

On the efficiency of fluid simulation of networks [☆]

Daniel R. Figueiredo ^{a,*}, Benyuan Liu ^b, Yang Guo ^c, Jim Kurose ^a, Don Towsley ^a

^a Department of Computer Science, University of Massachusetts, Amherst, MA 01003, United States

^b Department of Computer Science, University of Massachusetts, Lowell, MA 01854, United States

^c The MathWorks, Natick, MA 01760, United States

Available online 14 November 2005

Abstract

Performance evaluation of computer networks through traditional packet-level simulation is becoming increasingly difficult as networks grow in size along different dimensions. Due to its higher level of abstraction, fluid simulation is a promising approach for evaluating large-scale network models. In this paper we focus on evaluating and comparing the computational effort required for fluid- and packet-level simulation. To measure the computational effort required by a simulation approach, we introduce the concept of “simulation event rate”, a measure that is both analytically tractable and adequate. We identify the fundamental factors that contribute to the simulation event rate in fluid- and packet-level simulations and provide an analytical characterization of the simulation event rate for specific network models. Among such factors, we identify the “ripple effect” as a significant contributor to the computational effort required by fluid simulation. We also show that the parameter space of a given network model can be divided into different regions where one simulation technique is more efficient than the other. In particular, we consider a realistic large-scale network and demonstrate how the computational effort depends on simulation parameters. Finally, we show that flow aggregation can effectively reduce the impact of the ripple effect and that the ripple effect has less impact when simulating the WFQ scheduling policy.

© 2005 Elsevier B.V. All rights reserved.

Keywords: Fluid simulation; Packet simulation; Simulation efficiency; Network models

1. Introduction

Packet-level simulation is one of the most commonly used approaches to model and evaluate the

performance of computer networks. However, the vast increase in size and capacity of networks, such as the Internet, challenge the computational feasibility of packet-level simulation. Clearly, more efficient modeling techniques are needed in order to evaluate the performance of large-scale networks.

Many methods have been proposed to speed up the simulation of traditional network models. These methods can be loosely placed into three orthogonal categories (see Fig. 1): computational power; simulation technology; and simulation model. Simulations can be sped up by using more powerful

[☆] This research has been supported in part by the NSF under awards ANI-9809332, ANI-0085848 and EIA-0080119, and in part by CAPES (Brazil). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

* Corresponding author.

E-mail address: ratton@cs.umass.edu (D.R. Figueiredo).

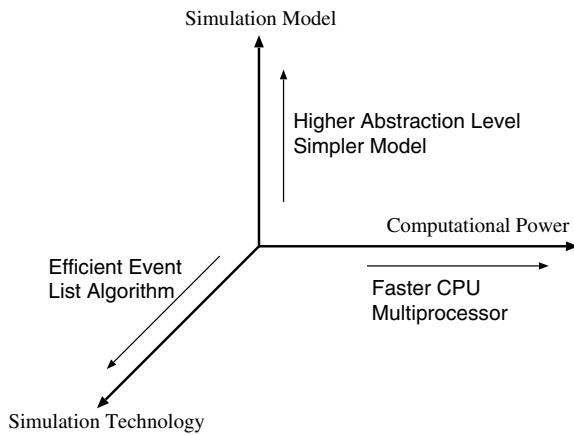


Fig. 1. Different dimensions in speeding up simulations.

processors or processor arrays. Improved simulation technology in the form of enhanced algorithms, such as the calendar queue and lazy queue algorithms [4,19], have been proposed to speed up event-list manipulation. Techniques based on importance sampling or the RESTART mechanism can facilitate rare event simulation [11,21]. The third dimension—simulation model—entails the development of network models that have higher levels of abstraction requiring fewer events to be simulated and, thus, improving efficiency. For example, the packet-train simulation technique models a cluster of closely-spaced packets as a single “packet-train” [1].

Another high-level model for computer networks are fluid models. Here, network traffic is represented as a continuous fluid flow, rather than a sequence of discrete packet instances. This modeling technique was initially applied by Anick et al. [2] to analytically evaluate the performance of data network traffic. Although originally proposed in the context of analysis, fluid models can also be simulated. Roughly speaking, while a packet-level simulator must keep track of all individual packets in the network, a fluid-level simulator must keep track of the fluid rates at all traffic sources and network queues.

The higher level of abstraction of fluid-level simulation suggests that it might require less computational effort than a packet-level simulation, since a large number of packets can be represented by a single fluid flow. We will see later in this paper, however, that this need not be the case. The complex dynamics of fluids that contend for resources in a queue can significantly increase the computational effort needed in a simulation, negating the benefits of a high abstraction level.

In this paper we investigate the dynamics of fluid simulation and characterize the computational effort required to simulate common network models. To characterize the computational effort we introduce a measure known as the *simulation event rate*. Using this measure we establish a direct comparison between packet-level and fluid-level simulation. For some of the scenarios considered, we derive analytical results for the simulation event rate of both fluid- and packet-level simulations. We also identify the major factors that contribute to the event rates of both approaches, such as the *ripple effect*, and establish the influence of model parameters such as source transmission rates and number of sources on the event rate. We also consider a realistic large-scale network model and analyze it in detail to characterize the dynamics of fluid simulation.

As with any model that abstracts away details, fluid models can potentially compromise accuracy of various performance metrics. This occurs because the timing information among closely-spaced packets is lost in the abstraction. In this work, we will not consider this potential loss of accuracy. We note, however, that several studies suggest that the loss in accuracy due to fluid models is usually small [16,15,14].

The remainder of this paper is organized as follows. Section 2 briefly presents related work in this area. Section 3 discusses the dynamics of fluid- and packet-level simulation. In Section 4, tandem network models are presented and analyzed. Section 5 considers cyclic network models. In Section 6, we evaluate the efficiency of a large-scale network model and discuss the tradeoffs involved in fluid simulation. In Section 7, we show that flow aggregation can reduce the fluid simulation event rate. Section 8 discusses the event rate of the WFQ (Weighted-Fair Queuing) policy. Finally, concluding remarks are presented in Section 9.

2. Related work

Fluid models have received much attention as they emerge as a promising paradigm to evaluate the performance of large-scale computer networks [2,15,12–14,3,9]. Although there is a vast literature on the analytical treatment of fluid models, many recent works have been concerned with the simulation of fluid models. Within this area, researchers have investigated both continuous- [14] and discrete-time simulation [12,13], and both pure

[12–14] and hybrid fluid models [3,9]. There has also been an emphasis in calculating performance metrics (e.g., throughput) [16,15,18] in fluid models and comparing these results with those obtained using equivalent packet-level models.

A related topic, which is the focus of this paper, is the *efficiency* of fluid-level simulation relative to packet-level simulation. Kesidis et al. [12] were the first to investigate the computational costs of simulating a given network model using fluid and packet-level simulation. They used actual simulators and compared the running times of both fluid- and packet-level models. Subsequent works have also focused on numerical comparisons between the running time required to solve both fluid- and packet-level models [14,9] and have shown that fluid models can yield enormous speedups. However, we note that such gains in efficiency cannot be generalized to arbitrarily network models. In particular, the network models investigated in [14,9] consider active queue management policies such as RED (Random Early Discard), and TCP sources, whose dynamics can be accurately described analytically using differential equations, facilitating the numerical solution.

Our work differs from these previous studies in that we focus on using the simulation event rate to characterize the computational effort analytically. We consider exact and discrete simulation of network fluid models with general, open-loop traffic sources. We view this as a fundamental step in exploring the full potential of fluid simulation.

3. Dynamics of fluid simulation

In this section we introduce a measure of the computational effort of simulations and describe the dynamics of sources and multiplexers of both fluid- and packet-level models. The dynamics of these building blocks are crucial for understanding the computational cost of simulating large network models.

3.1. Simulation event rate

In order to compare the efficiency of fluid- and packet-level simulation, we must establish a common metric for the computational effort required by both approaches. Clearly, the time required to execute each simulation is the ultimate comparison metric, since it directly reflects the computational effort required. However, this metric is not easily obtained analytically, and cannot be measured with-

out actually running the simulation. Moreover, simulation execution times can be both implementation and machine dependent. Thus, we introduce a more fundamental measure to represent the computational effort required by a simulation, namely its *simulation event rate*, defined as

$$E = \lim_{t \rightarrow \infty} \frac{N(t)}{t}, \quad (1)$$

where $N(t)$ is the total number of events that occur in the simulator by time t . In a packet-level simulation, events include source transitions, packet generation at sources and packet departures from queues. In fluid-level simulations, events include source transitions and fluid rate changes at sources and queues. This metric represents the average number of events that the simulator processes per unit of simulated time. The intuition is that simulation events are directly related to the computational effort involved in simulating a particular model. Different events will incur different computational costs, and these differences may vary from one simulator implementation to another. However, assuming that each event has some constant computational cost and considering all events during the execution of a simulation, we expect the simulation event rate to be a good indicator of the overall amount of computational effort required. Indeed, as we will see, the execution time of a given simulation is proportional to its corresponding event rate.

