SYSTEMATIC PERFORMANCE EVALUATION OF MULTIPOINT PROTOCOLS

Ahmed Helmy, Sandeep Gupta, Deborah Estrin, A. Cerpa, Y. Yu
University of Southern California
helmy@ceng.usc.edu, sandeep@poisson.usc.edu, {estrin,cerpa,yanyu}@catarina.usc.edu

Abstract

The advent of multipoint (multicast-based) applications and the growth and complexity of the Internet has complicated network protocol design and evaluation. In this paper, we present a method for automatic synthesis of worst and best case scenarios for multipoint protocol performance evaluation. Our method uses a fault-oriented test generation (FOTG) algorithm for searching the protocol and system state space to synthesize these scenarios. The algorithm is based on a global finite state machine (FSM) model. We extend the algorithm with timing semantics to handle end-to-end delays and address performance criteria. We introduce the notion of a virtual LAN to represent delays of the underlying multicast distribution tree.

As a case study, we use our method to evaluate variants of the timer suppression mechanism, used in various multipoint protocols, with respect to two performance criteria: overhead of response messages and response time. Simulation results for reliable multicast protocols show that our method provides a scalable way for synthesizing worst-case scenarios automatically. We expect our method to serve as a model for applying systematic scenario generation to other multipoint protocols.

1. INTRODUCTION

The longevity and power of Internet technologies derives from its ability to operate under a wide range of operating conditions. Perhaps more than any other technology, the range of operating conditions is enormous. It is this enormous set of conditions that motivates us to develop techniques for systematic testing of protocol behavior. We do not expect that complex adaptive protocols will be automatically verifiable under their full range of conditions. Rather, we are proposing a framework in which a protocol designer can follow a set of systematic steps, assisted

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35533-7_26

Tommaso Bolognesi and Diego Latella (Eds.), *Formal Methods for Distributed System Development.*

by automation where possible, to cover a specific part of the design and operating space.

In our proposed framework, a protocol designer needs to create the initial mechanisms, describe it in the form of a finite state machine, and identify the performance criteria or correctness conditions that need to be investigated. Our automated method will pick up at that point, providing algorithms that eventually result in scenarios or test suites that stress the protocol with respect to the identified criteria. This paper demonstrates our progress in realizing this vision as we present our method and apply it to the performance evaluation of multipoint protocols.

1.1. PROTOCOL DESIGN ISSUES

Through our proposed methodology for test synthesis, we hope to address the following key issues of protocol design and evaluation. *Scenario dependent evaluation, and the use of validation test suites:* In many evaluation studies of multipoint protocols, the results are dependent upon several factors, such as membership distribution and network topology. Hence, conclusions drawn from these studies depend heavily upon the evaluation scenarios. This brings about the need for validation test suites. Constructing these test suites can be an onerous and error-prone task if performed manually. In this paper, we propose a method for synthesizing test scenarios automatically for multipoint protocol evaluation. *Worst-case analysis of protocols:* Identifying breaking points that exhibit worst-case performance behaviors may give insight to protocol designers and help in evaluating design trade-offs. The method presented in this paper automates the generation of scenarios in which multipoint protocols exhibit worst and best case behaviors. *Performance benchmarking:* Newly proposed protocols are usually evaluated using scenarios that show performance improvement. However, without systematic evaluation, these studies often (though unintentionally) overlook other scenarios that may be relevant. To alleviate such a problem we propose to integrate stress test scenarios that provide an objective benchmark for performance evaluation. Using our scenario synthesis methodology we hope to contribute to the understanding of better performance benchmarking and the design of more robust protocols.

1.2. BACKGROUND

The design of multipoint protocols has introduced new challenges and problems. Some of the problems are common to a wide range of protocols and applications. One such problem is the *multi-responder* problem,

where multiple members of a group may respond (almost) simultaneously to an event, which may cause a flood of messages throughout the network, and in turn may lead to synchronized responses, causing additional overhead and leading to performance degradation (e.g., the well-known *Ack implosion* problem).

