

NOTE: This is a preliminary release of an article accepted by the ACM Transactions on Modeling and Computer Simulation. The definitive version is currently in production at ACM and, when released, will supersede this version.

Copyright (C) 2003 by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works, requires prior specific permission and/or a fee. Permissions may be requested from:

Publications Dept, ACM Inc.,  
1515 Broadway, New York, NY 10036  
USA fax +1 (212) 869-0481, or  
permissions@acm.org.

# MAYA: Integrating Hybrid Network Modeling to the Physical World

JUNLAN ZHOU

ZHENGRONG JI

MINEO TAKAI

and

RAJIVE BAGRODIA

University of California, Los Angeles

---

The flourish of large-scale network applications across the Internet and or MANET has raised a challenge to network modeling environments that support experimentation and analysis of close interactions between real applications and network dynamics. To facilitate such experimentations, this paper presents *MAYA*, a multi-paradigm network modeling framework including discrete event models, analytical models and physical network interfaces, together with its illustrative implementation using QualNet, fluid flow TCP model and physical network interface. *MAYA* framework allows users to interface simulated networks directly with physical networks, while attaining real-time constraints even for large-scale networks by incorporating above multi-paradigm network modeling techniques. It also gives user the flexibility to emulate applications on nodes in both real and simulated networks. Experiments are conducted to validate the interoperation of QualNet and fluid flow model, to examine the performance of *MAYA* as well as to evaluate the optimization techniques, namely interleaved execution of fluid flow model and causality-preserve realtime synchronization relaxation. Experimental results indicate that *MAYA* is a scalable and extensible solution to modeling of close interactions between real application and network dynamics.

Categories and Subject Descriptors: I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*; I.6.7 [**Simulation and Modeling**]: Simulation Support Systems—*Environments*

General Terms: Network modeling and simulation, Realtime simulation

Additional Key Words and Phrases: QualNet, Fluid flow model, Physical network interface

---

## 1. INTRODUCTION

Fast networks and low-cost processors are enabling the advent of a new wave of distributed applications ranging from client-server web services to highly distributed peer-to-peer computing environments. Unlike traditional standalone applications, developing distributed applications in various domains is significantly harder due to

---

This work is supported by the US Department of Defense/DARPA under Contract N66001-00-1-8937 “Maya: Next Generation Performance Prediction Tools for Global Networks”.

Author’s address: Junlan Zhou, Computer Science Department, UCLA, Los Angeles, CA 90095.

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 2003 ACM 0000-0000/2003/0000-0001 \$5.00

the fundamental characteristics of the underlying network, i.e. failure, heterogeneity, errors and so on. It is essential to be aware of the dynamics of the underlying network when developing a distributed application and to validate major design decisions prior to the actual deployment. However, a real test bed is generally not available, as the scale of the network on which the distributed applications are executed can be arbitrarily large.

The combination of real application codes and models for the underlying networks offers a promising solution to assess the impact of network dynamics on the performance of distributed applications. The network models mimic the behaviors of the underlying networks and provide programmers with a synthetic configurable environment to conduct repeatable experiments with the distributed applications. There are numerous challenges in building such a comprehensive environment that can satisfy the disparate needs of distributed applications. First, it must resemble the target network as closely as possible. Second, it needs to run in real time as the validity of the results depends on its ability to accommodate real time constraints. Third, it must be scalable in order to enable study and evaluation of distributed applications executing on large networks with high throughput of both foreground and background traffic. Finally, it must be transparent to applications and support direct evaluation of precompiled applications.

Many previous efforts have addressed similar objectives. Studies in [Zheng and Ni 2002; Daniel and Kurt ] simulate links among nodes in the target wireless network with regular computers and higher-speed network devices while real codes are running at high layers. A module is developed between the network layer and device driver in each computer, to introduce effects such as propagation delay, packet loss or bandwidth throttling, to emulate dynamic network conditions. As this approach requires the same number of networked machines or devices as in the target network, the amount of hardware required by these studies grows proportionally with the size of the target network, which limits its scalability. The works presented in [Davies et al. 1995], [Noble and et al 1997], [Ke et al. 2000], [Simmonds and Unger 2001] and [Fall 1999] utilize network simulators as their cores and develop an interface in the simulators for real world traffic to traverse the simulated network and interact with the simulated traffic. Using a parallel discrete event simulation (PDES) kernel, the studies in [Simmonds and Unger 2001] target construction of wide area networking environments. However, the runtime performance of discrete event models is highly dependent on simulated network size and the amount of traffic. This inherently limits scalability since the PDES techniques can speed up simulation executions only by the number of processors used.

Therefore, out of the many challenges in building an emulation environment for distributed applications, scalability has become a major concern as the number of distributed applications that target the Internet has grown significantly. Many research efforts have been dedicated to analysis of not only the statistical behaviors of traffic generated by Internet-scale systems or applications [Erramilli et al. 1996; Crovella and Bestavros 1997], but also the expected transient behaviors of networks supporting such traffic flows [Misra et al. 2000; Liu et al. 2003]. The analytical models emerging from these efforts predict the network dynamics quite accurately and scale well to a large number of flows, and thus are of direct benefit in developing

a framework for emulating distributed applications. There have been some attempts to integrate fluid based analytical models with discrete event models [Yung et al. 2001; Nicol 2001]. Note, however, that the application of analytical models is typically limited for specific network and traffic types. Therefore, the framework within which different types of analytical models can be integrated needs to be flexible to provide a wide range of networking scenarios needed for evaluation of distributed applications.

This paper presents *MAYA*, a multi-paradigm network modeling framework for studying close interactions among real distributed applications and underlying target network dynamics. *MAYA* provides an environment where many instances of three distinct paradigms (i.e., discrete event models, analytical models, and interfaces to the physical networks) can be integrated seamlessly and interact with distributed applications. In fact, while analytical models may run faster than the corresponding discrete event models, they have unique real time synchronization problems when integrated with a physical network interface due to its long turnaround time to advance its virtual time. Another real time synchronization issue is raised by bursty traffic from real network, which may cause the discrete event simulation to temporarily lag behind real time. This paper discusses the issues involved in integrating these different paradigms, and presents two techniques to improve the time synchronization among instances of different paradigms. The first technique utilizes interleaved execution of analytical models to hide the costly turnaround time of analytical models from other modeling paradigms. The second technique employs causality-preserving realtime synchronization relaxation to alleviate the temporal lag between the simulated and physical networks without losing modeling fidelity, by allowing simulation to run further ahead of real time whenever real traffic burst occurs. Experimental results show that the proposed optimizations improve the accuracy of time synchronization significantly, with *MAYA* delivering data packets to the distributed applications in a timely manner even if the underlying network is highly heterogeneous and scales substantially.

The remainder of this paper is organized as follows. Section 2 and 3 gives an overview of the *MAYA* framework and describes the approach to integrate heterogeneous paradigms into this unified framework. Section 5 validates the mixture of QualNet and fluid flow model, while section 6 shows its performance gains. Section 6 also analyzes *MAYA*'s performance and presents two optimizations that significantly improve the performance of *MAYA*. Last by not the least are conclusions and future works.