3.2. Source models

Due to their wide-spread use in the networking research literature, we will consider Markovian on–off sources as the traffic generation model. Such traffic sources are often used to capture the bursty nature of network traffic and can be easily combined to form more complex sources. Moreover, the on–off source model can be used in both packet and fluid models and has been widely studied in both contexts [20].

The on–off source consists of two states: *on* and *off*. State holding times are assumed to be exponentially distributed random variables with means $1/\lambda$ and $1/\mu$, respectively. In the packet model, a source generates packets according to a Poisson process with rate γ when in the *on* state. In the fluid model, the source generates fluid at a constant rate h . Neither packets nor fluids are generated when a source is in the off state. In order for a fluid source to be considered equivalent to a packet source, we

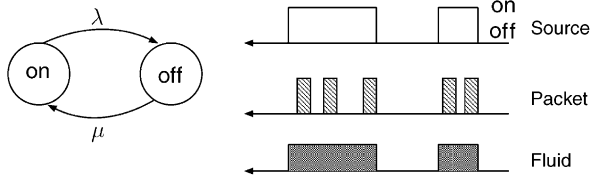


Fig. 2. On-off sources (packet and fluid).

minimally require that they have the same average data rate. Given this requirement, the constant fluid rate is $h = \gamma x$, where x is the average packet size of a packet source (without loss of generality, we assume that packets have a fixed size). Fig. 2 illustrates the behavior of the on-off packet and fluid sources.

Both fluid- and packet-level simulators must keep track of the state of on-off sources and schedule events for source state transitions. In addition, the packet-level simulator must generate one simulation event for each packet generated by the source, while the fluid simulator must generate a simulation event only when a source's fluid rate changes.

3.3. Multiplexers

The second, more interesting, basic network component is the multiplexer. An important characteristic of a multiplexer is its scheduling discipline. We consider two scheduling disciplines in this paper: FIFO and WFQ (Weighted Fair Queuing). The dynamics of FIFO scheduling are described below, while WFQ is presented in Section 8.

We start by revisiting the classic FIFO packet model, where packets arrive at a queue at discrete points in time and are placed in a buffer. Packets stored in the buffer are served from the queue in the order of their arrival and depart the queue at discrete points in time. Assume there are N packet sources feeding a FIFO queue with service rate c , and that packets arrive to the queue at time instants $\tau_1 < \tau_2 < \dots$. The dynamics of the queue size can be described by $q(t) = A(t) - D(t)$, where $A(t) = \arg$

$\max_{i \geq 1} \{t : \tau_i < t\}$ represents the number of arrivals in time $(0, t)$ and $D(t)$ represents the number of departures in time $(0, t)$. Fig. 3 depicts a FIFO queue being fed by two sources. Note that while packet departures occur in the same order as packet arrivals, their inter-packet spacings may change.

The dynamics of the FIFO fluid multiplexer are much more subtle. First, note that fluids from two different sources are distinct and do not mix in the queue, just as packets transmitted by two sources are differentiated within the queue. Fluids from different sources may arrive simultaneously at a queue (which cannot occur in the packet-level model) and the FIFO fluid queue can serve fluids from multiple sources simultaneously (again, unlike the packet-level queue). Due to this complexity, we provide a formal description of the dynamics of the fluid FIFO queue.

Assume there are N fluid sources feeding a FIFO queue with service capacity c . Let $a_k(t)$ and $d_k(t)$ be the fluid arrival and departure rates (amount of fluid arriving/departing the queue per time unit) of the k th source at time t . The aggregate fluid arrival rate at the queue is given by $a(t) = \sum_{k=1}^N a_k(t)$. Suppose that the arrival rate of fluids entering the queue change at times $\tau_1 < \tau_2 < \dots$ (i.e., at each τ_i , the fluid rate arriving at the queue of at least one source has changed). The dynamics of the queue size, $q(t)$, can be described by the following recursive equation:

$$q(t) = \max(0, q(\tau_i) + (a(t) - c)(t - \tau_i)),$$

$$\tau_i \leq t < \tau_{i+1}.$$

We can also describe the fluid departure rate of the k th source as a function of the fluid arrival rates. Note that the departure rate of a fluid flow depends not only on its own arrival rate, but on those of all other flows as well. A departing fluid flow shares the queue's service capacity with all other flows currently being FIFO-served in proportion to their arrival rates. There are two possible scenarios to consider when the aggregate flow arrival rate changes at time τ_i

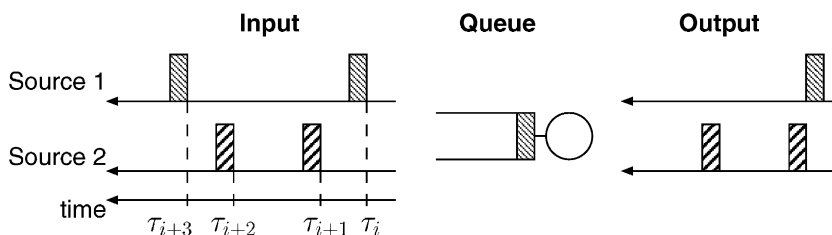


Fig. 3. Dynamics of a FIFO packet multiplexer.

- $q(\tau_i) = 0$. In this case, the arrival rate change results in an immediate change in the departure rate, since no fluid is backlogged in the queue. There are two subcases to consider:
 - $a(\tau_i) \leq c$. In this case, the departure rate of each source is equal to its arrival rate, $d_k(\tau_i) = a_k(\tau_i)$.
 - $a(\tau_i) > c$. In this case, a backlog begins to form at time τ_i . For a FIFO queue, each source receives service at a rate proportional to its arrival rate, thus, $d_k(\tau_i) = ca_k(\tau_i)/a(\tau_i)$.
- $q(\tau_i) > 0$. In this case, the fluids backlogged at τ_i must first be served before the arrival rate change at time τ_i is reflected in a changed departure rate. There are again two subcases to consider:
 - $a(\tau_i) > c$. In this case, once the backlog $q(\tau_i)$ is served, the departure rate of flow k is given by $ca_k(\tau_i)/a(\tau_i)$. Thus, $d_k(\tau_i + q(\tau_i)/c) = ca_k(\tau_i)/a(\tau_i)$.
 - $a(\tau_i) \leq c$. In this case, although the aggregate arrival rate at τ_i is less than or equal to the service capacity of the queue, the fluid that arrives while the current backlog $q(\tau_i)$ is being served will accumulate in the queue. Hence, after the current backlog is served, the new corresponding departure rate of source k will be $d_k(\tau_i + q(\tau_i)/c) = ca_k(\tau_i)/a(\tau_i)$. Note that $d_k(\tau_i + q(\tau_i)/c) > a_k(\tau_i)$ for all k , i.e., the departure rate of fluid that arrived at time τ_i will be larger than its arrival rate. This newly created backlog will eventually be completely served and the departure rate for each source will drop to its instantaneous arrival rate at that moment. Since it takes $(a(\tau_i)q(\tau_i)/c)/(c - a(\tau_i))$ time units to completely serve this newly accumulated backlog, we have $d(\tau_i + q(\tau_i)/c + (a(\tau_i)q(\tau_i)/c)/(c - a(\tau_i))) = a_k(\tau_i)$, for all t such that $\tau_i + q(\tau_i)/c + (a(\tau_i)q(\tau_i)/c)/(c - a(\tau_i)) \leq$

$t < \tau_{i+1}$. Note that depending on the time of the next arrival rate change, τ_{i+1} , this rate change may not occur.

Fig. 4 illustrates the dynamics of the FIFO fluid queue being fed by two independent and identical on–off sources. In this example, we assume that the fluid rate of each source, h , is less than the service rate of the multiplexer, c , but that $2h > c$. At time τ_1 the first source enters the *on* state and immediately starts to receive service at its arrival rate, h . At time τ_2 , source 2 switches to the *on* state and both sources now share the service rate equally, each being assigned a departure rate of $c/2$. Note that a backlog forms in the queue, as the aggregate arrival rate exceeds the service capacity. At time τ_3 , source 2 moves to the *off* state, but due to the accumulated backlog, this change will not be reflected in the departure rates until time $\tau_3 + q(\tau_3)/c$. At that time, source 1 is assigned a departure rate of c , the queue’s full service rate, as its fluid has accumulated in the queue after source 2 switched off. However, at time $\tau_3 + q(\tau_3)/c + (a(\tau_3)q(\tau_3)/c)/(c - a(\tau_3))$ the newly accumulated backlog will have been served and the departure rate drops to the arrival rate, h . Finally, at time τ_4 source 1 switches off. This last change is reflected immediately in the departure rate, since there is no backlog present.