One common technique to alleviate the above problem is the *multicast damping* technique, which employs a *timer suppression* mechanism (TSM). In TSM, a member of a multicast group that has detected loss of a data packet multicasts a request for recovery. Other members of the group, that receive this request and that have previously received the data packet, schedule transmission of a response. In general, randomized timers are used in scheduling the response. While a response timer is running at one node, if a response is received from another node then the response timer is suppressed to reduce the number of responses triggered. Consequently, the response time may be delayed to allow for more suppression. TSM is employed in several multipoint protocols. IP-multicast protocols, e.g., PIM [1] [2], IGMP [3], use TSM on LANs to reduce Join/Prune control overhead. Reliable multicast schemes, e.g., SRM [4], RRM [5], use this mechanism to alleviate *Ack implosion.* Multicast address allocation schemes, e.g., AAP [6], use TSM to avoid an implosion of responses during the collision detection phase. Active services [7] use multicast damping to launch one service agent 'servent' from a pool of servers. TSM is also used in self-organizing hierarchies (SCAN [8]).

We believe TSM is a good building block to analyze as our first end-to-end case study, since it is rich in multicast and timing semantics, and can be evaluated using standard performance criteria. In this study, we examine its worst and best case behaviors in a systematic fashion [1].

Two performance criteria commonly used to evaluate TSM are overhead of response messages and time to recover from packet loss. Depending on the relative delays between group members and the timer settings, the mechanism may exhibit different performance. In this study, our method attempts to obtain scenarios of best case and worst case performance according to the above criteria.

The rest of the paper is organized as follows. Section 2 introduces the protocol and topology models. Section 3 outlines the main algorithm, and Section 4 presents the model for TSM. Sections 5 and 6 present performance analyses for protocol overhead and response time, and Sec-

[1]Such behavior is not protocol specific, and if a protocol is composed of previously checked building blocks, these parts of the protocol need not be revalidated in full. However, interaction between the building blocks still needs to be validated.

tion 7 presents simulation results. Related work is given in Section 8. We present concluding remarks in Section 9.

2. THE MODEL

The model is a processable representation of the system under study that enables automation of our method. Our overall model consists of:

The Protocol Model. We represent the protocol by a finite state machine (FSM) and the overall system by a global FSM (GFSM).

I. FSM model: Every instance of the protocol, running on a single end-system, is modeled by a deterministic FSM consisting of: (i) a set of states, (ii) a set of stimuli causing state transitions, and (iii) a state transition function (or table) describing the state transition rules. A protocol running on an end-system i is represented by the machine $\mathcal{M}_i = (\mathcal{S}_i, \tau_i, \delta_i)$, where $\mathcal{S}_i$ is a finite set of state symbols, τ_i is the set of stimuli, and δ_i is the state transition function $\mathcal{S}_i \times \tau_i \rightarrow \mathcal{S}_i$.

II. Global FSM model: The global state is defined as the composition of individual end-system states. The behavior of a system with n end-systems may be described by $\mathcal{M}_\mathcal{G} = (\mathcal{S}_\mathcal{G}, \tau_\mathcal{G}, \delta_\mathcal{G})$, where $\mathcal{S}_\mathcal{G}$: $\mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$ is the global state space, $\tau_\mathcal{G}$: $\bigcup_{i=1}^{n} \tau_i$ is the set of stimuli, and $\delta_\mathcal{G}$ is the global state transition function $\mathcal{S}_\mathcal{G} \times \tau_\mathcal{G} \rightarrow \mathcal{S}_\mathcal{G}$.

The Topology Model. The topology cannot be captured simply by one metric. Indeed, its dynamics may be complex to model and sometimes intractable. We model the delays using the delay matrix and loss patterns using the fault model. We use a virtual LAN (VLAN) model to represent the underlying network topology and multicast distribution tree. The VLAN captures delay semantics using a delay matrix D, where $d_{i,j}$ is the delay from system i to system j.

The Fault Model. A *fault* is a low level (e.g., physical layer) anomalous behavior that may affect the protocol under test. Faults may include packet loss, system crashes, or routing loops. For brevity, we only consider selective packet loss in this study. Selective packet loss occurs when a multicast message is received by some group members but not others. The selective loss of a message prevents the transition that this message triggers at the intended recipient.

3. ALGORITHM AND OBJECTIVES

To apply our method, the designer specifies the protocol as a global FSM model. In addition, the evaluation criteria are given as input to the method [2]. The algorithm operates on the protocol model and synthesizes a set of 'test scenarios'; protocol events and relations between topology delays and timer values, that stress the protocol according to the evaluation criteria (e.g., exhibit maximum overhead or delay).

The basic algorithm for our method passes through three main steps [3].

1 **The target event:** The algorithm starts from a given event, called the 'target event'. The target event (e.g., sending a message) is identified by the designer based on the protocol evaluation criteria, e.g., overhead.