## 2. MAYA ARCHITECTURE

The goal of *MAYA* is to provide a scalable and extensible network modeling framework incorporating disparate network models while interfacing to real networks and applications; thus a layered approach is followed in the conceptual design of the framework where existing and future modeling techniques can be easily integrated. The architectural overview of *MAYA* that captures the essential interoperations among different components is depicted in Figure 1. *MAYA* employs a three-tier approach, and leverages from existing tools or libraries, where possible.

### Top tier:

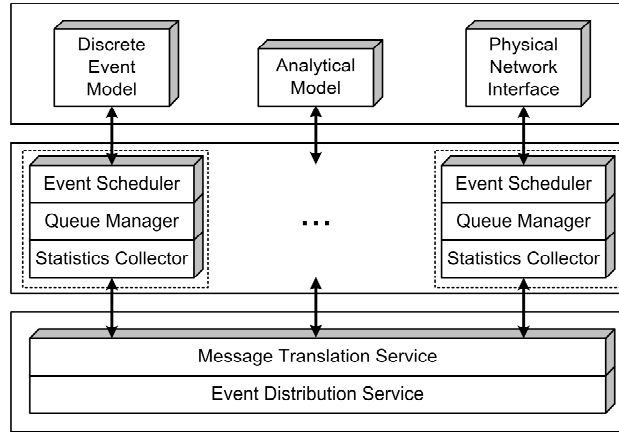


Fig. 1. Overview of MAYA Architecture

The top tier of *MAYA* provides the building blocks for constructing underlying networks for emulating distributed applications. Three types of components are incorporated at this tier, namely discrete event model, analytical model and the physical network interface, corresponding to the three modeling paradigms. In contrast to the other components, the physical network interface captures and creates real IP packets, through which physical networks can be attached to simulated networks. Multiple instances of these components can be juxtaposed; each corresponds to a portion of the underlying networks on which distributed applications can be executed. The combination of modeling paradigms offer users the flexibility to tradeoff fidelity with scalability of network modeling by abstracting portions of the networks with analytical models. The top tier components exchange global events via services provided at lower tiers to model packets flowing among different portions of the networks. Each generated event is tagged with a global timestamp and carries a packet to be forwarded.

**Middle tier:**

The middle tier enables interoperability among the heterogeneous components at the top tier. It handles incoming and outgoing global events, ensuring that top tier components exchange meaningful information despite their semantic differences. When needed, it is also responsible for scheduling global events and synchronizations with a wall clock time, as required by emulation of distributed application. The components providing the above functionalities include event schedulers, statistics collectors, and queue managers. Two types of messages are exchanged among the components serving different modeling instances at the top tier: timestamped events that correspond to packets exchanged between the modeling paradigms, and synchronization messages that contain time critical information for performance optimization, which needs to be delivered as soon as possible.

**Bottom tier:**

The bottom tier provides basic services such as time synchronization and global event distribution among the disjoint components at upper tiers. There have been

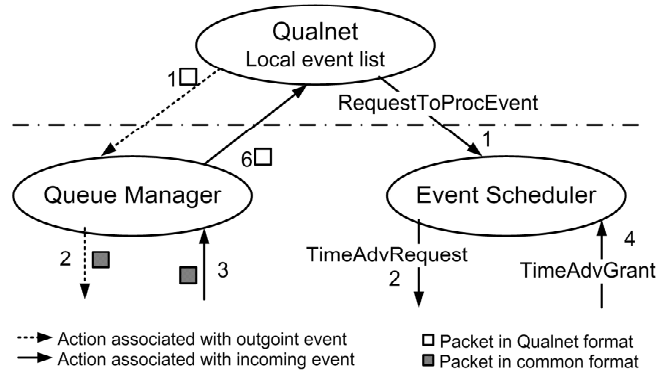


Fig. 2. Integration of QualNet

many research studies on event distribution and time management among processes in distributed environments, such as the High Level Architecture proposed by the Department of Defense(DoD)[HLA ], PARSEC[Bagrodia et al. 1998], RTI-KIT[Fujimoto and et al 2001] etc, all of which conceptually can be leveraged by *MAYA*. RTI-KIT is chosen for its open design and good performance in high-speed LAN environments as well as SMP machines. A dynamic simulation backplane [Riley et al. 2001] is utilized to allow components at the top tier to exchange protocol information and dynamically convert packets between native formats and a common format. No time ordering constraints will be enforced on performance optimization messages exchanged among upper tiers, which will be delivered to their destination once received.

### 3. IMPLEMENTATIONS

There are many simulation and modeling tools falling into the paradigms of either discrete event model (such as *NS-2*[NS2 ], *GloMoSim*[GLOMOSIM ], *QualNet*[QUALNET ], *OPNET*[OPNET ], *PDNS*[Riley et al. 1999] *DaSSF*[DaSSF ]etc.) or analytical model (such as fluid flow based TCP model[Misra et al. 2000] and Markovian model[Abouzeid et al. 2000]). The current implementation of *MAYA* includes one tool from each of the proceeding groups to demonstrate multi-paradigm integration: the QualNet discrete event simulator and fluid flow model of TCP, with an implementation of a Physical Network Interface in Linux. The rest of the section describes how each of them has been integrated into the framework.

#### 3.1 Integration of discrete event models

Figure 2 shows the operations of QualNet and how it interacts or exchanges events with other components at the top tier of *MAYA* via the services provided by the components at the lower tier, namely, the queue manager and event scheduler.

The queue manager exports and imports global events, employing the utilities available in the backplane and RTI-KIT at the bottom tier. It converts a packet between its native format used by a specific tool like QualNet and the common format, and fills up the extra data fields of a native format packet in the global event appropriately before delivering the packet to QualNet. The event scheduler ensures

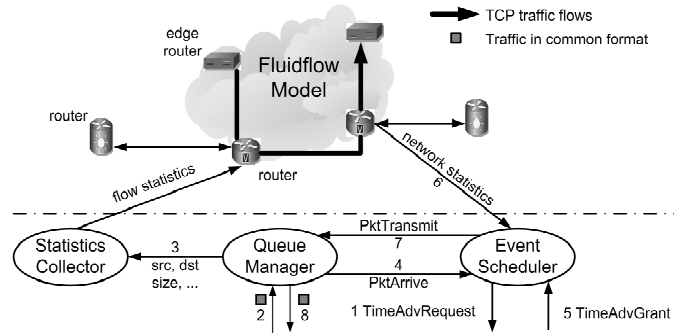


Fig. 3. Integration of the Fluid Flow Model

that QualNet processes all events in strict timestamp order. Before scheduling a local event, QualNet consults the event scheduler (see Figure 2), which in turn makes a time advancement request to the RTI-KIT. Inbound global events with timestamps smaller than the requested time are passed to the queue manager by the RTI-KIT. After all the eligible global events are delivered, the RTI-KIT grants the request and guarantees that no more events earlier than the granted time will arrive later. The event scheduler then informs the queue manager to forward all received packets to QualNet, allowing QualNet to schedule the outstanding event with the smallest timestamp for processing.

One challenge to be addressed by the event scheduler is that QualNet must tightly synchronize with real time during its execution. Suppose another top-tier component is tightly synchronized with real time. As long as the event scheduler ensures QualNet is time constrained by that component, QualNet will be transitively synchronized with real time. In this way, direct wall clock time synchronization of QualNet is avoided, resulting in better efficiency. A typical candidate for bootstrapping the real time synchronization is the physical network interface, since it needs to timestamp the real IP packets with the wall clock anyway.

