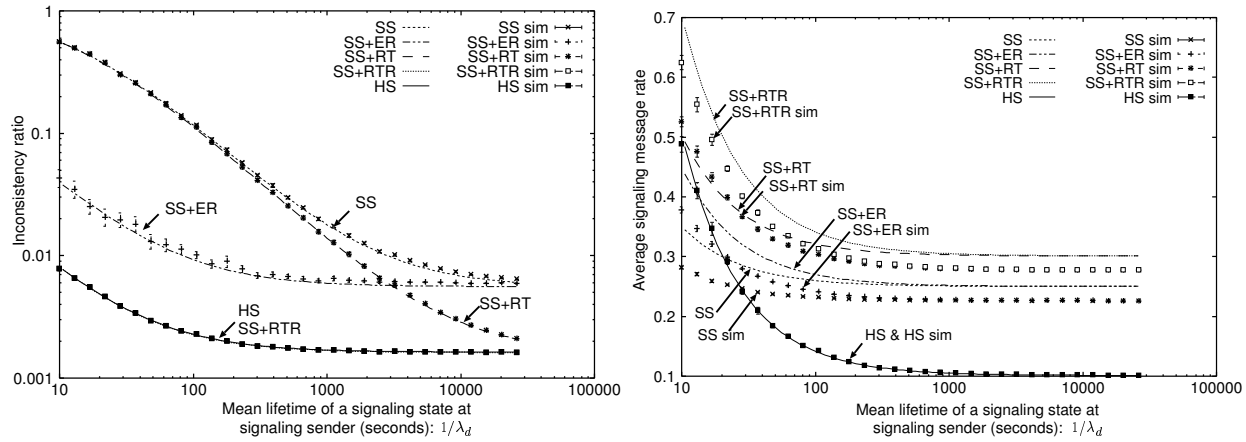

Fig. 11. Impact of using exponential timers, with changing state lifetime $1/\lambda_d$

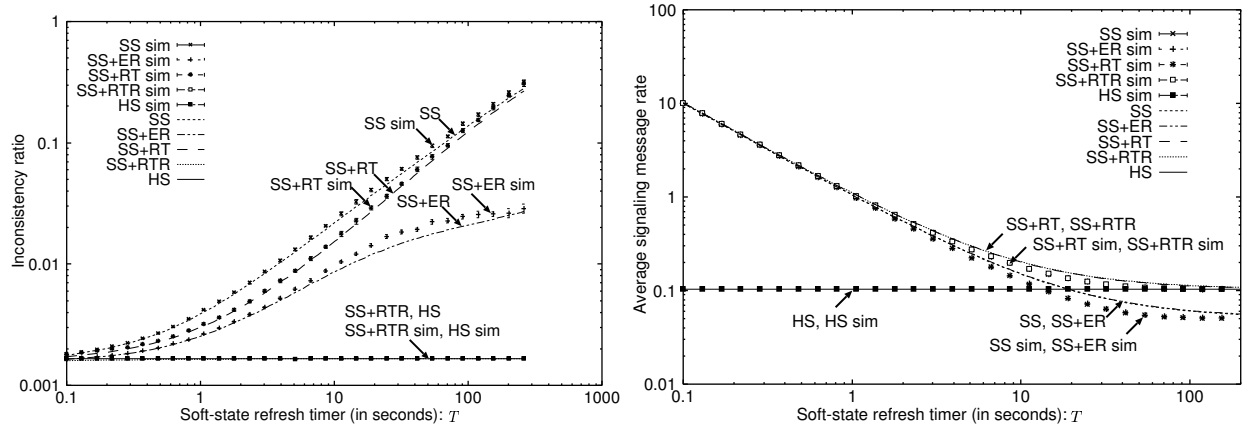

Fig. 12. Impact of using exponential timers, with changing soft-state refresh timer T

B. Signaling in a Multi-hop System

In this subsection, we consider the case in which only the end systems (the signaling sender and receiver) but also intermediate routers or relay nodes must maintain signaling state. This would occur, for example, in the case the intermediate routers need to maintain a bandwidth reservation for communication between the end systems. Figure 13 illustrates the abstract model of a multi-hop signaling system that involves a single signaling sender and a chain of signaling destinations. We require that signaling state generated at the signaling sender be maintained at all nodes along the path between the signaling sender and the final destination.

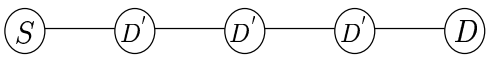

Fig. 13. A sample of a multi-hop system

1) Model for Multi-hop Systems: In Section III-A, 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 of multiple hops. We focus on the stationary process in a multi-hop signaling system, in which a 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 ($\lambda_d \to \infty$). We model state updates as a Poisson process with rate λ_u.