2 **The search:** Three steps are taken in the search: *(a) Identifying conditions:* The algorithm uses the protocol transition rules to identify transitions necessary to trigger the target event and those that prevent it, these transitions are called *wanted transitions* and *unwanted transitions*, respectively. *(b) Obtaining sequences:* Once the above transitions are identified, the algorithm uses backward and forward search to build event sequences leading to these transitions and calculates the times of these events as follows. **(i) Backward search** is used to identify events preceding the wanted and unwanted transitions, and uses implication rules that operate on the protocol's transition table. Section 4.2 describes the implication rules. **(ii) Forward search** is used to verify the backward search. Every backward step must correspond to valid forward step(s). Branches leading to contradictions between forward and backward search are rejected. Forward search is also used to complete event sequences necessary to maintain system consistency [4]. *(c) Formulating inequalities:* Based on the transitions and timed sequences obtained in the previous steps, the algorithm formulates relations between timer values and network delays that trigger the wanted transitions and avoid the unwanted transitions.

[2]Evaluation criteria may be related to performance or correctness. In this paper we address performance criteria, correctness has been addressed in previous studies [9, 10].

[3]Our algorithm is a variant of the fault-oriented test generation (FOTG) algorithm presented in [10]. It includes the topology synthesis, the backward search and the forward search stages. Here, we describe those aspects of our algorithm that deal with timing and performance semantics.

[4]The role of forward search will be further illustrated in the response time analysis in Section 6.

3 **Task specific solution:** The output of the search is a set of event sequences and inequalities that satisfy the evaluation criteria. These inequalities are solved mathematically to find a topology or timer configuration, depending on the task definition.

We apply our method to two kinds of tasks: **Topology syntehsis** is performed when the timer values are given, and the objective is to identify the delay matrix that produces the best or worst case behavior. **Timer configuration** is performed when the topology or delay matrix is given, and the timer values that cause the best and worst case behavior are to be determined.

4. TIMER SUPPRESSION (TSM)

In this section, we present a simple description of TSM, then present its model, used thereafter in the analysis. TSM involves a request q and one or more responses p. When a system Q detects the loss of a data packet it sets a request timer and multicasts a request q. When a system i receives q it sets a response timer (e.g., randomly), the expiration of which, after duration Exp_i, triggers a response p. If the system i receives a response p from another system j while its timer is running, it suppresses its own response.

4.1. EVALUATION CRITERIA

We use two performance criteria to evaluate TSM: (1) Overhead of response messages, where the worst case produces the maximum number of responses per data packet loss. As an extreme case, this occurs when all potential responders do indeed respond and no suppression takes place. (2) The response delay, where the worst case scenario produces maximum loss recovery time.

4.2. TIMER SUPPRESSION MODEL

Protocol states ($\mathcal{S}$). Following is the state symbol table.

State	Meaning
R	original state of the requester Q
R_T	requester with the request timer set
D	potential responder
D_T	responder with the response timer set

Stimuli or Events. *(1) Sending/receiving messages:* transmitting response (p_t) and request (q_t), receiving response (p_r) and request (q_r). *(2) Timers and other events:* the events of firing the request timer Req and response timer Res and the event of detecting packet loss L.

Notation. Following are the notations used in the transition table.

- An event subscript denotes the system initiating the event, e.g., p_{t_i} is response sent by system i, while the subscript m denotes multicast reception, e.g., p_{r_m} denotes reception of a multicast response by all members of the group if no loss occurs. When system i receives a message sent by system j, this is denoted by the subscript i,j, e.g., $p_{r_{i,j}}$ is system i receiving response from system j.

- The state subscript T denotes the existence of a timer, and is used by the algorithm to apply the 'timer implication' to fire the timer event after the expiration period.

- A state transition has a start state and an end state and is expressed in the form *startState* $\rightarrow$ *endState* (e.g. $D \rightarrow D_T$). It implies the existence of a system in the *startState* (i.e., D) as a condition for the transition to the *endState* (i.e., D_T).

- An *effect* in the transition table may contain state transition and stimulus in the form (*startState* $\rightarrow$ *endState*).*stimulus*, which indicates that the condition for triggering *stimulus* is the state transition. An effect may contain several transitions (e.g., 'Trans1, Trans2'), which means that out of these transitions only those with satisfied conditions will take effect.

Transition Table. Following is the transition table for TSM.