### 3.2 Integration of analytical models

The fluid flow based TCP model[Misra et al. 2000] is the representative analytical model integrated into the initial implementation of *MAYA*<sup>1</sup>. As its name suggests, the fluid flow model treats TCP sessions as fluids. It derives a set of ordinary differential equations (ODE) to model the rates of traffic changes and queuing process at routers. It excels in modeling wired backbone networks with high volumes of traffic. Unlike a discrete event model or physical network interface, there are no notions of “packet” or “event” in the fluid flow model. A queue manager, statistics collector, and event scheduler have been developed at the middle tier to enable the fluid flow model to process global events and interact with its peers.

Figure 3 shows how packets originated from a physical network interface or discrete event model traverse a wired backbone network represented by the fluid flow

<sup>1</sup>Modeling limitations to fluid flow TCP model shall not be considered the limitation to *MAYA* because other analytical models could be integrated similarly when required to model a target network.

model. Information (such as packet size etc.) of the packets entering the wired network will be extracted by the queue manager and used by the statistics collector to compute statistics of the traffic flows, which is the input to the ODE solver of the fluid flow model. Since the amount of traffic entering the wired network varies over time, the statistics of traffic flows are averaged within a certain time interval (called time averaging interval) and fed into the ODE solver periodically. As an output, the ODE solver produces network statistics, such as queue length, TCP window size and so on, for the next time averaging interval. The results will be used to estimate the delay of packets traveling across the wired network during that time interval.

The queue manager handles importing, exporting and queuing of global events. It extracts information such as source IP address, destination IP address, packet size, etc., and feeds it to the statistics collector. The event scheduler maintains a local event list comprising two types of events: those related to packet arrivals or transmissions and those responsible for periodically invoking the ODE solver of the fluid flow model. An event of “PktArrive” is triggered via reception of a global event carrying a packet. To process the “PktArrive” event, the event scheduler estimates the delay of the packet indicated by the “PktArrive” event to traverse the wired network based on the network statistics produced by the ODE solver, and schedules the transmission of the packet accordingly, as if the packet has traveled across the wired network. The event scheduler will not be able to process an event until its time advances to the event’s timestamp, via which the fluid flow model is synchronized with its peers.

### 3.3 Integration of physical network interfaces

Contemporary large-scale networks usually comprise multiple LANs or even autonomous systems (AS) interconnected via gateways. When emulating distributed applications, it is logical and convenient to partition the underlying networks into LANs or autonomous systems, each of which is either physical or modeled by an instance of discrete event or analytical model. The physical network interface should reside at the gateways of the physical networks to inter-connect the physical networks with the modeled portions of the underlying network when emulating distributed applications.

As shown in Figure 4, the physical network interface has two modules, a sending interface and a receiving interface. The raw packet receiver of the sending interface captures the real IP packets originated from distributed applications running on machines in the physical networks using the Berkeley Packet Filter (BPF)[McCanne and Jacobson 1993]. The timestamp of a captured packet is set to the wall clock time when captured by the raw packet receiver. The timestamped packet is sent to the queue manager for immediate event delivery. The event scheduler of the sending interface tries to synchronize the sending interface with the wall clock time by requesting its virtual time to be equal to current wall clock time. As the real packet can come at any time, the raw packet receiver runs as a separate thread polling the BPF kernel for newly captured packets.

The event scheduler of the receiving interface keeps polling RTI-KIT by making time advancement requests. Its queue manager will import the events received from the lower tier during the time advancement requesting process. When the

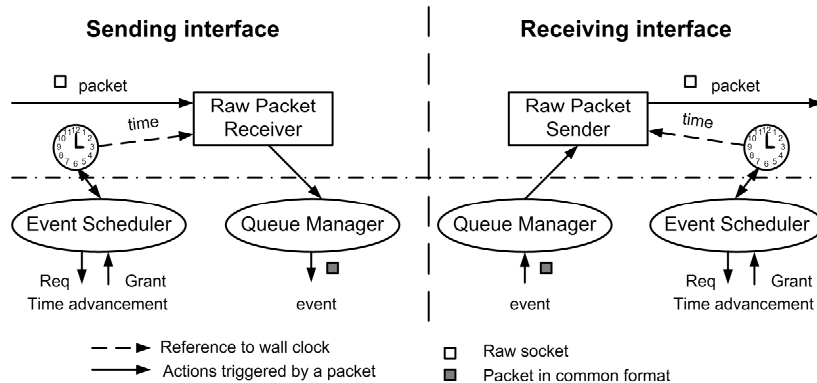


Fig. 4. Integration of Physical Network Interfaces

time advancement request is granted, the event scheduler will instruct the queue manager to convert these events into timestamped real packets and ship them to the raw packet sender. The raw packet sender will then send the packets at the exact time specified by the timestamp. The timer for the Linux operating system running on the Intel PC has a granularity of  $10ms$ , which may not be sufficiently accurate. Thus, the raw packet sender is implemented as a separate thread that polls the system clock between times when it should send packets, which improves accuracy down to microseconds.

A look-ahead[Fujimoto 1998] is needed at the sending interface in order for the discrete event model and the analytical model to run faster than real time. As *MAYA* aims to emulate distributed applications running on top of large-scale worldwide networks, the queuing and propagation delays between autonomous systems can range from a few milliseconds to tens of milliseconds, which makes them a good choice for look-ahead. Even longer delays may be seen for satellite links. As a matter of fact, similar look-ahead values can be determined for all the components at the top tier. However, an exception can be made for the receiving interface module of the physical network interface, because it does not time-constrain[Fujimoto 1998] the sending interface module, and other components at the top tier.

#### 4. CONFIGURING UNDERLYING NETWORK FOR EMULATING DISTRIBUTED APPLICATION

Figure 5 illustrates how a target network is partitioned and mapped to top-tier components of Maya for emulating distributed applications. In the example, the network is partitioned into virtual subnets and physical subnets. A virtual subnet is modeled by a discrete event simulator (such as QualNet or PDNS in this example) or an analytical model. The choice of mapping is based on a combination of factors that may include applicability, desired level of fidelity, and execution speed. A physical subnet comprises regular computers (referred to as operational nodes) inter-connected by network devices, on which the distributed applications reside. The physical network interfaces, which execute on the gateways of physical subnets, allow live traffic generated by the distributed applications to traverse virtual subnets

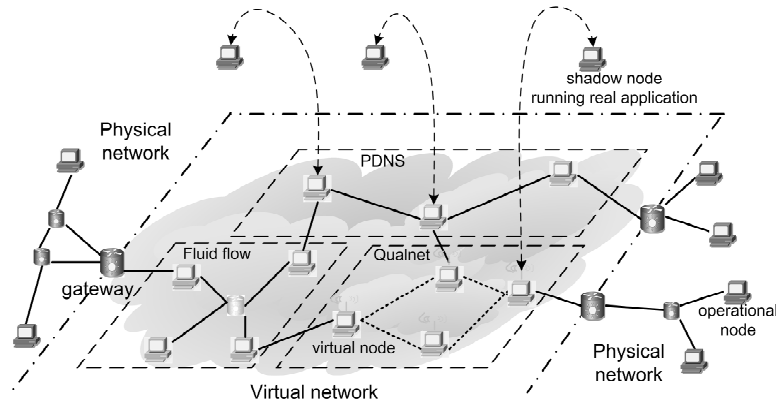


Fig. 5. An Example of Configuring Underlying Network

and be subjected to the network conditions present there.