Note that the arrival rate changes of source 2 at times τ_2 and τ_3 cause changes in the *departure* rate of source 1. In general, a change in the arrival rate of single flow can cause changes in the departure rate of many other flows. In such cases, we say that a change in a flows’s arrival rate *interferes* with other flows traversing the queue. Due to such interference, the departure rate process of a given flow may contain many *more* rate changes than its arrival process, as illustrated in Fig. 4. However, not every arrival

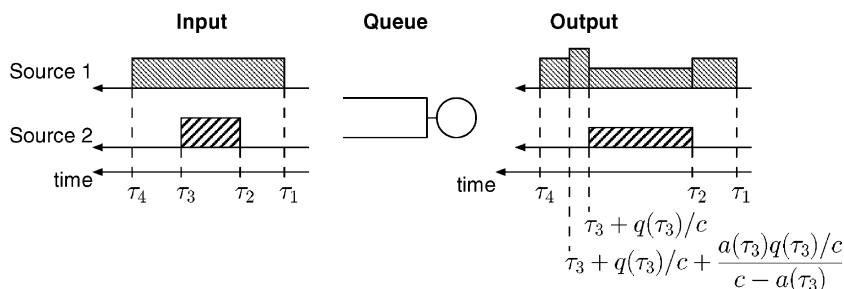


Fig. 4. Dynamics of a FIFO fluid queue.

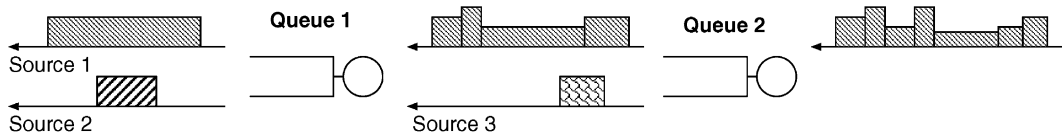


Fig. 5. Dynamics of fluid flow traversing two FIFO queues.

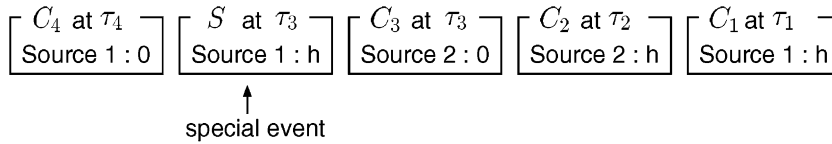


Fig. 6. Different fluid chunks of a FIFO queue.

rate change affects the departure rates of other flows being served, for example, when $a(\tau_i) \leq c$.

In a larger network model, a rate change in a given flow may interfere with many other flows that share common downstream queues. In particular, a rate change can propagate through the network causing changes in the departure rates of other flows. This phenomena is known as the *ripple effect* and has been observed before [12,13]. The ripple effect has a profound impact on the number of rate changes that must be simulated and, thus, in the efficiency of fluid simulation. Fig. 5 illustrates this phenomena by showing the actual fluid rates produced by flow interference. In this figure, source 1 flows through queues 1 and 2, while source 2 flows only through queue 1 and source 3 flows only through queue 2. We observe that the number of rate changes in the departure rate of flow 1 at queue 2 is much larger (8 rate changes) than the number of rate changes at the source (2 rate changes).

3.4. Events in fluid simulation

The discrete-time simulation of a packet-level FIFO queue is straightforward since packets are handled individually. The simulation of a fluid FIFO queue, however, is much more complicated since the arrival rates of all flows must be maintained during periods that the queue is backlogged, as illustrated in the above example. To perform this bookkeeping, we introduce the notion of *fluid chunks*. A fluid chunk C_i is defined as the set of fluid arrival rates that changed at time τ_i . Flows traversing the queue that did not change their arrival rates at time τ_i are not part of fluid chunk C_i . Since several arrival rates can change simultaneously, the fluid chunk need not be a singleton set. Note that

fluid chunk C_i will be reflected at the departure side of the queue when the backlog $q(\tau_i)$ has been served. If the queue is empty ($q(\tau_i) = 0$) at the time the fluid chunk is formed, then it is processed immediately.

Unfortunately, this definition of fluid chunk is not sufficient to handle all events associated with a fluid FIFO queue. In particular, recall from our discussion above that when $q(\tau_i) > 0$ and $a(\tau_i) \leq c$, a change in the departure rate of fluids can occur without any changes in their arrival rates. In this case, we need a special event that will change the departure rate of the fluids after the backlog is served. Fig. 6 illustrates the fluid chunks that are formed when simulating the example shown in Fig. 4. Note that a special event must be scheduled at time τ_3 as the departure rate of source 1 will first change to c (the full capacity of the queue), which occurs when C_3 is processed, and only change to h after the backlog is served.

An important property of fluid chunks is the fact that two consecutive chunks can merge in the queue. Chunk merging occurs in the following scenario: Assume a queue has a backlog $q(\tau_i)$ when the fluid rate of flow k changes from h to zero at time τ_i . At this instant a fluid chunk C_i is formed. Now suppose that at time τ_{i+1} the same flow k changes its fluid rate back to h . A new fluid chunk, C_{i+1} will then be formed at the queue. However, if $\tau_{i+1} < q(\tau_i)/c$ then C_i can be discarded and need not be processed by the simulator. In this case, we say that the two consecutive on periods of flow k have merged in the queue.

Special events and chunk merging can impact the computational effort required by fluid simulation and the number of events simulated. However, in most cases the impact of special events and chunk merging is very small. In particular, the probability

that such events occur during simulation decreases as the size of the simulated model increases (e.g., in number of sources, higher loads, etc.). In the analytical evaluation that follows, these events will be conveniently approximated or ignored for tractability, although they are fully accounted for in our empirical results.

3.5. Fluid- and packet-level simulation framework

In order to compare the actual execution times of a fluid- and packet-level simulator, and also to collect the simulation event rate and validate our analytical approximations for the event rate, we designed and implemented two simulation frameworks. Each framework consists of simple and independent network components such as sources, links, routers and queues that form the building blocks of a network model. These building blocks can be instantiated by specifying their specific parameters, and connected together to form a large-scale network model. The two simulation frameworks have very similar interfaces, such that a network model can be executed in either simulator with minimal modification. Both simulation frameworks were implemented using SSF (Scalable Simulation Framework), which allows the construction of general purpose event-driven simulations [5]. In particular, we used the DaSSF implementation of SSF which provides enhanced simulation features [6]. Both frameworks, as well as several network models developed as part of this study, are publicly available.¹

4. Tandem networks

In this section we compare the simulation event rate of fluid- and packet-level simulation for tandem network models. We start with a very simple model composed of a single on-off source feeding a single FIFO queue with an infinite buffer and service capacity c . This simple model allows us to derive exact analytical expressions for the simulation event rates of both simulation techniques, and thus illustrate several fundamental differences in their computational requirements.

Let e^t be the event rate associated with source state transitions. Let e^a be the event rate associated

with the arrival of fluid rate changes at the queue, which for this simple model is simply given by the source rate change. Let e^d be the event rate associated with fluid rate changes at the departure side of the queue. Then, assuming $h > c$, the exact simulation event rate for this fluid model is given by

$$E_F = e^t + e^a + e^d \\ = \frac{2\lambda\mu}{\lambda + \mu} + \frac{2\lambda\mu}{\lambda + \mu} + 2\lambda \left(1 - \frac{\lambda h}{c(\lambda + \mu)}\right). \quad (2)$$

The three terms in Eq. (2) correspond to e^t , e^a and e^d , respectively. The event rates e^t and e^a derive directly from the on-off source model, while e^d is more subtle. Note that if $h \leq c$, then the queue's departure event rate equals the queue's arrival event rate, as no queuing will take place. However, if $h > c$, then chunk merging can occur and changes in the departure fluid rate will occur only when the queue is empty and the source is *off*. The parenthesized term within the expression for e^d gives the exact probability that the queue is empty and the source is in the *off* state. This joint probability can be computed using the technique presented in [2]. Given that the source is off, the rate at which the source changes its transmission rate is simply λ . Since a busy period is followed by an idle period, we actually have two rate changes.

The simulation event rate for the packet-level simulation is simply given by

$$E_P = \frac{2\lambda\mu}{\lambda + \mu} + \frac{2\gamma\mu}{\lambda + \mu}. \quad (3)$$

The first term in (3) is identical to that of fluid simulation and represents the event rate associated with the source transitions. The second term is the event rate for the packet arrival and departure events. This is simply twice the source packet generation rate, since each packet creates both an arrival and departure event.

From Eqs. (2) and (3) we observe that neither fluid- nor packet-level simulation is consistently more efficient. In fact, the event rate of each simulation paradigm depends on the parameters of the network model. The choice of parameters effectively dictates the more efficient simulation paradigm. For example, when the packet generation rate, γ , is sufficiently large (more specifically, when $\gamma > \lambda/\mu(\lambda + 2\mu - \lambda h/c)$), the fluid simulation event rate will be less than its packet-level counterpart. In this case, the fluid simulation will require less computation effort than the equivalent packet-level simulation.