Symbol	Event	Effect	Meaning
loss	L	$(R \rightarrow R_T).q_t$	loss detection causes q_t and setting of request timer
tx_req	q_t	q_{r_m}	transmission of q causes multicast reception of q
rcv_req	q_r	$D \rightarrow D_T$	reception of q causes setting of response timer
res_tmr	*Res*	$(D_T \rightarrow D).p_t$	response timer expiration causes p and change to D
tx_res	p_t	p_{r_m}	transmission of p causes multicast reception of p
rcv_res	p_r	$R_T \rightarrow R$, $D_T \rightarrow D$	reception of p causes timer suppression
req_tmr	*Req*	q_t	request timer expiration causes transmission of q

The model contains one requester Q and several potential responders (e.g., i and j). [5] Initially, the requester Q exists in state R and all potential responders exist in state D. Let t_0 be the time at which Q sends the request q. The request sent by Q is received by i and j at times $d_{Q,i}$ and $d_{Q,j}$, respectively. When the request q is sent, the requester transitions into state R_T by setting the request timer. Upon receiving a request, a potential responder in state D transitions into state D_T, by

[5]Since there is only one requester, we simply use q_t instead of q_{t_Q}, and q_{r_i} instead of $q_{r_{i,Q}}$.

setting the response timer. The time at which an event occurs is given by $t(event)$, e.g., q_{r_j} occurs at $t(q_{r_j})$.[6]

Implication Rules. The following cause-effect implication rules are used by the backward search:

1 Transmission/Reception (**Tx_Rcv**): By the reception of a message, the algorithm implies the transmission of that message – without loss– sometime in the past (after applying the network delays). An example of this implication is $p_{r_{i,j}} \Leftarrow p_{t_j}$, where $t(p_{r_{i,j}}) = t(p_{t_j}) + d_{j,i}$.

2 Timer Expiration (**Tmr_Exp**): When a timer expires, the algorithm infers that it was set Exp time units in the past, and that no event occurred during that period to reset the timer. An example of this implication is $Res_i.(D_i \leftarrow D_{T_i}) \Leftarrow D_{T_i}$, where $t(Res_i) = t(D_{T_i}) + Exp_i$, and Exp_i is the duration of the response timer Res_i.[7]

3 State Creation (**St_Cr**): A state is created from another by reversing the transition rules and going towards the *startState* of the transition. For example, $D_{T_i} \Leftarrow (D_{T_i} \leftarrow D_i)$.

In the following sections we use the above model to synthesize worst case scenarios according to protocol overhead and response time.

5. PROTOCOL OVERHEAD ANALYSIS

In this section, we conduct worst case performance analysis for TSM with respect to the number of responses. We assume no loss of request or response messages until recovery, and that the request timer is high enough that the recovery will occur within one request round [8]. Worst-case analysis aims to obtain scenarios with maximum number of responses per data loss [9]. In this section we present the algorithm to obtain inequalities that lead to worst-case scenarios. These inequalities are a function of network delays and timer expiration values.

Target event and conditions. Since the overhead in this case is measured as the number of response messages, the designer identifies

[6]The time of a state is when the state was first created, so $t(D_{T_i})$ is the time at which i transited into state D_T.
[7]We use the notation $Event.Effect$ to represent a transition.
[8]The case of multiple request rounds is discussed in [11].
[9]For best-case analysis see [11] [12].

the event of triggering a response p_t as the target event, and the goal is to maximize the number of response messages.

The search. Following, we apply to the case study our search algorithm as described in section 3.

- **Identifying conditions:** The algorithm searches for the transitions necessary to trigger the target event, and their conditions, recursively. These are called *wanted transitions* and *wanted conditions*, respectively. The algorithm also searches for transitions that nullify the target event or invalidate any of its conditions. These are called *unwanted transitions*. In our case the target event is the transmission of a response (i.e, p_t). From the transition table described in Section 4.2, the algorithm identifies transition *res_tmr* $[Res.(D_T \rightarrow D).p_t]$ as a *wanted transition* and its condition D_T as a *wanted condition*. Transition *rcv_req* $[q_r.D \rightarrow D_T]$ is also identified as a *wanted transition* since it is necessary to create D_T. The *unwanted transition* is identified as transition *rcv_res* $[p_r.D_T \rightarrow D]$ since it alters the D_T state without invoking p_t.