It is possible that certain real application running on a operational node in a physical subnet needs to communicate with another application which ideally runs at a virtual node in the virtual subnet. *MAYA* allows user to run the real application on another physical node which acts as a shadow of the virtual node. The incoming traffic to the virtual node will be forwarded to the shadow node outside the simulation while the outgoing traffic from the shadow node will be forwarded to the virtual node in the simulation as if the traffic comes out of the virtual node. Because the *MAYA* framework is designed to be generic for any existing and future real networking applications, it utilizes the physical network interface as the bridge between the communication of a virtual node and its shadow node. However, the shadowing of virtual node results in more stringent realtime synchronization because the virtual node has to synchronize with the shadow node tightly without much lookahead to be explored by *MAYA*. This configuration often causes simulation to temporarily run slower than real time especially when real application running on a shadow node generates very bursty traffic. A performance optimization technique that alleviates this problem will be presented later in this paper.

## 5. VALIDATION OF MIXED MODELING PARADIGMS

The validity of fluid flow TCP model has been demonstrated by its original authors in [Misra et al. 2000]. Nevertheless, before any claims can be made on performance gains of integrating the fluid flow model in the *MAYA*, it must be verified that the mixture of QualNet and fluid flow model (refer to as mixed modeling paradigms) will not affect the validity of network modeling results. To demonstrate the correctness of this mixed modeling paradigms, we start with a simple network topology with 6 nodes, as shown in Figure 6. The network includes two sets of traffic flows: from node 0 to 1 and from node 4 to 5. Link capacities of 40Mbps are assigned for all links except the one between nodes 2 and 3, which is assigned a capacity of 2Mbps. Thus the link between nodes 2 and 3 will be the bottleneck link. A propagation delay of 5ms is assumed in all the experiments. The subnet defined by nodes 4, 2, 3 and 5 is modeled using the fluid flow model, while nodes 0 and 1 will be simulated

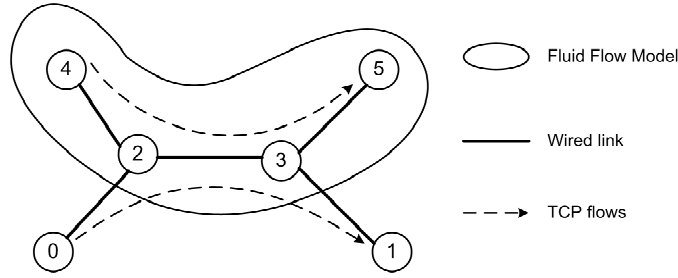


Fig. 6. Network Topology for Validation of Mixed Modeling Paradigms

by QualNet. Conceptually there is an end-to-end path connecting node 0 and 1 with an estimated delay based on traffic in the fluid flow model, and packets may be dropped as they are transmitted across the network portion modeled by fluid flow model.

Validation experiments are conducted to compare the instantaneous and average queue lengths obtained in the pure fluid flow model implemented in MATLAB, with the results of equivalent systems simulated using high fidelity discrete event simulation models in QualNet, and with those predicted by the mixed modeling paradigms. For the first experiment, 20 TCP flows are set up from node 4 to 5 while 1 TCP flow is established from node 0 to node 1. RED is used as the AQM policy, with a minimum threshold of 150 packets before the router starts dropping packets and a maximum dropping probability of 10% at maximum threshold of 200 packets. The simulation runs for 200 seconds, while all traffic flows start at time 0s and continue through the entire simulation. As shown in Figure 7(a) and 7(b), the results of instantaneous and average queue lengths at node 2 are comparable across the modeling methods, with some differences in the initial phases of network operations. This divergence is due to the fact that the fluid flow model does not model the effect of slow start in TCP protocol. This experiment also measures the end-to-end delays for the single TCP traffic flow from node 0 to 1 in pure QualNet simulation and QualNet plus fluid flow model simulation. Again the results in Figure 7(c) indicate good matches between those obtained by mixed modeling paradigms and those obtained via QualNet only.

The second validation experiment exploits the same network topology as in the previous experiment, but increases the number of flows and introduces some traffic dynamics: initially, the network has 15 TCP flows each from nodes 4 to 5 and nodes 0 to 1; from 100s to 200s, there are only 5 TCP flows from node 4 to 5, as well as 5 TCP flows from node 0 to 1 left. The results of instantaneous and average queue length at node 2 have been shown in Figure 8. Again as seen from the figure, the results match quite well except the initial slow start phase, reflecting the change to the flow dynamics in the midst of the simulation.

Besides the disparity in execution speeds, QualNet aims at providing detailed discrete event network simulation, while fluid flow TCP predicts network statistics such as queue lengths and queuing delays of packets, especially when large aggregation of TCP flows are directed through the same wired communication link(s). The results obtained from above experiments have demonstrated that integrating

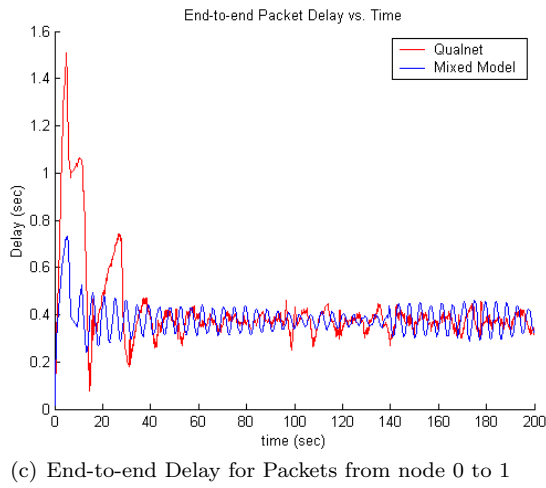
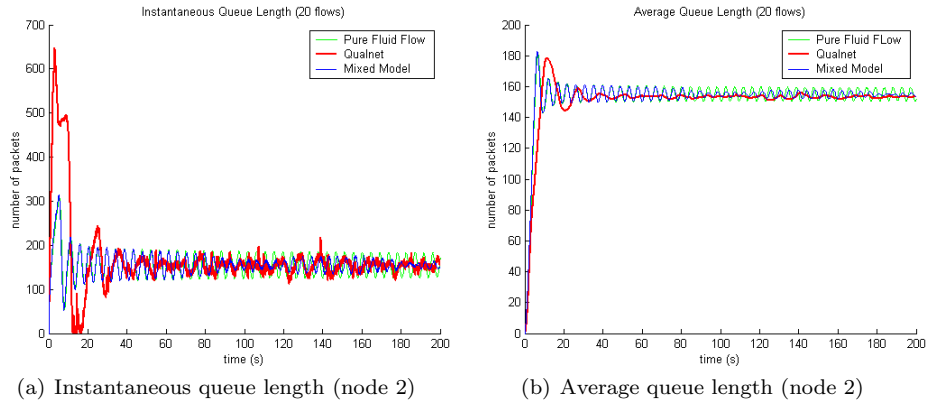