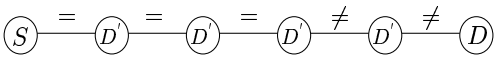

Fig. 14. A five-hops system with three consistent hops

Our modeling framework for a multi-hop system is an extension of the single-hop model. Let N be the total number of hops (links) in a multi-hop system and n the number of consistent hops (links). We assume that these hops are homogeneous, i.e., they have identical channel loss rate (p_l) and mean channel delay (D), and assume that channel losses are independent. Here, we define a *consistent hop* to be a hop (link) with two ends having consistent state information. Figure 14 illustrates a five-hop ($N = 5$) system with three consistent hops ($n = 3$).

We define the state space as $\mathcal{S} = \{(n, s)\}$, where $0 \leq n \leq N$ is the number of consistent hops, and s is a special variable taking on values of 0 or 1 to indicate whether the Markov chain is in a *fast path* Markov state ($s = 0$) or a *slow path* ($s = 1$) Markov 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 period when 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 signaling 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 in one node, all receivers beyond this node in the linear topology will also time out, because they too will not receive the refresh messages. We use the Markov model in Figure 15 to model the SS protocol in the multi-hop environment.

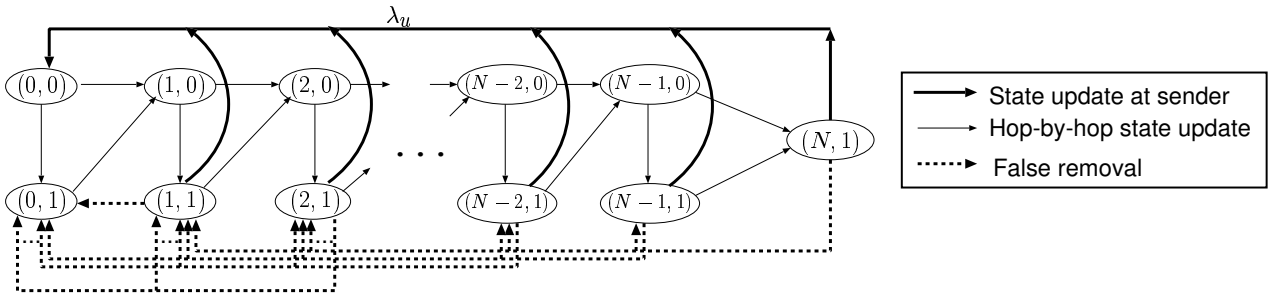

Fig. 15. Markov model for end-to-end soft-state approach

Consider the state update process. When signaling state is updated at the signaling sender, the Markov chain transits to state $(0, 0)$ from other Markov states. State $(0, 0)$ 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, $s=1$).

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 i-th 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 N-th 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 $\lambda^{(e)}_{(i,1),(j,1)}$ can be calculated by Equation (9).

$$\lambda_{(i,1),(j,1)} = \begin{cases} \frac{1}{X}\cdot[1-(1-p_l)^{j+1}]^{\lfloor\frac{X}{T}\rfloor} \\ -\frac{1}{X}\cdot[1-(1-p_l)^{j}]^{\lfloor\frac{X}{T}\rfloor}, \quad 1\le j<i\le N \\ \\ 0, \qquad\qquad\qquad\qquad \text{otherwise} \end{cases} \tag{9}$$

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 the timeout happens at the $(j+1)$-th signaling receiver, but not at any preceding hop.

Model transitions for SS+RT protocol. Under SS+RT protocol, when the system is trapped 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 that reliable transmission is used. Therefore, in SS+RT, the transition rate from state $(i-1,1)$ to state $(i,0)$ becomes