- **Obtaining sequences:** Using backward search, the algorithm obtains sequences and calculates time values for the following transitions: (1) wanted transition, *res_tmr*, (2) wanted transition *rcv_req*, and (3) unwanted transition *rcv_res*, as follows: (1) To obtain the sequence of events for transition *res_tmr*, the algorithm applies implication rules (see Section 4.2) Tmr_Exp, St_Cr, Tx_Rcv in that order, and we get $Res_i.(D_i \leftarrow D_{T_i}).p_{t_i} \Leftarrow q_{r_i}.(D_{T_i} \leftarrow D_i) \Leftarrow q_{t_Q}$. Hence the calculated time for $t(p_{t_i})$ becomes $t(p_{t_i}) = t_0 + d_{Q,i} + Exp_i$, where t_0 is the time at which q_{t_Q} occurs. (2) To obtain the sequence of events for transition *rcv_req* the algorithm applies implication rule Tx_Rcv, and we get $q_{r_i}.(D_{T_i} \leftarrow D_i) \Leftarrow q_{t_Q}$. Hence the calculated time for $t(q_{r_i})$ becomes $t(q_{r_i}) = t_0 + d_{Q,i}$. (3) To obtain sequence of events for transition *rcv_res* for systems i and j the algorithm applies implication rules Tx_Rcv,Tmr_Exp, St_Cr, Tx_Rcv in that order, and we get $p_{r_{i,j}}.(D_i \leftarrow D_{T_i}) \Leftarrow Res_j.(D_j \leftarrow D_{T_j}).p_{t_j} \Leftarrow q_{r_j}.(D_{T_j} \leftarrow D_j) \Leftarrow q_{t_Q}$. Hence the calculated time for $t(p_{r_{i,j}})$ becomes $t(p_{r_{i,j}}) = t_0 + d_{Q,j} + Exp_j + d_{j,i}$.

- **Formulating Inequalities:** Based on the above wanted and unwanted transitions the algorithm avoids transition *rcv_res* while invoking transition *res_tmr* to transit out of D_T. To achieve this, the algorithm automatically derives the following inequality (see [11] for more details): $t(p_{t_i}) < t(p_{r_{i,j}})$. (1). Substituting expressions

for $t(p_{t_i})$ and $t(p_{r_{i,j}})$ previously derived, we get: $d_{Q,i} + Exp_i < d_{Q,j} + Exp_j + d_{j,i}$. Alternatively, we can avoid the unwanted transition *rcv_res* if the system did not exist in D_T when the response is received. Hence, the algorithm automatically derives the following inequality (see [11] for more details): $t(p_{r_{i,j}}) < t(q_{r_i})$. (2).

Substituting, we get: $d_{Q,i} > d_{Q,j} + Exp_j + d_{j,i}$. [10] Figure 1 (a) and (b) show equations (1) and (2), respectively.

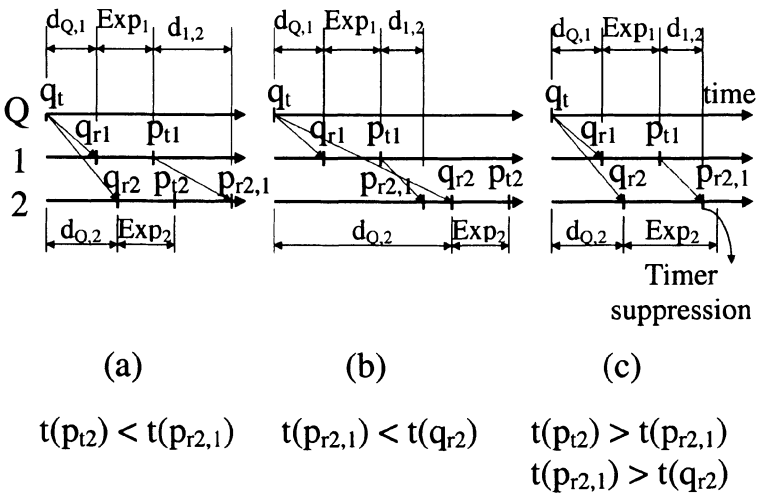

Figure 1 (a) & (b) sequences do not lead to suppression, (c) leads to suppression.

Task specific solutions.

- **Topology synthesis:** Given the timer expiration values or ranges, we want to find a feasible solution for the worst-case delays. A feasible solution in this context means assigning positive values to the delays $d_{i,j} \forall i, j$. In equation (1) above, if we take $d_{Q,i} = d_{Q,j}$ [11], we get: $Exp_i - Exp_j < d_{j,i}$. [12]

[10] Note that equations (1) and (2) are general for any number of responders, where i and j are any two responders in the system.