Fig. 7. Statistics of First Validation Experiment

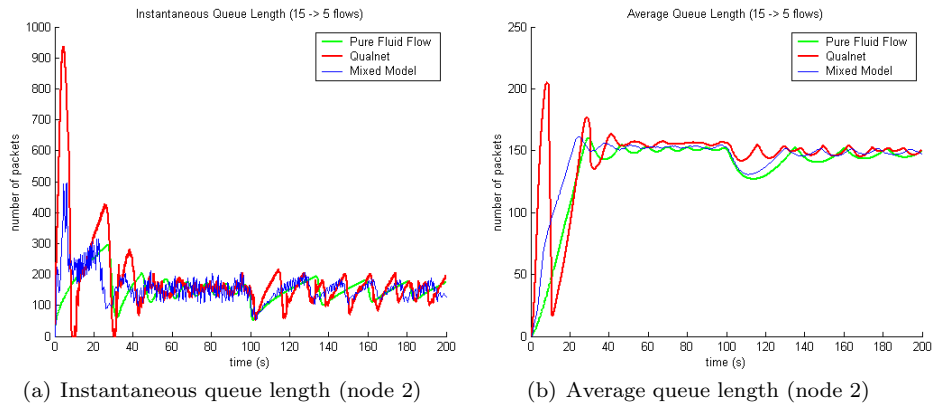


Fig. 8. Statistics of Second Validation Experiment

QualNet and fluid flow TCP model into the same *MAYA* framework will not affect the accuracy of the mixed modeling paradigms. The integration of these two distinct network modeling paradigms will greatly improve the scalability of *MAYA* framework, as described in the next section.

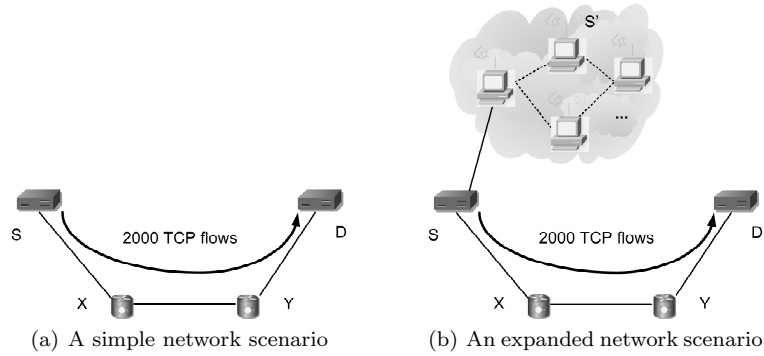


Fig. 9. Network Modeling Scenarios(Non-Realtime)

## 6. PERFORMANCE ANALYSIS AND OPTIMIZATIONS

This section presents a set of experiments to evaluate the performance of *MAYA*. First, it demonstrates the performance advantage of *MAYA* over the pure discrete event simulator based framework. It then evaluates the ability of *MAYA* to conform to real time constraints so as to provide a platform to emulate distributed applications. Finally, it proposes two optimization techniques, interleaved execution of fluid flow models and causality-preserving realtime synchronization relaxation, and shows that they significantly improve the performance of *MAYA*.

### 6.1 Improving execution speed with fluid flow model

Compliance with real time constraints is one of the key requirements for *MAYA* to emulate distributed applications. While it is intuitive that performance of a discrete event network simulator degrades as the size of the target network grows, its performance also degrades with increased capacity of heavily loaded links. On the other hand, a fluid flow based analytical model should be able to model large number of flows with constant execution time.

A simple network simulation scenario depicted in Figure 9(a) is created to compare the performance of these two modeling paradigms quantitatively. Nodes *S* and *D* are edge routers and nodes *X* and *Y* are core routers. Each solid line in the graph represents a full-duplex wired link with  $50ms$  propagation delay. There are 2000 identical TCP flows from *S* to *D* sharing the network path  $S - X - Y - D$ . Both QualNet and fluid flow model simulate the above scenario for a 500-second duration and the ratios of actual execution time over the simulation time (500 seconds in this case) are shown in Figure 10(a). As the link capacity increases from  $10Mbps$  up to  $70Mbps$ , the execution time of QualNet increases almost linearly while the execution time of the fluid flow model remains rather constant and is

much smaller than that of QualNet. It indicates that even if the network topology is simple, the growing number of background traffic flows will prevent discrete event simulators from keeping up with real time. Thus, it justifies the need to integrate fluid flow model into *MAYA* for emulating distributed applications in large-scale heterogeneous networks.

Next, the previous scenario is augmented with an additional subnet and expanded application traffic, to evaluate how fast *MAYA* can run. As shown in Figure 9(b), there is an 802.11b wireless subnet of 60 nodes connected to the wired network via a gateway in this scenario. The nodes are uniformly distributed in a  $300 \times 300$  square-meter terrain. There is moderate traffic among the nodes in the wireless subnet. There are 2000 background TCP flows between  $S$  and  $D$  in the wired network. At the same time, one wireless node  $S'$  has a synthetic application sending CBR traffic using UDP protocol to destination  $D$ . These UDP packets are of the same size (1000 bytes) and evenly distributed in the time scale. Given the *MAYA* framework, a natural way to configure the underlying network is to partition it according to the wireless subnet and the wired subnet. The wireless subnet is modeled by a QualNet instance while the wired subnet is modeled by a fluid flow model instance. As no physical network is involved in this experiment, *MAYA* can run as fast as possible without synchronizing with wall clock time, which permits a fair comparison with the pure discrete event simulation of QualNet. The experiment has been conducted on an SMP machine with eight 550MHz Intel CPUs. Default process scheduling policy of the system is used. The execution time of *MAYA* to model the network for 500 seconds duration is measured and the results are shown in Figure 10(b).

As packets flown between the wireless subnet and the wired subnet belong to aforementioned UDP traffic, the packet arrival interval at the source of the traffic can be varied to determine how *MAYA* will perform under various traffic volumes across the wireless subnet and the wired subnet. As shown in Figure 10(b), when such traffic volume is small, the execution time of *MAYA* is less than one fifth of the simulation time. The execution time reaches its lower bound when the packet arrival interval is about 10ms because time synchronization is dominant when the traffic volume is relatively low. The execution time will increase as the traffic volume further increases. However, *MAYA* performs quite well as it only takes about 250 seconds to run the 500-second long scenario even when packet arrival interval is as small as 1ms.