¹ The simulator can be obtained at <http://gaia.cs.umass.edu/fluidsim>.

Although not presented here, this analysis was confirmed using actual simulations which show that the predictions using the simulation event rate are in accordance with the simulation running time. Note that the event rates associated with source transitions, e^t , are identical in both simulation approaches and thus will be ignored in future comparisons.

4.1. Multiple sources, single queue

We now extend the above model and consider the case in which a FIFO queue is fed by N statistically independent and identical on–off sources. We assume there exists a k , $1 \leq k \leq N$, such that $k\gamma \leq c < (k+1)\gamma$, i.e., that $k+1$ sources (or more) are required to be *on* in order for the queue to accumulate a backlog. Consider a single flow, i , traversing this queue in the fluid model and the events that must be simulated. The flow arrival event rate is identical to the single source case. The departure event rate is now more subtle due to flow interference. If the queue is non-empty and flow i is in the *on* state, a rate change in any other flow will affect the eventual departure rate of flow i . This occurs since the number of flows sharing of the link changes, thus creating a new flow chunk. Similarly, if the queue is empty and flow i is *on*, the departure rate of flow i will change when more than k sources are *on*. The following Proposition for the departure event rate of a fluid source summarizes this discussion:

Proposition 1. *Consider N identical on–off sources flows feeding a FIFO fluid queue. Then, the departure event rate of a flow traversing this queue is given by*

$$e^d = e^a + \alpha(N-1)e^a + \beta, \quad (4)$$

where α is the probability that source i is in the *on* state and the queue is non-empty; and β is the rate at which the queue transits from a non-empty to an empty period given that source i is *on*.

The quantities α and β can be computed numerically using techniques presented in [2]. Note that β is the rate at which the queue changes from the empty state with $k-1$ sources being active (*on*) and source i being in the *on* state, to the state where k sources and source i are *on*.

The above proposition ignores chunk merging and special events. By ignoring chunk merging, we overestimate the event rate, while by ignoring special events, we underestimate the event rate. Although these quantities can be non-negligible

under certain circumstances (e.g., a small number of sources, low loads), we expect them to be negligible for most interesting scenarios (e.g., a large number of sources, high loads). The quantitative results below obtained through simulation support this conjecture.

From the proposition above, we have that the total fluid simulation event rate of the model is given by

$$E_F(N) = N(e^a + e^d) = \alpha N(N-1)e^a + 2Ne^a + N\beta. \quad (5)$$

The event rate in Eq. (5) has a quadratic dependency on N . This occurs because sources can interfere with the departure rate of one another. However, this dependency will be mitigated if the queue is empty most of the time (i.e., α is very small). To investigate this behavior we next consider some numerical results.

The packet simulation event rate is just N times the packet simulation event rate of the single source case, thus

$$E_P = N \frac{2\gamma\mu}{\lambda + \mu}. \quad (6)$$

Let ρ be the load of the queue, which is defined as the ratio of the average aggregate arrival rate to the service rate of the queue. Thus, $\rho = 1/c \sum_{i=1}^N h_i \mu_i / (\lambda_i + \mu_i)$, where N is the number of sources traversing the queue and h_i , μ_i and λ_i are the parameters of source i . In the examples that follow, we let $\lambda_i = \mu_i = h_i = 1$, for all i .

In the first experiment we fix the service rate of the queue ($c = 50$) and vary the number of sources, N , from 2 to 95 (which corresponds to loads of 0.02–0.95). Fig. 7(a) shows the fluid and packet simulation event rate as a function of N ; both analytical results and a direct measurement of the fluid simulation event rate are shown ($\gamma = 5$). The packet simulation event rate increases linearly, while the fluid simulation event rate exhibits two different behaviors. For a small number of sources (which corresponds to low loads), the fluid simulation event rate increases linearly, since interference among fluid flows is minimal. For a large number of sources, the event rate increases dramatically as flow interference becomes significant. Recall that fluid flows interfere only when the queue is non-empty and that the probability of a non-empty queue increases sharply as the load approaches 1. Thus, fluid simulation has a higher computational cost at high loads. We will see shortly that the cross-over

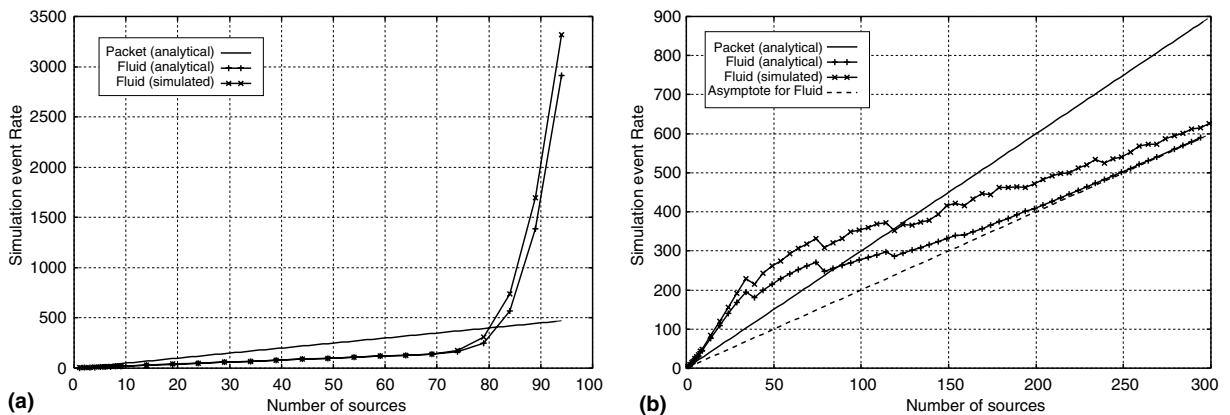


Fig. 7. Simulation event rate of multiple sources traversing a single FIFO queue: (a) fixed service capacity; (b) fixed load.

point will depend on the packet generation rate γ . Finally, we observe that the analytical expression for the fluid event rate from Proposition 1 (which ignores special events and chunk merging) closely matches the measured simulation event rate.

We next fix the load ($\rho = 0.8$) and vary the number of sources from 2 to 295. In order to maintain a constant load we scale the service rate of the queue from 1.25 to 184.375. The resulting fluid simulation event rate is shown in Fig. 7(b) together with the packet simulation event rate ($\gamma = 3$). Note that the fluid simulation event rate increases with the number of sources, and then converges to a linear growth rate. The dashed line represents twice the arrival event rate, which is the total event rate if no flow interference were to occur. Note that the analytical and measured fluid simulation event rates exhibit similar behavior and converge to a value twice that of the arrival event rate. This convergence occurs because the probability that the queue is non-empty goes to zero as the number of sources increases and the load remains constant. Therefore, the probability of flow interference also goes to zero and the departure event rate becomes equal to the arrival event rate.

In the analysis above, we have assumed that the queue has an infinite buffer capacity. A natural extension would be to consider the impact of finite buffers in the simulation event rate. In packet-level simulation, a finite buffer can only reduce the simulation event rate, as a departure event is suppressed when a packet arrives to a full queue and is discarded. However, in the single queue, single source case, a finite buffer can only increase the fluid simulation event rate. This occurs because the finite buffer reduces the chance that fluid will merge in the

queue (i.e., higher probability that the queue is empty). On the other hand, in the case of multiple sources, the fluid simulation event rate can both increase and decrease in the presence of finite buffers. The latter occurs because the probability of flow interference is reduced (i.e., higher probability that the queue is empty). In any case, for most network models of interest, where the loss probability is usually low, the impact of finite buffers on the simulation event rate is expected to be small. Although not presented here, we have conducted simulations using finite buffers that support this claim.

4.2. Multiple sources, multiple queues

We further generalize the above single queue model and consider a tandem queuing network formed by a sequence of queues and multiple sources. This scenario is of interest since complex tandem networks can generally be decomposed into various intersecting tandem queues whose structure provides for a tractable analytical evaluation of the fluid simulation event rate.

Our tandem queuing network model of K FIFO queues is depicted in Fig. 8. A single on-off source enters the first queue and traverses all other queues. Each queue is additionally fed by $N - 1$ identical and independent additional on-off sources that leave the system after traversing that queue.

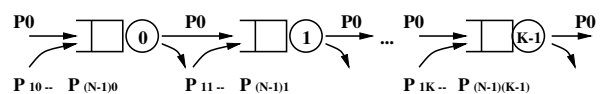


Fig. 8. A tandem queuing network.

Although, this system has many parameters, two parameters—the number of sources entering each queue (N) and the number of queues in the system (K)—are the most important, since they have direct influence on flow interference and the ripple effect.