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

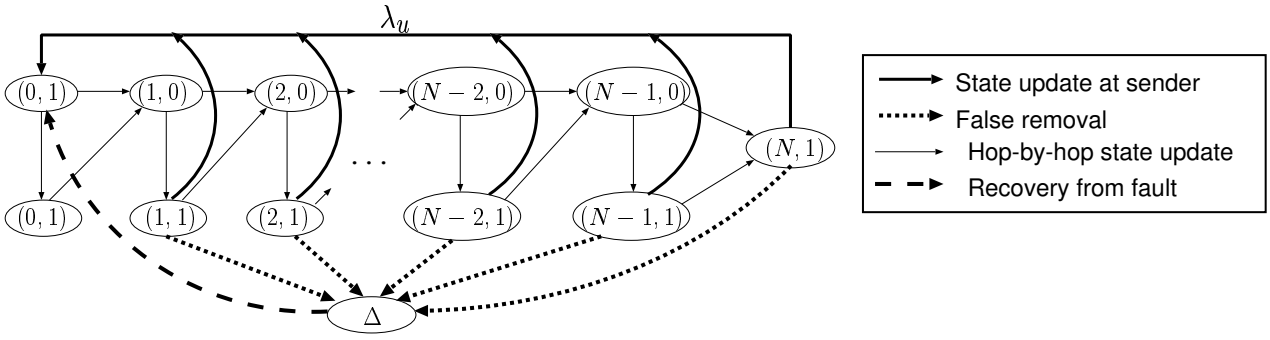

Fig. 16. Markov model for hard-state signaling approach

Model transitions for Hard-state (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 soft-state timeout removal are employed. Thus, a state transition from state $(i-1,1)$ to state (i,0) is achieved via retransmission only, and the transition rate is

$$\lambda_{(i-1,1),(i,0)} = \frac{1}{R}(1-p_l). \tag{11}$$

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) happens. 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 message, it sends a trigger signaling message to re-install 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.

2) Multi-hop Model Solution and Results: The solution of the multi-hop model is similar to that of the single-hop model. Let π_i be the stationary probability of the multi-hop model in state i. The inconsistency ratio δ for multi-hop system is

$$\delta = 1 - \pi_{(N,1)} \tag{12}$$

The mean signaling message rate γ for the SS approach is

$$\gamma_{ss} = \sum_{k=1}^{N-1} \pi_{(k,0)} \cdot \frac{1}{D} + \sum_{k=0}^{N} \pi_{(k,1)} \cdot \frac{1}{T} \cdot \bar{E} \tag{13}$$

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

$$\bar{E} = E[k|loss] \cdot P[loss|msg] + P[succ|msg] \cdot (k+1) \tag{14}$$

and $E[k|loss]$ is the expected number of hops that a packet is lost before it arrives at the k-th hop,

$$\begin{aligned} E[k|loss] &= \sum_{i=1}^{k} i \cdot (1-p_l)^{i-1} \cdot p_l \\ &= -p_l \left(\frac{(1-p_l)(1-(1-p_l)^k)}{p_l} \right)' \\ &= \frac{1-(1-p_l)^k}{p_l} - k(1-p_l)^k. \end{aligned} \tag{15}$$

The mean signaling message rate for SS+RT approach is

$$\gamma_{ss+rt} = \sum_{k=1}^{N-1} \pi_{(k,0)} \cdot \frac{1}{D} + \sum_{k=0}^{N} \pi_{(k,1)} \cdot \left(\frac{1}{R} + \frac{1}{T}\bar{E} \right), \tag{16}$$

and the mean signaling message rate for HS is

$$\gamma_{hs} = \sum_{k=1}^{N-1} \pi_{(k,0)} \cdot \frac{1}{D} + \sum_{k=0}^{N} \pi_{(k,1)} \cdot \frac{1}{R} + \pi_{\Delta} \cdot \frac{2}{D}. \tag{17}$$

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

In Figure 17, we plot the fraction of time that the i-th 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, as the hop is further away

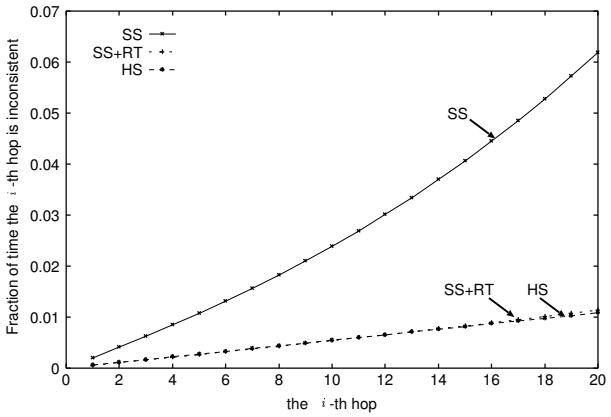

Fig. 17. Fraction of time that the i-th hop is inconsistent, where $1 \leq i \leq N$ and $N = 20$

from the signaling sender. This increasing inconsistency exhibits an approximately linearly trend for all signaling approaches. Combining hop-by-hop reliable triggers with the end-to-end soft-state approach significantly improves the consistency at all signaling hops, with the consistency of SS+RT being comparable to that of the hard-state approach. In our evaluation, the hard-state (HS) approach has 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 soft-state 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.