## 6.2 Time synchronization at finer granularity

The experiments in the previous subsection were used to evaluate whether *MAYA* can keep up with real time in the long run using synthetic application traffic as well as to measure how fast the framework can execute relative to the time duration it models. The results clearly demonstrate that *MAYA* is capable of modeling large scale network scenarios faster than real time. But it is important to see how *MAYA* performs at finer time granularity as well, as the objective is to use the framework to emulate distributed applications in real time. Experiments are conducted to model a so-called full network scenario shown in Figure 11(a). In this scenario, two physical subnets are added into the previous scenario depicted in Figure 9(b). Two real applications are running on two machines, one in each physical subnet. For easy measurement and analysis, the experiment employs a

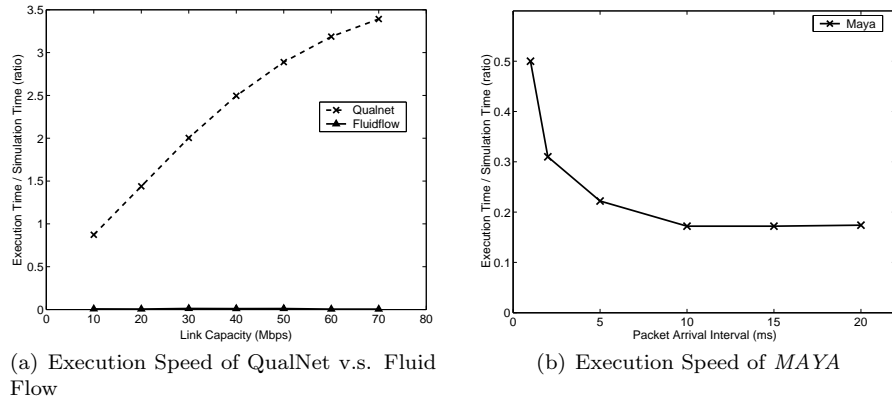


Fig. 10. Execution Speeds

simple UDP sender application capable of sending fixed-sized UDP packets at a fixed time interval as accurate as  $1ms$ . A UDP receiver application is used to consume these packets. The previously mentioned SMP machine is configured as the gateway between the two physical subnets, with *MAYA* executed entirely on this SMP machine. While the simulated network portions are modeled in the same way as that of the previous scenario, two physical network interface instances are added to inter-connect modeled subnets with the physical subnets. This configuration is used to emulate the above UDP applications.

The percentage of packets missing their deadlines has been used as the performance metric of this experiment. In this context, a packet moving from simulated subnet to physical subnet is considered missing its deadline if the wall clock has bypassed its timestamp. Due to tight time synchronization, *MAYA* might not always run ahead of real time. When that happens, packets going to physical subnets may miss their deadlines. As shown in Figure 12(a), the percentage of packets missing their deadlines is smaller than 10% for packet arrival intervals ranging from 1 millisecond to 20 milliseconds. This ratio remains rather constant within the range of the packet arrival interval shown in the figure, which motivates us to take a chronological snapshot of the packets going to the UDP receiver. Figure 12(b) visualizes a small portion of the trace immediately after finding a packet that misses

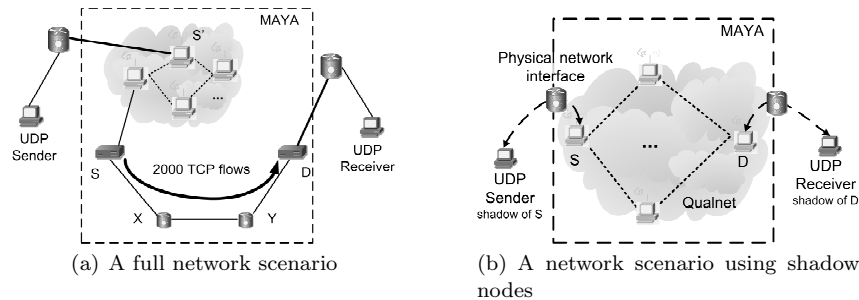


Fig. 11. Network Modeling Scenarios(Realtime)

its deadline. The snapshot is extracted from the trace of the experiment in which the packet arrival interval is 2 milliseconds. The graph reveals that late packets occur in bursts (each cross in Figure 12(b) actually denotes a sequence of consecutively late packets in the trace) and appear periodically. The number below a cross indicates the number of packets missing their deadlines in the single burst. In fact, similar behaviors can be observed on traces of all experiments that were conducted. They affect the accuracy of emulating distributed applications and are undesirable. The cause of this problem and its solution will be discussed in the next subsection.

### 6.3 Interleaved executions of the fluid flow model

As explained earlier in Section 2, the fluid flow model computes the instantaneous network statistics such as queuing delay and packet drop rate for the next time averaging interval at the beginning of such an interval. If incoming packets arrive at the fluid flow model before the analytical results for that interval are ready, the processing of these packets must be delayed. Time profiling on the fluid flow model indicates that this delay could be very costly because it takes up to 100ms for the ODE solver of the fluid flow model to compute the results for the next 1 second interval on the SMP machine. Furthermore, running as a single process, the fluid flow model will not be responsive to time synchronization events of the RTI-KIT until the ODE solver completes the computation. This in turn hinders the time advancements of other top tier components as well. Hence there are packets missing their real time deadlines periodically due to the periodic execution of ODE solver.

To tackle this problem, this paper presents an approach to interleaved executions of the fluid flow model. Assume the time averaging interval is  $T$ . The ODE solver takes time  $\phi$  to execute each time it is invoked, where  $\phi < T/2$ . Normally there is only one fluid flow model instance modeling a portion of the underlying network. If the ODE solver is invoked at time  $t$ , the output is meant for use in time interval  $[t, t + T]$ . However, the result is only ready after time  $t + \phi$ . In the new approach depicted in Figure 13(a), a second instance of the fluid flow model modeling the same portion of the network is interleaved with the original instance of the fluid flow model. The two intertwined instances are essentially the same except that

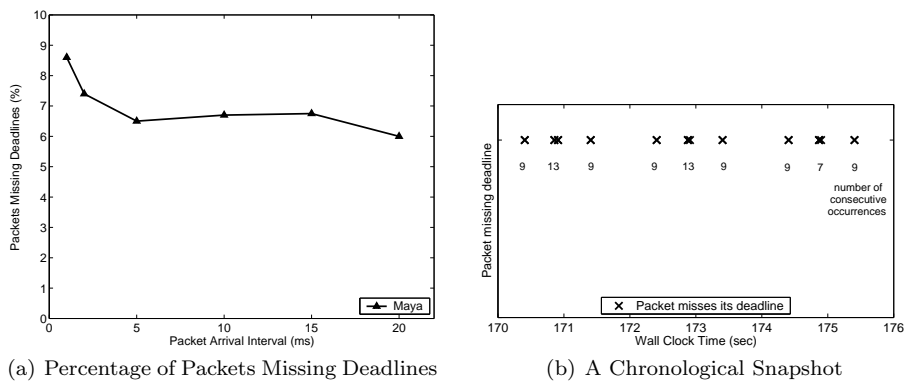


Fig. 12. Effects of Realtime Constraints on MAYA

the computations performed by the second instance are shifted by  $T/2$  from the first instance in the time scale. Thus, results produced by the first instance will be ready for use at the same time that the second instance is computing results for its next interval, and vice versa. In this interleaved execution approach, invocations to the ODE solver will be executed in a worker thread, while the main thread of the fluid flow model can process time synchronization events and schedule packet transmissions using results obtained earlier. This method should eliminate the penalties of invoking the ODE solver except for the first time when there are no previous results ready for use, which should be tolerable in most cases.

The experiments described in the previous subsection are repeated with the optimization presented above. Figure 13(b) shows the percentage of packets missing their deadlines in the experiments with the full network scenario, both with and without the interleaved executions of the fluid flow model in *MAYA*. It is encouraging to see that the average ratio of packets missing their deadlines is reduced very significantly to less than 0.2% with the interleaved executions of the fluid flow model.