We can approximate the fluid simulation event rate by considering one queue at a time. Let q_i be the event rate at queue i , $0 \leq i < K$. Then

$$q_0 = N(e^a + e_r^d(0)), \tag{7}$$

$$q_i = (N - 1)(e^a + e_r^d(i)) + e_s^d(i - 1) + e_s^d(i), \tag{8}$$

where e^a is the arrival event rate from a single on–off source to a queue, $e_s^d(i)$ is the departure event rate at queue i of the source that traverses all queues, and $e_r^d(i)$ is the departure event rate of a single cross-traffic source (that exits after one queue) at queue i . We can easily compute e^a as in the single queue case, while the departure event rates are given by

$$e_r^d(i) = e^a + \alpha(e_s^d(i - 1) + e^a(N - 2)) + \beta, \tag{9}$$

$$e_s^d(i) = e_s^d(i - 1) + \alpha e^a(N - 1) + \beta, \tag{10}$$

where we let $e_s^d(-1) = e^a$. Note that α and β actually depend on i . In the above equations, we consider an approximation where α is the probability that a given source is in the *on* state and the queue is non-empty, and β is the rate at which the queue transits from a non-empty to an empty state given that a particular source is *on*. This approximates the effects of flow interference. We can solve the recursion in Eq. (10) and obtain

$$e_s^d(i) = e^a + \alpha e^a(i + 1)(N - 1) + \beta(i + 1). \tag{11}$$

Thus, the fluid simulation event rate can be approximated by

$$E_F(K) = \sum_{i=0}^{K-1} q_i, \tag{12}$$

$$= 2NKe^a + (N - 1)^2Ke^a\alpha + (N - 1)K^2e^a\alpha + (N - 1)^2K(K - 1)e^a\alpha^2/2, \tag{13}$$

where we have ignored the contribution of β . As in the single queue case, this approximation also ignores flow merging and events associated with special chunks. Note that the fluid simulation event rate, $E_F(K)$, depends quadratically on the number of queues in the tandem system and on the number of flows traversing the queues. This quadratic behavior is a manifestation of the ripple effect, which increases as the number of queues or sources increase. However, we will see shortly that another fundamental quantity—the probability the queue is non-empty—can mitigate this quadratic growth.

The packet-level simulation event rate is much simpler to calculate. Its exact form is given by

$$E_P(K) = KN \frac{2\mu\gamma}{\lambda + \mu}. \tag{14}$$

Note that this is simply KN times the packet rate of each source, with each packet generating two events (arrival and departure) at each queue. As expected, the event rate is linear in each of K and N .

In order to illustrate the tradeoffs, we simulate an instance of the tandem model above. We first consider the impact of K , the number of queues in the tandem system, on the simulation event rate, for $N = 15$ sources with $\lambda = \mu = 1$ for all sources and $\rho = 0.8$. Fig. 9(a) shows the fluid and packet simulation event rate (for $\gamma = 25$). We observe that the fluid simulation event rate increases quadratically

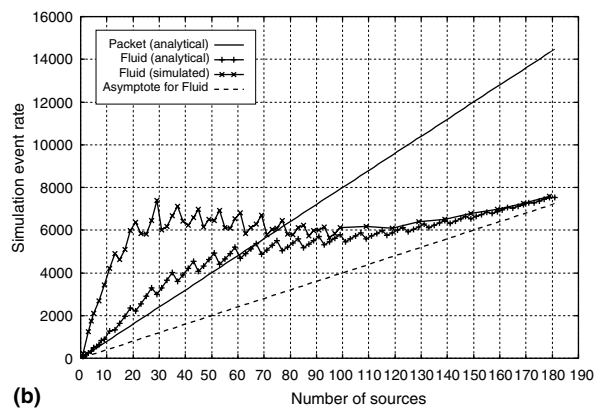
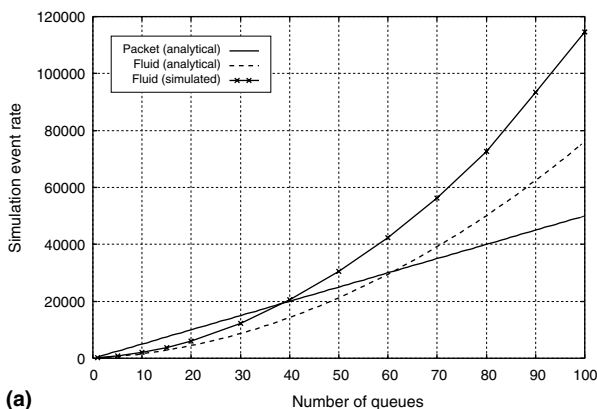


Fig. 9. Simulation event rate of a tandem system: (a) as a function of the number of queues (fixed $N = 15$, load $\rho = 0.8$, $\gamma = 25$); (b) as a function of the number of sources (fixed $K = 20$, load $\rho = 0.8$, $\gamma = 4$).

with the number of queues in the system, as predicted by our analysis, while the packet simulation event rate exhibits a linear increase. Note that if the network is small enough, fluid simulation can be more efficient, despite its quadratic dependence on K . Also, as for the single queue case, fluid simulation can be more efficient if γ is sufficiently large. Other models, with differing number of sources entering each node, also exhibit similar qualitative behavior.

The second important parameter to consider is N , the number of flows entering each queue, as this affects the interaction among different flows. We consider the same scenario as above, but with $K=20$ queues. We maintain a constant load ($\rho = 0.8$) for all simulations by increasing the service rate of the queues as N increases. Fig. 9(b) presents the results obtained for this scenario for $\gamma = 4$. Surprisingly, the fluid simulation event rate increases sub-linearly with the number of sources and then approaches a linear increase equal to twice the arrival event rate of all sources in the system. Note that the analytical and measured fluid simulation event rate have very similar behavior, and both approach the same asymptotic value. The explanation for the sublinear growth and the asymptotic value is similar to that of the single queue case—the probability that flows interfere with one another goes to zero at constant load as the number of sources increases. Since the packet-level simulation exhibits a linear increase, the fluid simulation will have a smaller event rate than its packet counterpart for large enough values of γ .

4.3. Packet-level and fluid simulation efficiency regions

In the previous section we have seen that either fluid- or packet-level simulation can be more efficient depending on model parameter values (N and K , among others). In order to determine the conditions at which the simulation event rate of both approaches are equal, we can equate our two equations for the fluid and packet simulation event rates

$$E_F(K) = E_P(K). \quad (15)$$

This equation is quadratic in N and K , and can be solved numerically for any of the model parameters, giving the boundary between the efficiencies of the two simulation approaches. For any given point on one side of this boundary, one simulation para-

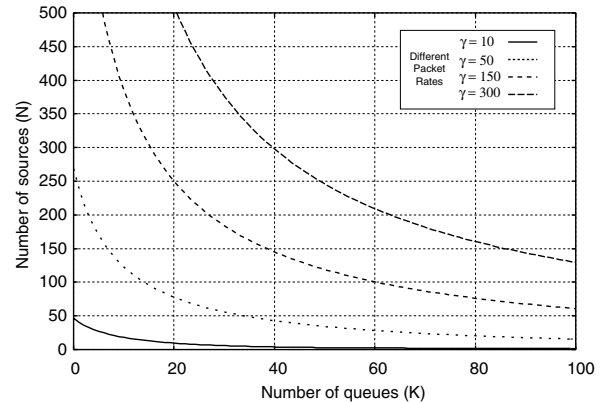


Fig. 10. Boundary condition for efficiency regions between fluid- and packet-level simulation.

digm will always outperform the other. Fig. 10 shows the (N, K) boundary region curves with different values of the packet rate (γ) using the parameters as previously defined. We observe that as the number of queues or number of sources increase, fluid simulation becomes less efficient and eventually packet-level simulation becomes more efficient (area to the right of the curves). Also, increasing the packet generation rate, γ , shifts the boundary curve in the upper-right direction, increasing the size of the parameter space in which fluid simulation is more efficient. It is also possible to establish boundaries based on other parameters, such as (N, γ) , as discussed in [10].

5. Cyclic networks

In this section we consider a network topology with a feedback cycle. This poses a significant challenge to the computational requirements of fluid simulation, since a source rate change can modify the rate of several other flows, with these rate changes propagating forward to other queues in a cycle, creating additional rate changes and resulting in a cascading effect. Indeed, in such scenario, one might expect the simulation event rate to grow without bound, with the simulator processing simultaneous rate changes in queues forming a cycle without ever advancing simulation time.