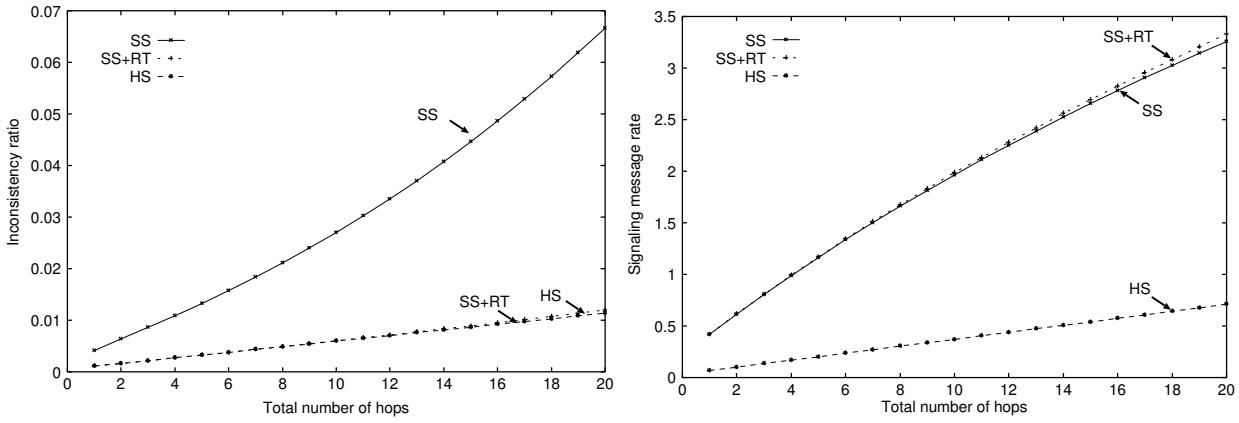

Fig. 18. Inconsistency ratio and signaling message rate against total number of hops

Figure 18 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 multi-hop system. We observe that both inconsistency and signaling message overhead monotonically increase with an increasing number of hops. From Figure 18, comparing the hard-state approach (HS) and the soft-state with reliable trigger approach (SS+RT), indicates that the consistency of the pure soft-state approach (SS) is more sensitive an increase in the number of hops. Figure 18(b) suggests that adding a reliable trigger to end-to-end soft-state approach, while significantly improving consistency, introduces little additional signaling overhead. This benefit increases as the number of signaling hops increases.

We also evaluated the impact of system and design parameters on the performance of multi-hop signaling approaches. Figure 19 shows the impact of soft-state refresh timer T on the performance of signaling approaches, SS,

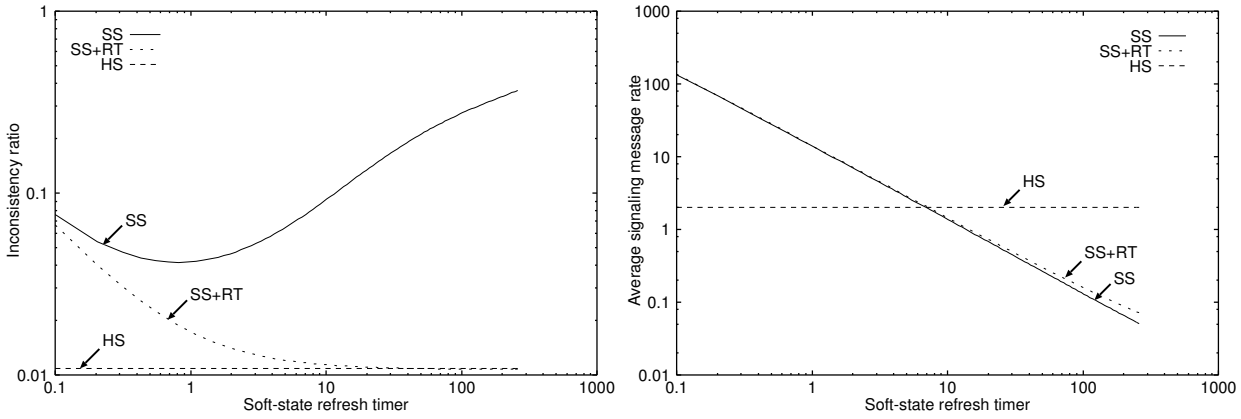

Fig. 19. Comparison against soft-state refresh timer T, multi-hop case

SS+RT and HS for a multi-hop system. From Figure 19 (a), we observe that when the soft-state refresh timer is relatively small ($< 0.9s$), the inconsistency ratio of SS decreases when T increases; however when the soft-state refresh timer gets larger ($> 0.9s$), the inconsistency ratio of SS significantly increases with the increasing value of T. On the other hand, the inconsistency ratio of SS+RT decreases with the increasing refresh timer value, until it reaches an optimal value at the point of $T = 10s$. From Figure 19 (b), we observe that with increasing soft-state refresh timer, the average signaling message rate for both SS and SS+RT decreases.