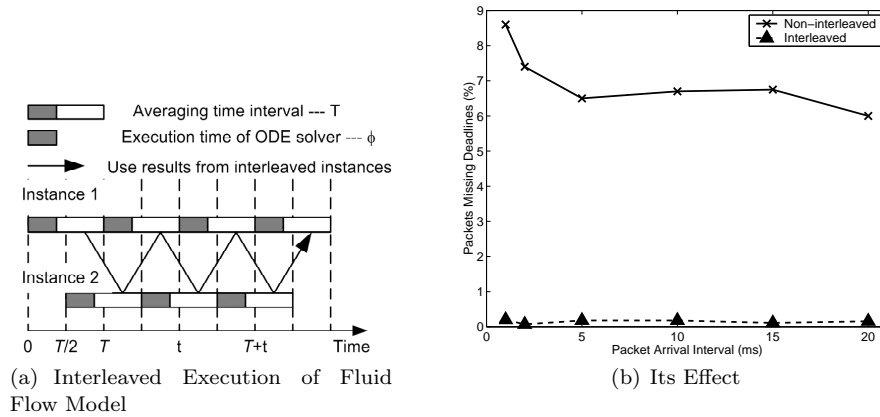


Fig. 13. Interleaved Executions of the Fluid Flow Model and the Effect

#### 6.4 Causality-preserving realtime synchronization relaxation

As explained in section 4, *MAYA* gives users the flexibility to not only run real applications on a node in a physical subnet, but also on a physical node (shadow node) which postures as a virtual node modeled by simulation, or more specifically, QualNet in current *MAYA* implementation. In this context, the physical network interface will act as the packet transporter between QualNet and the shadow node. One known problem aggravated by this situation is that there is little static lookahead value to explore at the physical network interface instance, which delivers packets from a shadow node to the corresponding virtual node. This is because there is ideally no propagation delay between this pair of nodes and additionally the simulation clock is bounded by the wall clock time as real applications may generate packets at any time. Thus, simulation will not be able to run ahead of

real time. As a consequence, when a peak burst of traffic generated by a real application enters simulation, simulation will not be able to simulate their traverse before their real time deadlines.

However, as the maximum throughput of each traffic flow is limited by the link capacity, bursty traffic often cannot be sent out of a node immediately and need to stay in the nodes buffer anyway. In other words, even if the physical network interface artificially delays the delivery of real application packets to QualNet by letting QualNet advance its simulation time faster without being hard-bounded by the real time constraint that may be present, the deferred packets can still be simulated by QualNet correctly later as long as they do not change the fates of already simulated packets or events. This kind of optimization will be referred to as *causality-preserving realtime synchronization relaxation*. Similar optimization techniques have been exploited in previous studies for dynamic lookahead extraction in PDES[Meyer and Bagrodia 1998; Fujimoto 1999; Liu and Nicol 2002]. The exact implementation of causality-preserving realtime synchronization relaxation depends on specific queueing management policies in modeled scenario. The most commonly used first in first out (FIFO) queue is used here to illustrate the idea. As shown in Figure 14(a), real application codes reside at the application and transportation layer of the protocol stack in a shadow node. Packets generated by real applications at the shadow node will be captured at the network layer and sent to the physical network interface immediately. The physical network interface timestamps the packets and sends them to the incoming packet queue at the corresponding virtual node in QualNet. The incoming packet queue may receive packets from local and other simulated nodes as well. QualNet can use its knowledge about the entire incoming packet queue to conservatively predict the earliest time that a packet would be simulated if it was received from physical network immediately. If the queue is empty, that time would be now. Otherwise, it should be some time  $t$  in future. The latter also means that it is safe for QualNet to simulate ahead of real time until time  $t$  without jeopardizing correctness of simulation results. QualNet will send this knowledge back to the physical network interface urgently using the second type of message at middle tier. Upon receiving such messages, the physical network interface will advance its virtual time ahead of its real time up to the bound predicted by QualNet. In this way, QualNet will be able to simulate further ahead of real time. However, it is important to know that the incoming packet queue at a virtual node may also receive packets (events) from local and other simulated nodes. Since simulation can run ahead of real time when the queue is not empty, real packets must be inserted into the queue using the timestamp when they are captured by the physical network interface rather than the expected timestamp with which they will be actually processed by QualNet. This is to ensure real packets captured with timestamps smaller than those of other simulation-generated packets will be processed earlier. Thus, this technique will preserve timestamp order of all events while improving the emulation performance.

A scenario shown in Figure 11(b) is used to study the effects of causality-preserving realtime synchronization relaxation. As shown in the figure, there are 50 wireless nodes (channel capacity 1.4Mbps) modeled by QualNet, with S and D posture as the sender and receiver of real applications. Two real machines are set

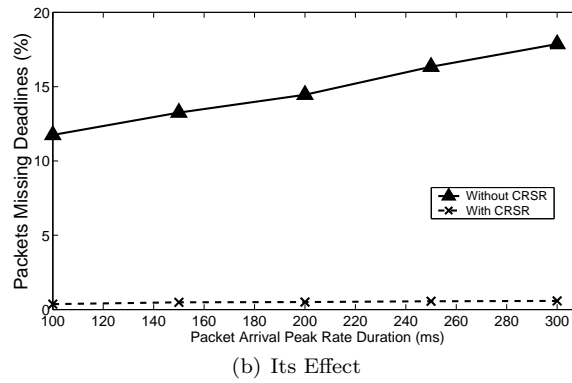
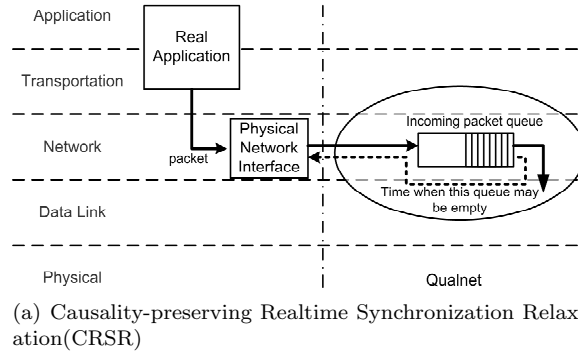


Fig. 14. Causality-preserving Realtime Synchronization Relaxation and its Effect

up as the shadows of S and D, where the real UDP sender and receiver applications run. Physical network interface acts as the bridge for connecting virtual nodes S, D to there shadows. The fluid flow model is not used in the experiment scenario as it is an abstract model not suitable for overlaying real applications. To model the behavior of bursty traffic, the UDP sender is modified to send at two rates alternatively, 1000 packets per second and 10 packets per second. Each packet has size of 1400 bytes. The UDP sender repeats sending at low and high data rate with a period of 1000ms. Synthesization of different length bursts of high data rate traffic is realized by changing the duty cycle of sending at high data rate. The experiment measures the percentage of packets missing their deadlines while varying the high data rate burst duration in each run within the range from 100ms up to 300ms. Results in 14(b) indicate that without causality-preserving realtime synchronization relaxation, the percentage of packets missing their deadlines keeps increasing as the traffic becomes more and more bursty. On the other hand, with causality-preserving realtime synchronization relaxation, *MAYA* is able to keep the percentage of packets missing their deadlines below 1%.