In order to investigate the simulation event rate in a network with a feedback cycle we consider the cyclic system depicted in Fig. 11. This scenario contains four identical, infinite buffer FIFO queues connected together in a cycle. An on-off source generates traffic into each of the queues. The routing of

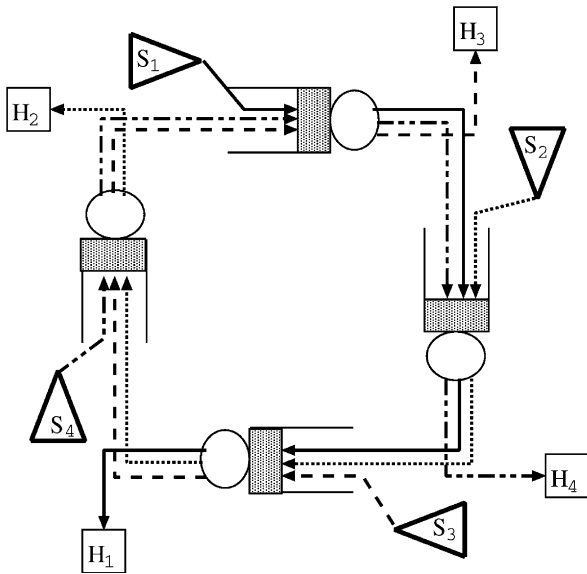


Fig. 11. A cyclic network model with a feedback cycle.

the flows is such that each flow traverses exactly three queues before departing the system: three different flows thus traverse each queue. The four traffic sources are identical, having equal on and off periods and a peak rate equal to 1. The stability condition of this system requires each service rate to be greater than 1.5, the average aggregate arrival rate to each queue.

5.1. Packet-level simulation of the cyclic network

A cyclic model does not introduce any special considerations into a packet-level simulation. A packet generated at a source simply traverses the queues, and leaves the system without introducing additional events in other packet flows. When the system is stable (load < 1), the simulation event rate is independent of the service rate and link propagation delay. The packet-level simulation event rate for the model in Fig. 11 is thus fully determined by the source packet rate and the number of nodes a flow traverses, and is given by

$$E_p = (e_s + e_q N_q) N_n \tag{16}$$

where e_s denotes the event rate of a single source; e_q is the per-source event rate of the queue; N_q is the number of queues traversed by each flow; and N_n is the total number of nodes in the system. The event rate associated with each flow is given by $e_s + e_q N_q$. Since each node introduces a single source into the system, the overall event rate is given by

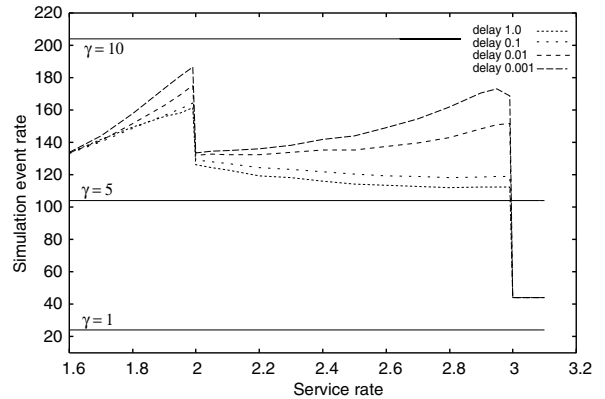


Fig. 12. Simulation event rate for the cyclic network model.

Eq. (16). This equation was verified against simulation results (not presented here) and proved to be a very good estimate of the measured event rate.

In order to evaluate the system under different loads, we vary the service rate of the queues from 1.6 to 3.1, which corresponds to system loads of 0.94–0.48, respectively. The link propagation delay will also be varied, taking values of 0.001, 0.01, 0.1 and 1 time units. Note that neither of these parameters influence the packet-level simulation event rate, but, as we will see, are fundamental in determining the event rate of fluid-level simulation. Eq. (16) was used to plot the packet-level simulation event rates for $\gamma = 1, 5, 10$ in Fig. 12 (solid lines).

5.2. Fluid-level simulation of the cyclic network

It is not easy to obtain an accurate closed form approximation for the fluid simulation event rate in cyclic network models, as the cyclic nature of the topology creates complex dynamics in the flow’s rate change process. Thus, we consider the event rate of the fluid cyclic network through simulation only.

Using our simulator, we measured the fluid simulation event rate and observed that it converged to a finite value under all scenarios considered. The results are shown in Fig. 12 as a function of the service rate of the queues. Each staircase shaped curve corresponds to a different link propagation delay. Given our discussion above about the cascading (ripple) effect in fluid cyclic networks, it may seem surprising that the simulation event rate converged at all. Recall, however, that an arrival flow rate change will not cause other flows to change their departure rates if the queue is empty and the aggregate arrival rate

into the queue is smaller than its service rate. Thus, even in a cyclic queuing system, a flow rate change will only cause a finite number of new rate changes at “downstream” queues, if the joint probability that a queue is empty and the instantaneous aggregate arrival rate is smaller than the service rate is greater than zero. Under this condition, a flow rate change will eventually arrive at a queue at which no other flow will suffer a departure rate change, terminating the ripple effect.

The above condition for obtaining a finite event rate is directly related to the queue’s service rate. Interestingly, the service rate has two opposite effects. First, as the service rate increases, the average queuing delay decreases, making fluid rate changes propagate faster through the cyclic network. This effect tends to increase the event rate, since more fluid rate changes occur within a fixed time interval. On the other hand, a larger service rate implies that the queue is more likely to be empty and have an instantaneous aggregate arrival rate that is smaller than its service rate, resulting in a larger probability that a fluid rate change will not interfere with other flows or propagate downstream. This consideration reduces the simulation event rate. The combination of these two opposite effects determines the fluid simulation event rate. Thus, as shown in Fig. 12, the simulation event rate can both increase and decrease with an increase in the service rate.

Another important factor with a direct impact on the fluid simulation event rate of cyclic network models is the link propagation delay. A smaller link delay will require less time for a flow rate change to propagate to the next queue and consequently, further downstream. This results in more flow rate changes per unit of time, causing an increase in the overall simulation event rate. Therefore, for a fixed service rate, the simulation event rate will increase as the link delay decreases, as shown in Fig. 12.

An interesting characteristic of the simulation results are the two discontinuities in the event rate at service rates 2 and 3. To understand this phenomena, recall that the probability that a rate change will interfere with other flows is given by the probability that the queue is either not empty *or* is empty but has an instantaneous aggregate arrival rate larger than the service rate. Now consider two service rates, one slightly less than 2, the other slightly greater, denoted by $(2 - \epsilon)$ and $(2 + \epsilon)$, where $\epsilon \ll 1$. The probability that a queue is empty is approximately the same for these two cases. Now

we consider the probability that aggregate arrival rate is greater than service rate. With a service rate of $(2 - \epsilon)$, the aggregate arrival rate is larger than the service rate when two or more sources are simultaneously on. In the four-queue cyclic network this probability is 11/16. When the service rate is $(2 + \epsilon)$, the aggregate arrival rate is larger than the service rate only when at least three sources are simultaneously on; the probability of this event is 5/16. Thus, the probability that a rate change interferes with other flows is much larger when the service rate approaches 2 from the left than from the right, resulting in the discontinuity in the simulation event rate evident in Fig. 12. The discontinuity at service rate 3 can be explained using similar arguments.

Finally, a direct comparison between fluid- and packet-level simulation event rates again reveals that fluid simulation is more efficient when the packet generation rate of the sources is sufficiently large. For the parameters we consider, fluid simulation outperforms its packet-level counterpart when $\gamma > 10$, as illustrated in Fig. 12.

We have further investigated the impact of the cyclic network topology on the fluid simulation event rate by considering a four-node tandem model that breaks the loop in Fig. 11. Our results (not shown here) show that the simulation event rate is considerably larger in the cyclic model, indicating that a significant fraction of the event rate is due to the topological nature of the model [8]. We have also investigated cyclic models with a larger number of queues and with flows traversing more than one queue. As expected, as flow interference becomes more pronounced, the simulation event rate increases [8].

5.3. Execution time and event rate

Thus far, we have used the simulation event rate as our sole measure of computational effort. In order to evaluate the appropriateness and accuracy of the event rate as indicator of computational effort, we consider the execution time required to simulate 1 s of simulation time. As with the simulation event rate, this value converges as the system enters steady state. Fig. 13 shows the measured simulation execution time of the four-node cyclic network for different parameters. The similarity between the plots for execution time (Fig. 13) and simulation event rate (Fig. 12) for both fluid- and packet-level models is striking, with the two plots exhibiting very similar trends, including the discon-

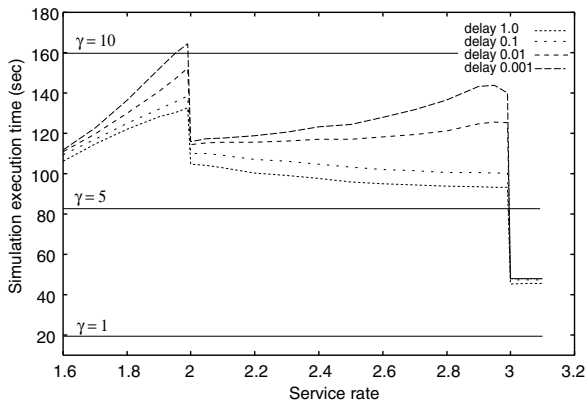


Fig. 13. Measured simulation execution time for cyclic network model.

tinuities and slopes. This suggests that the event rate defined in Eq. (1) is a good measure of the computation effort required by a simulation. Unfortunately, using the event rate as a sole metric of comparison has some drawbacks, as highlighted in the next section.