[11] The number of inequalities (n^2, where n is the number of responders) is less then the number of the unknowns $d_{i,j}$ ($n^2 - n$), hence there are multiple solutions. We can obtain a solution by assigning values to n unknowns (e.g., $d_{Q,i}$) and solving for the others.

[12] These inequalities put a lower limit on the delays $d_{j,i}$, hence, we can always find a positive $d_{j,i}$ to satisfy the inequalities. Note that, the delays used in the delay matrix reflect delays

- **Timer configuration:** Given the delay values or ranges (i.e., bounds), we want to obtain timer expiration values that produce worst-case behavior.

  We can obtain a range for the relative timer settings (i.e., $Exp_i - Exp_j$) using equation (1) above.

The solution for the system of equations given by (1) and (2) above can be solved in the general case using linear programming (LP) techniques (see [11] for more details). Section 7 uses the above solutions to synthesize simulation scenarios. [13]

6. RESPONSE TIME ANALYSIS

In this section, we conduct the performance analysis with respect to the response time. For our analysis, we allow selective loss of a single response message during the recovery phase [14]. In this case, transition rules are applied to only those systems that receive the message.

The algorithm obtains possible sequences leading to the target event and calculates the response time for each sequence. To synthesize the worst case scenario that maximizes the response time, the sequence with maximum time is chosen.

6.1. TARGET EVENT

The response time is the time taken by the mechanism to recover from the packet loss, i.e., until the requester receives the response p and resets its request timer by transitioning out of the R_T state. In other words, the response interval is $t(p_{r_Q}) - t(q_{t_Q}) = t(p_{r_Q}) - t_0$. The designer identifies $t(p_{r_Q})$ as the target time, hence, p_{r_Q} is the target event.

6.2. THE SEARCH

We present in detail the case of single responder, then discuss the multiple responders case.

over the multicast distribution tree. In general, these delays are affected by several factors including the multicast and unicast routing protocols, tree type and dynamics, propagation, transmission and queuing delays. One simple topology that reflects the delays of the delay matrix is a completely connected network where the underlying multicast distribution tree coincides with the unicast routing.

[13]Note, however, that it may not be feasible to satisfy all these constraints, due to upper bounds on the delays for example. In this case the problem becomes one of maximization, where the worst-case scenario is one that triggers maximum number of responses per packet loss. This problem is discussed in [11].

[14]Without loss of response messages this problem becomes one of maximizing the round trip delay from the requester to the first responder.

- **Backward search:** As shown in Figure 2 (a), the backward search starts from p_{r_Q} and is performed over the transition table (in Section 4.2) using the implication rules in Section 4.2, yielding [15]:

  $D_j.p_{r_Q}.(R_Q \leftarrow R_{T_Q}) \Leftarrow p_{t_j}.(D_j \leftarrow D_{T_j}).Res_j.R_{T_Q} \Leftarrow q_{r_j}.(D_{T_j} \leftarrow D_j).R_{T_Q}$

  At which point the algorithm reaches a branching point, where two possible preceding states could cause q_{r_j}:

  - The first is transition *loss* $[D_j.q_{t_Q}.(R_{T_Q} \leftarrow R_Q)]$ and since the initial state R_Q is reached, the search ends for this branch.
  - The second is transition *req_tmr* $[D_j.Req_Q.q_{t_Q}.R_{T_Q}]$. Req_Q indicates the need for a transition to R_{T_Q}, and the search for this last state yields eventually $D_j.q_{t_Q}.(R_{T_Q} \leftarrow R_Q)$.

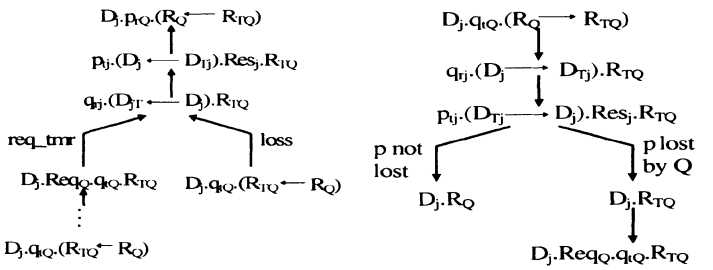

Figure 2 Backward search for response time analysis.