IV. Related Work

The most closely related work to our present work is [15], the first effort to develop analytic models of soft-state protocols, and also the first (and only) that has sought to develop a more principled understanding of soft-state 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 soft-state (SS) signaling system. However, there are several counterintuitive 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 model artifacts in Raman & MacCane's model, the soft-state refresh timer also depends on the rate of generating new signaling states and the channel service capacity, and as the capacity of the signaling channel increases the degree of inconsistency also *increases*. Our model, by creating an absorbing state, avoids such rather counterintuitive 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 Raman & MacCane's work, the authors proposed a modification for the soft-state signaling protocol, where a NACK message is sent by the receiver when a signaling message is detected to be lost, and they 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 & MacCane's work, such retransmission timer depends on channel capacity and other parameters. In addition, Raman & MacCane use simulation to evaluate the parameters that affect the consistency performance of their modified soft-state 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 hard-state 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 soft-state protocols but to compare and contrast a variety of signaling approaches and their mechanisms, ranging from a "pure" soft state, to soft-state approaches augmented with explicit state removal and/or reliable signaling, to a "pure" hard state approach.

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

V. Conclusion and Future Work

In this paper we have compared and contrasted the performance of a variety of signaling approaches ranging from a "pure" soft-state approach, to soft-state approaches augmented with explicit state removal and/or reliable signaling, to a "pure" hard-state approach. Our goal in doing so was not to argue whether a hard-state or a soft-state 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 hard-state and soft-state 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 hard-state/soft-state 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 soft-state to hard-state, and variations in between) suggests that the protocols are not as different as one might think. Our results indicate that among the class of soft-state approaches, a soft-state 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 soft-state approach to achieve comparable (and sometimes better) consistency than that of the hard-state approach.

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

VI. ACKNOWLEDGEMENT

This work is supported in part by DARPA subcontracts with Northrop Grumman Information Technology 2000-012 and 2000-109, National Science Foundation subcontract with the University of Florida UF-EIES-0205003-UMA,

and National Science Foundation grant ANI-0085848.

REFERENCES

[1] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, and S. Molendini. Rsvp refresh overhead reduction extensions. RFC 2961, April 2001.

[2] B. Cain, S. Deering, B. Fenner, and A. Thyagarajan. Internet group management protocol, version 3, Oct. 2002. RFC 3376.

[3] David D. Clark. The design philosophy of the DARPA internet protocols. In *Proceeding of SIGCOMM*, Stanford, CA, Aug 1988.

[4] S. Deering. Host extensions for ip multicasting, August 1989. RFC 1112.

[5] S. Deering, D. Estrin, D.Farinacci, V.Jacobson, C.Liu, and L.Wei. The pim architecture for wide-area multicast routing. *ACM Transactions on Networks*, April 1996.

[6] S. Deering, D. Estrin, D. Faranacci, V. Jacobson, C. Liu, and L. Wei. The pim architecture for wide area multicasting. *IEEE/ACM Transactions on Networking*, 4(2), Apr. 1996.

[7] 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, June 1998. RFC2362, http://www.faqs.org/rfcs/rfc2362.html.

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

[9] 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 Transactions on Networking*, 5(6), December 1997.

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

[11] Kazaa file sharing network, 2002. http://www.cazaa.com/.

[12] Ping Pan and Henning Schulzrinne. Staged refresh timers for RSVP. In *2nd Global Internet Conference*, Phoenix, AZ, 1997.

[13] C. Partridge and S. Pink. An Implementation of the Revised Internet Stream Protocol (ST-2). *Journal of Internetworking: Research and Experience*, 3(1), March 1992.

[14] Q2931. ITU-T Recommendation.

[15] Suchitra Raman and Steven McCanne. A model, analysis, and protocol framework for soft state-based communication. In *Proceeding of SICOMM*, Boston, MA, 1999.

[16] Puneet Sharma, Deborah Estrin, Sally Floyd, and Van Jacobson. Scalable timers for soft state protocols. In *Proceeding of IEEE INFOCOM*, Kobe, Japan, 1997.

[17] C. Topolcic. Experimental Internet Stream Protocol: Version 2 (ST-II), October 1990. Internet RFC 1190.

[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. In *DARPA Active Networks Conference and Exposition (DANCE)*, San Francisco, California,USA, May 2002.

[19] L. Zhang, S. Deering, D. Estrin, S. Shenker, , and D. Zappala. RSVP: A new resource reservation protocol. *IEEE Network*, September 1993.