6. The Abilene network topology

In this section we compare the computational effort required by fluid- and packet-level simulation of a large-scale network. We use the topology of the Abilene network (an Internet provider for academic institutions) and a diverse traffic matrix with various source/destination pairs.

We consider the portion of the Abilene network² (also known as Internet2) that includes all of its core nodes as well as all nodes directly attached to the core, as shown in Fig. 14.³ Nodes that are not directly linked to a core node are excluded to simplify the simulations. Each node in the topology is either a core router (a total of 12) or an access router (a total of 64). Traffic sources and destinations are attached only to access routers. To model the traffic, we assume each access router has a fixed number, N , of identical on-off sources, each with a peak rate of $\gamma = 192$ packets/second. Packets have a fixed size of 1500 bytes, a common Internet MTU size. Each source repeatedly chooses a new random

destination (access router) using a uniform distribution over all possible destinations and generates traffic to that destination for a geometrically distributed number of on-off periods. Data flow in the network is routed using the shortest-path (in terms of hop count) between a source/destination pair. Each link in the topology corresponds to two infinite buffer capacity FIFO queues (one in each direction) with a service rate of either an OC48, OC12 or OC3 link as illustrated in Fig. 14. All links between core routers have a small propagation delay such that the one way propagation delay between an east-coast and west-coast router is about 30 ms.

In order to scale the offered load to the network, we increase the number of sources, N , attached to each access router. By assuming all sources are identically distributed and all access routers have the same number of sources, the value of N provides a simple, parsimonious model parameter for scaling load. The load ρ , as previously defined, at each queue can be computed exactly since routing is fixed and the traffic generated by each source is known. In particular, the load of a link l in the network is given by $N \sum_{r \in R_l} p_r \mu / (\lambda + \mu) \gamma / c$, where R_l is the set of routes (determined by all source/destination pairs) that traverse link l , p_r is the probability that route r is selected by the source, and c is the link capacity. Note that for access links, $\sum_{r \in R_l} p_r = 1$, which might not be the case for core links. Under these assumptions, access links have a higher average utilization than core routers for the same number of sources. This reflects current engineering practices of ISPs which usually over provision the core links of their network. Although there is spare capacity in the core, note that the access routers will become overloaded and unstable if the number of sources per access router is increased beyond 130.

Let us now compare the computational effort required by both fluid- and packet-level simulation using both the simulation event rate and the measured simulation execution time. Fig. 15(a) shows the simulation event rate of both approaches for a varying number of sources per access node, N . The insert shows the same results using a log-scale for the vertical axis. As expected, the simulation event rate for the packet-level simulation (dotted line) exhibits a linear increase, as increasing the number sources linearly increases the number of packets traversing the network. The packet-level simulation event rate can again be closely approximated using Eq. (14), with a value of K equal to the average number of hops between a randomly

² Details of the Abilene backbone can be found at <http://abilene.internet2.edu>.

³ The topology and link speeds considered here was the production network in September 2002.

tical axis. Again, we consider the execution time required to simulate 1 s of simulation time. Although the execution times shows a trend similar to their respective event rates, we can observe some important differences. First, the sharp increase in the fluid simulation event rate is much larger (about 2 orders of magnitude) than the sharp increase in execution time (about one order of magnitude). Second, when the number of sources is less than 100, the difference between the execution time of fluid- and packet-level simulation is less than one order of magnitude (as opposed to one and a half orders of magnitude in the event rate). Such differences were not noticeable in the previous sections, where the simulation event rate and execution time were much more closely related.

To help understand the causes of these differences we investigate the average execution time of an event as a function of the number of sources for both fluid- and packet-level simulations, as shown in Fig. 16. Note that the event execution time for packet-level simulation remains nearly constant as a function of the number of sources. However, the event execution time for fluid simulation can vary significantly with load. This variation in the execution time of an event (of up to a factor of 8) shows a limitation of using the simulation event rate as a measure of the computational effort.

In a packet-level simulation, events such as packet arrivals and departures have a constant computational cost independent of network load (assuming efficient event-list manipulation). Recall that all queues are FIFO and packets either enter the tail of the queue or are removed from the head of the queue; thus, no queue-length dependent com-

putation is required to process either event. Moreover, the amount of memory consumed by an arrival event is also constant, (as a single packet is allocated and placed in the queue) and every arrival event consists exactly of a single packet.

Unfortunately, the same is not true for fluid simulation. Below we qualitatively discuss the factors that lead to a load-dependent simulation event cost.

- *Arrival event:* Arrival events can occur simultaneously in a fluid queue. This happens when an upstream queue simultaneously changes the output rate of several flows and these flows enter the same downstream queue. Although each arrival rate change is counted as a single event, only a single fluid chunk (if any) will be created, and the cost of creating a fluid chunk containing i simultaneous arrivals will be less than i times the cost of processing a single arrival. Moreover, the amount of memory required by a fluid chunk is proportional to the number of arrival rate changes that it contains.
- *Departure event:* Departure events can also occur simultaneously in a fluid queue. Recall that when a fluid chunk is processed, all flows in that chunk will suffer a departure rate change. Other flows that are currently being serviced can also suffer a rate change as a consequence of processing the fluid chunk. Although each departure rate change will count as one event (as with arrivals), the cost of processing a batch of departure rate changes will not be linear in the number of departure rate changes.

The variable processing cost of fluid events becomes more apparent when simulating large-scale networks, where the number of flows traversing a queue can be very large. In the previous sections this variable cost was not noticeable, as the number of flows considered was relatively small. In the Abilene scenario considered here, a single core link (from node “Denver” to node “Kansas City”) can have up to 72,250 simultaneous flows (observed in our simulations when $N = 125$).

To quantitatively illustrate the variable cost of an event, we computed the average number of flows departing a queue and the average number of flows that form a fluid chunk. Note that this metric is queue dependent and we report on the average over all queues in the network. Fig. 17 shows the result as a function of the number of sources, N . Note that the average number of flows departing a queue

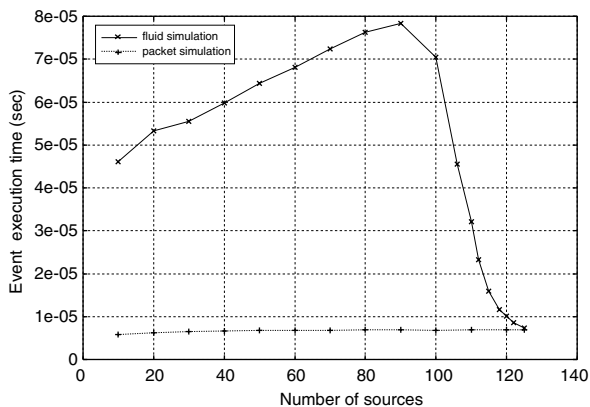


Fig. 16. Average execution time of an event in fluid- and packet-level simulations.

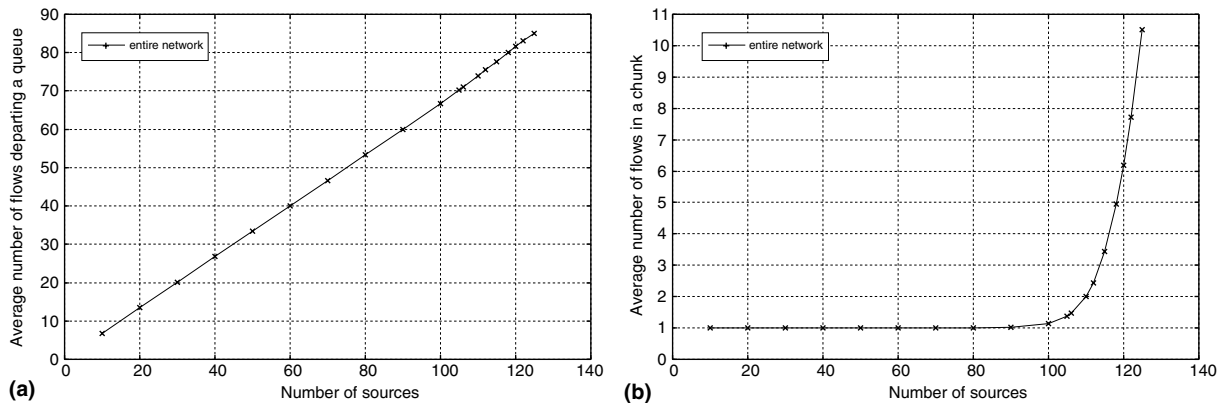


Fig. 17. (a) Average number of flows departing a queue. (b) Average number of flows that form a fluid chunk.

increases linearly (Fig. 17(a)) with an increasing number of sources. This is expected as this metric reflects the average offered load to a queue. More interesting is the behavior illustrated in Fig. 17(b). The average chunk size is one when the number sources is less than 90. Thus, for low utilization, a source rate change rarely interferes with any other flow in the network. As the utilization increases above 90 sources, the chunk size increases dramatically. This occurs because a source rate change now interferes with many other flows causing an arrival event in a downstream queue which itself is likely to have a large number of flows, which leads to a large fluid chunk. Note that this increase in fluid chunk size is independent of the ripple effect, and results only from high utilization. However, the ripple effect magnifies this problem as rate changes will propagate large fluid chunks to downstream queues.