- **Forward search:** The algorithm performs a forward search and checks for consistency of the GFSM. The forward search step may lead to contradiction with the backward search, causing rejection of that branch. For example, as shown in Figure 2 (b), one possible forward sequence from the initial state gives:

  $D_j.q_{t_Q}.(R_Q \rightarrow R_{T_Q}) \Rightarrow q_{r_j}.(D_j \rightarrow D_{T_j}).R_{T_Q} \Rightarrow p_{t_j}.(D_{T_j} \rightarrow D_j).Res_j.R_{T_Q}$

  The algorithm then searches two possible next states:

[15]The GFSM may be represented by composition of individual states (e.g., $State_1.State_2$ or $transition_1.State_2$).

- If p_{t_j} is not lost, and hence causes p_{r_Q}, then the next state is $D_j.R_Q$. But the original backward search started from $D_j.q_{t_Q}.Req_Q.R_{T_Q}$ which cannot be reached from $D_j.R_Q$. Hence, we get contradiction and the algorithm rejects this sequence.
- If the response p is lost by Q, we get $D_j.R_{T_Q}$ that leads to $D_j.Req_Q.q_{t_Q}.R_{T_Q}$. The algorithm logs this feasible sequence.

Calculating the time for each feasible sequence, the algorithm identifies the latter sequence as one of maximum response time.

For **multiple responders**, the algorithm automatically explores the different possible selective loss patterns of the response message. The search identifies the sequence with maximum response as one in which only one responder triggers a response that is selectively lost by the requester. To construct such a sequence, the algorithm creates conditions and inequalities similar to those formulated for the best-case overhead analysis with respect to number of responses (see [11]).

7. SYSTEMATIC SIMULATION

We have conducted a set of simulations for the scalable reliable multicast (SRM) [4] based on our worst-case analysis. We tied our method to the network simulator (NS). The output of our method, in the form of inequalities (see Section 5), is solved using a mathematical package (LINDO). The solution, in terms of a delay matrix, is then used to generate the simulation topologies for NS automatically.

We measured the number of responses triggered for each data packet loss. We have conducted two sets of simulations, each using two sets of topologies. The simulated topologies included topologies with up to 200 nodes. The first set of *stress* topologies was generated according to the overhead analysis presented in this paper. The second set of *random* topologies was generated by the GT-ITM topology generator [13].

The first set of simulations was conducted for the SRM deterministic timers [16]. The results of the simulation are shown in Figure 3. The number of responses triggered for all the *stress* topologies was $n - 1$, where n is the number of nodes in the topology (i.e., no suppression occurred). For the *random* topologies, the number of responses triggered was almost 20 responses in the worst case.

[16]SRM response timer values are selected randomly from the interval $[D_1.d_r, (D_1 + D_2).d_r]$, where d_r is the estimated distance to the requester, and D_1, D_2 depend on the timer type. For deterministic timers $D_2 = 0$ and $D_1 = 1$.

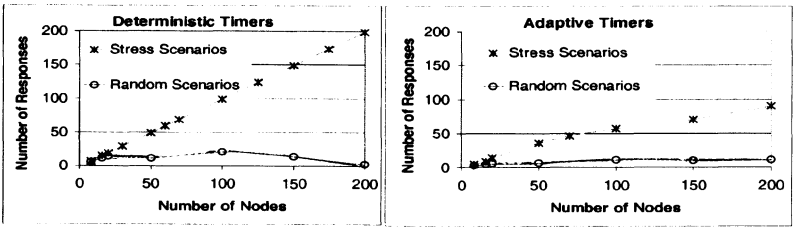

Figure 3 Simulation results for deterministic and adaptive timers over *stress* and *random* topologies.

Using the same two sets of topologies, the second set of simulations was conducted for the SRM adaptive timers [17]. The results are given in Figure 3. For the *stress* topologies almost 50% of the nodes in the topology triggered responses. Whereas *random* topologies simulation generated almost 10 responses in the worst case.

These simulations illustrate how our method may be used to generate consistent worst-case scenarios in a scalable fashion. It is interesting to notice that worst-case topologies generated for simple timers also experienced substantial overhead for more complicated adaptive timers. It is also obvious that *stress* scenarios are consistent when used to compare different mechanisms; the performance gain for adaptive over deterministic timers is very clear under *stress* scenarios.

8. RELATED WORK

Verification systems typically aim to detect violation of well-defined properties (e.g., safety). In general, the two main approaches for protocol verification are theorem proving and reachability analysis [14]. Theorem proving systems define a set of axioms and relations to prove properties.

[17] Adaptive timers adjust their interval based on the number of duplicate responses received and the estimated distance to the requester.