## 7. CONCLUSIONS

This paper has presented *MAYA*, a scalable and extensible network modeling framework for emulating distributed applications. It spans three distinct network mod-

eling paradigms: discrete event models, analytical models and physical network interfaces. In *MAYA*'s three-tier architecture, components at the upper tier utilize the functionalities offered by those at the lower tiers. Multiple instances of modeling paradigms reside at the top tier. Implementation issues of integrating heterogeneous modeling paradigms into this unified framework have been discussed with specific examples. Experiments has been conducted to validate the concept of mixing distinct modeling paradigms like QualNet and fluid flow TCP model and the results look promising. A series of experiments have been conducted to evaluate *MAYA*'s ability to conform to real time constraints. The results show that the fluid flow model executes much faster than QualNet and its execution time remains constant as network capacities increase. As a result, *MAYA* can keep up with real time in the long run. However, periodical invocations of the computation intensive ODE solver in the fluid flow model causes packets to miss their real time deadlines repeatedly. An approach to interleaved executions of the fluid flow model has been proposed to mask the turnaround time of the single instance fluid flow model. On the other hand, the flexibility of scaling network size and overlaying real applications on top of virtual networking nodes aggravate the problem for simulation to keep close time synchronization with wall clock. A causality-preserving realtime synchronization relaxation technique is also proposed to alleviate this undesirable side effect. Experiment results show that both approach significantly reduces the ratio of packets missing deadlines.

*MAYA* has been successfully used in the lab for emulating small scale distributed applications like NetMeeting and multimedia streaming applications. In the future, more experiments emulating more sophisticated distributed applications on larger-scale networks will be conducted. Study of further network modeling scalability improvement will be carried out by integrating parallel discrete event networks into the framework. The future work also includes evaluating the performance of distributed execution of *MAYA* over the NOW (network of workstations) infrastructure, especially when wall clock synchronization between multiple machines becomes necessary. A third dimension of the future works will be integrating application traffic models such as web traffic models[Crovella and Bestavros 1997; Erramilli et al. 1996] into the framework so as to generate more accurate background traffic when emulating distributed applications. Last but not the least, the framework will be exploited for quality of service (QoS) prediction and on-line control of large scale QoS aware networking application environment such as battlefield communication networks.

## REFERENCES

- ABOUZEID, A. A., ROY, S., AND AZIZOGLU, M. 2000. Stochastic modeling of TCP over lossy links. In *Proceedings of IEEE INFOCOM'00*.
- BAGRODIA, R., MEYER, R., TAKAI, M., CHEN, Y., ZENG, X., MARTIN, J., PARK, B., AND SONG, H. 1998. Parsec: A parallel simulation environment for complex systems. *Computer* 31, 10 (October), 77–85.
- CROVELLA, M. E. AND BESTAVROS, A. 1997. Self-similarity in world wide web traffic evidence and possible causes. *IEEE/ACM Transactions on Networking*.
- DANIEL, H. AND KURT, R. *A Dynamic Network Scenario Emulation Tool*. Institute of Parallel and Distributed High-Performance Systems (IPVR), University of Stuttgart.
- DASSF. *DaSSF 3.0 User's Manual*. Dartmouth College.

- DAVIES, N., BLAIR, G. S., CHEVERST, K., AND FRIDAY, A. 1995. A network emulator to support the development of adaptive applications. In *Proceedings of USENIX Symposium on Mobile and Location Independent Computing*.
- ERRAMILLI, A., NARAYAN, O., AND WILLINGER, W. 1996. Experimental queueing analysis with long-range dependent packet traffic. *IEEE/ACM Transactions on Networking* 4, 2, 209–223.
- FALL, K. 1999. Network emulation in the vint/ns simulator. In *the Fourth IEEE Symposium on Computers and Communications (ISCC'99)*.
- FUJIMOTO, R. AND ET AL. 2001. *FDK User Guide*.
- FUJIMOTO, R. M. 1998. Time management in the high level architecture. *Simulation* 71, 6 (December), 388–400.
- FUJIMOTO, R. M. 1999. Exploiting temporal uncertainty in parallel and distributed simulations. In *Proceedings of the 13th Workshop on Parallel and Distributed Simulation (PADS 1999)*. Atlanta, Georgia.
- GLOMOSIM. <http://pcl.cs.ucla.edu/projects/glomosim/>.
- HLA. *High Level Architecture*. <http://www.hla.mil>.
- KE, Q., MALTZ, D., AND B., D. 2000. Emulation of multi-hop wireless ad hoc networks. In *Proceedings of 7th International Workshop on Mobile Multimedia Communications (MoMuC'00)*.
- LIU, J. AND NICOL, D. M. 2002. Lookahead revisited in wireless network simulations. In *Proceedings of the 16th Workshop on Parallel and Distributed Simulation (PADS 2002)*. Washington D.C.
- LIU, Y., PRESTI, F. L., MISRA, V., TOWSLEY, D., AND GU, Y. 2003. Fluid models and solutions for large-scale ip networks. In *Proceedings of ACM SIGMETRICS'03*. San Diego, CA.
- MCCANNE, S. AND JACOBSON, V. 1993. The BSD packet filter: a new architecture for user-level packet capture. In *Winter USENIX Technical Conference*. San Diego, CA, 25–29.
- MEYER, R. A. AND BAGRODIA, R. 1998. Improving lookahead in parallel wireless network simulation. In *Proceedings of the Sixth International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems - MASCOTS '98*. Montreal, Quebec, Canada.
- MISRA, V., GONG, W., AND TOWSLEY, D. 2000. Fluid-based analysis of a network of AQM routers supporting tcp flows with an application to RED. In *Proceedings of ACM/SIGCOMM*.
- NICOL, D. M. 2001. Fluid simulation: Discrete event fluid modeling of tcp. In *Proceedings of WSC 2001*. Arlington, VA.
- NOBLE, B. D. AND ET AL. 1997. Trace-based mobile network emulation. In *Proceedings of SIGCOMM '97*. Cannes, France.
- NS2. *NS-2 user manual*. [www.isi.edu/nsnam](http://www.isi.edu/nsnam).
- OPNET. <http://www.opnet.com>.
- QUALNET. *QualNet user manual*. [www.scalable-networks.com](http://www.scalable-networks.com).
- RILEY, G., AMMAR, M., FUJIMOTO, R., PERUMALLA, K., AND XU, D. 2001. Distributed network simulations using the dynamic simulation backplane. In *Proceedings of the International Conference on Distributed Computing Systems*.
- RILEY, G. F., FUJIMOTO, R. M., AND AMMAR, M. H. 1999. A generic framework for parallelization of network simulations. In *MASCOTS*.
- SIMMONS, R. AND UNGER, B. 2001. Towards scalable network emulation. In *SPIE ITCOM 2001*. Denver.
- YUNG, T., TAKAI, M., AND BAGRODIA, R. 2001. Mixed fluid flow and packet level simulation models for large scale networks. In *Proceedings of SPIE*. Vol. 4523.
- ZHENG, P. AND NI, L. M. 2002. EMWIN: Emulating a mobile wireless network using a wired network. In *Proceedings of The Fifth International Workshop on Wireless Mobile Multimedia (WoWMoM 2002)*. Atlanta, Georgia, U.S.A.