These systems are useful in many applications. However, they do not synthesize network topologies and do not address performance issues per se. Reachability analysis algorithms [15] try to inspect reachable protocol states, and suffer from the 'state space explosion' problem. Reduced reachability analysis has been used in the verification of cache coherence protocols [16], using a global FSM model. We adopt a similar FSM model and extend it for our approach in this study. However, our approach differs in that we address end-to-end protocols, that encompass rich timing, loss and performance semantics.

In [10] we developed a fault-oriented test generation (FOTG) for multicast routing. In [10], FOTG was used to study correctness of a multicast routing protocol on a LAN. Here, we extend FOTG to study performance of end-to-end multipoint mechanisms. In [9] we proposed a simulation-based stress testing framework based on heuristics. However, that method does not provide automatic topology generation, nor does it address performance issues. The VINT [17] tools provide a framework for Internet protocols simulation with a library of protocols and a set of validation test suites. However, it does not provide a generic tool for generating these tests automatically. Work in this paper is complementary to such studies, and may be integrated with network simulation tools as we do in Section 7.

9. CONCLUSION AND FUTURE WORK

We have presented a methodology for scenario synthesis for performance evaluation of multipoint protocols. We used a virtual LAN model to represent the underlying network topology and an extended global FSM model to represent the protocol mechanism. We adopted the fault-oriented test generation algorithm for search, and extended it to capture timing/delay semantics and performance issues for end-to-end multipoint protocols. Our method was applied to performance evaluation of the timer suppression mechanism; a common building block for various multipoint protocols. Two performance criteria were used for evaluation of the worst and best case scenarios; the number of responses per packet loss, and the response delay. Simulation results illustrate how our method can be used in a scalable fashion to test and compare reliable multicast protocols. We hope that similar approaches may be used to identify and analyze other protocol building blocks.

Our future work includes applying our methodology to sensor networks. These networks, similar to ad-hoc networks, assume dynamic topologies, lossy channels, and deal with stringent power constraints, which differentiates their protocols from Internet protocols [18]. Possible

research directions in this respect include: (i) Extending the topology representation or model to capture dynamics, where delays vary with time. (ii) Defining new evaluation criteria that apply to the specific problem domain, such as power usage. (iii) Investigating the algorithms and search techniques that best fit the new model or evaluation criteria.

References

[1] D. Estrin, D. Farinacci, A. Helmy et al. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification. *RFC 2117*, March 1997.

[2] D. Estrin, D. Farinacci, A. Helmy et al. Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification. *Proposed RFC*, September 1996.

[3] W. Fenner. Internet Group Management Protocol, Version 2. *Internet-Draft*, November 1997.

[4] 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*, November 1996.

[5] R. Govindan, H. Yu, and D. Estrin. Large-scale weakly consistent replication using multicast. *USC-CS-TR 98-682*, September 1998.

[6] M. Handley. The Address Allocation Protocol. *Internet-Draft*, August 1998.

[7] E. Amir et al. An active service framework and its application to real-time multimedia transcoding. *ACM SIGCOMM'98*, September 1998.

[8] A. Reddy, D. Estrin, and R. Govindan. Fault Isolation in Multicast Trees. *ACM SIGCOMM*, August 2000.

[9] A. Helmy and D. Estrin. Simulation-based STRESS Testing Case Study: A Multicast Routing Protocol. *IEEE MASCOTS*, July 1998.

[10] A. Helmy, D. Estrin, and S. Gupta. Fault-oriented test generation for multicast routing protocol design. *FORTE/PSTV*, November 1998.

[11] A. Helmy et al. Performance Evaluation of Multipoint Protocols Using Systematic Scenario Synthesis. *USC-CS-TR 00-726*, March 2000.

[12] Ahmed Helmy. Systematic Test Synthesis for Multipoint Protocol Design. *USC-CS-TR 99-716, Ph.D. Dissertation*, August 1999.

[13] K. Calvert, M. Doar, and E. Zegura. Modeling Internet Topology. *IEEE Communications*, June 1997.

[14] E. Clarke and J. Wing. Formal Methods: State of the Art and Future Directions. *ACM Wkshp on Strategic Directions in Computing Research*, December 1996.

[15] F. Lin, P. Chu, and M. Liu. Protocol Verification using Reachability Analysis. *Computer Communication Review, Vol. 17, No. 5*, 1987.

[16] F. Pong and M. Dubois. Verification Techniques for Cache Coherence Protocols. *ACM Computing Surveys*, March 1996.

[17] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy et al. Advances in Network Simulation. *IEEE Computer*, May 2000.

[18] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. *ACM MobiCom*, August 1999.