The increase in the number of flows traversing a queue helps explain why the execution time of an event increases linearly when the number of sources is below 90 (Fig. 16). In contrast, the large number of events generated when the number of sources is above 90 helps explain why the execution time of an event decreases sharply (Fig. 16).

Another important consideration in a simulation is the amount of memory required. We observed that memory requirements have similar behavior to the simulation event rate, increasing linearly for packet-level simulation and sharply at higher utilization for fluid-level simulation [8]. This sharp increase in the memory requirements is a direct consequence of the sharp increase in the simulation event rate and on the fluid chunk size.

A simulation's execution time is a combination of the simulation event rate and the event execution

time. We have seen that at higher utilization, the ripple effect significantly increases the simulation event rate, which directly impacts the execution time. Our above discussion shows that event execution time in fluid simulation can depend on network utilization. This observation suggests that the simulation event rate may not be an accurate measure of a simulation's execution time in large-scale networks at very high loads.

7. Flow aggregation in fluid simulation

The examples in the previous sections indicate that flow interference and its consequence, the ripple effect, can significantly impact the efficiency of fluid simulation. In this section we show how flow aggregation can be used to reduce the fluid simulation event rate.

Flow aggregation, the bundling together background traffic into a single data flow, is a standard technique to reduce the complexity of network models. The number of flows that are aggregated into a single flow can vary and a network model can have several aggregated flows, each representing an aggregate of individual flows. In packet-level simulation, flow aggregation does not reduce the number of events that must be simulated, as the simulator must still keep track of each packet generated by the aggregate source. However, fluid simulation can benefit from flow aggregation since flow interference will be reduced when fewer flows are present in the model.

To illustrate the benefits of flow aggregation in fluid simulation, consider the example shown in Fig. 18. The figure shows three flows traversing a FIFO queue each with an arrival process corresponding to a_1 , a_2 and a_3 . We assume that

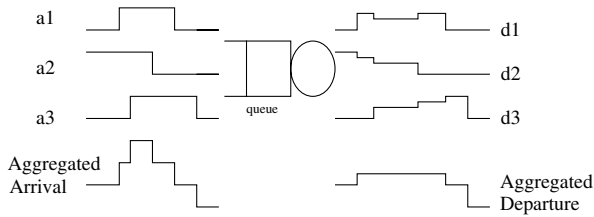


Fig. 18. Example of flow aggregation.

$h < c < 2h$, such that the queue accumulates a backlog when two or more sources are on. The departure processes of each flow are given by d_1 , d_2 and d_3 , respectively. Suppose we aggregate the three flows into a single flow. The arrival and departure process of the aggregate flow is also shown in Fig. 18. Note that the number of rate changes in the aggregated arrival process is equal to the sum of the rate changes in the arrival process of the individual flows. Thus, flow aggregation does not change the event rate of the arrival process. However, there are substantial reductions in the number of events in the departure process. In our example, the total number of rate changes in the departure process of the individual flows is 11, while the aggregated departure process exhibits only 3 rate changes.

There are two important factors that lead to a large reduction in the number of rate changes in the departure process of an aggregate flow. First, a single rate change in the departure process of the aggregated flow can represent many simultaneous rate changes in the model with individual flows. Second, the probability that fluid chunks merge in the queue increases. This occurs because aggregation bundles individual flows into a single larger flow, increasing the chances for fluid chunks (that were separate in the case of individual flows) to be merged.

To generalize the above considerations, we present a simple approximate analysis of the fluid simulation event rate when flow aggregation is used. Since the sources' contributions to the overall simulation event rate are the same in both the aggregated and non-aggregated scenarios, the following analysis only considers the event rate associated with the departure process.

Recall the single FIFO queue fed by N on-off sources and the result in Eq. (4). Ignoring the last term (β), the event rate of the departure process without flow aggregation can be approximated by

$$E_F = Ne^d = \frac{2\lambda\mu}{\lambda + \mu}(1 - \alpha + \alpha N)N. \quad (17)$$

Suppose we are interested in the behavior of one of the sources traversing this queue, say the N th source. We can then create a model in which the N th source shares the queue with an aggregate flow representing the other $N - 1$ sources. In this case, each rate change in the N th flow can cause one rate change in the aggregated flow, and each rate change in the aggregated flow can cause one rate change in the N th flow. Let α' denote the probability that at least one source in the aggregated flow is on, $\alpha' = 1 - (1 - \mu/(\lambda + \mu))^{N-1}$. Then, the total departure event rate in this aggregated model, E'_F , can be approximated by

$$E'_F = \left[\frac{2\lambda\mu}{\lambda + \mu} + \frac{2\lambda\mu}{\lambda + \mu}(N - 1)\alpha \right] + \left[\frac{2\lambda\mu}{\lambda + \mu}(N - 1) + \frac{2\lambda\mu}{\lambda + \mu}\alpha' \right]. \quad (18)$$

The first bracketed term is the departure event rate of the N th flow. The second bracketed term is the departure event rate of the aggregate flow, which represents the other $N - 1$ sources.

Eq. (17) shows that in the worst case, the departure event rate increases quadratically with the number of sources entering the queue. However, with flow aggregation the departure event rate increases linearly (Eq. (18)) with the number of sources being represented by the aggregate flow. We have also investigated the potential of flow aggregation using simulations and directly measuring the event rate. Experiments with the model presented above showed that the quadratic behavior in the event rate is indeed avoided when flow aggregation is used [8].

8. Weighted fair queuing in fluid simulation

All previous examples have assumed a FIFO queuing policy. As we have shown, the dynamics of the FIFO queuing policy can lead to a large amount of flow interference. In this section, we consider the Weighted Fair Queuing (WFQ) policy [7], which is designed to provide performance isolation among different classes. Intuitively, the isolation that is inherent in the scheduling policy itself should decrease the amount of flow interference, decreasing the fluid simulation event rate.

A WFQ node is formed by a set of K classes or queues, each having a FIFO policy. The service rate is distributed among the queues according to a set of positive partitions $\{\phi_1, \dots, \phi_K\}$, such that each

queue is guaranteed a minimum rate when it is backlogged. Since WFQ is work conserving, the node operates at full capacity as long as there is fluid stored in any of its queues. The details of the dynamics of a WFQ node can be found in [17].

Due to the work conserving nature of WFQ, the service rates allocated to each class can change dynamically. These changes occur when a queue requires less service rate than its assigned partition while some other queue could use more service rate than its assigned partition. We will refer to this dynamic allocation of service rates as the “service rate redistribution process”. Because of service rate redistribution, the service rate of a class depends not only on the state of its queue, but also on the states of all other queues. Also, note that a service rate change in a class can cause departure rate changes on all flows in this class, but also in flows currently being serviced in other classes.

The simulation event rate for a WFQ node will depend on the number of flows entering each class and on the frequency that the service rate is redistributed among the classes. For a WFQ node with low utilization, we expect the impact of the service rate redistribution process to be small.

To understand the impact of service rate redistribution, we measured the simulation event rate of a WFQ node. We considered 20 independent and identical on–off sources with $\lambda = \mu = 1$ and $h = 1$. The sources were equally allocated to the classes, so that each class was fed by approximately the same number of sources. The number of classes was varied from 1 to 10 and the service rate was distributed equally among the classes. The service capacity of the node was set such that loads of 0.5, 0.8, and 0.9 were obtained. The simulation event rate of this model is shown in Fig. 19 as a function of the number of classes.

When the number of classes is one, the WFQ node behaves as a single FIFO queue. Note that at loads of 0.9 and 0.8, there is a sharp decrease in the simulation event rate when a FIFO node (single class WFQ) is replaced by a 2-class WFQ node. The event rate further decreases as the number of classes in the WFQ node increases, since increased performance isolation among the classes reduces the amount of flow interference. However, as the number of classes increases beyond 7, the simulation event rate actually increases although the event rate to simulate the WFQ node with any number of classes is always less than the event rate to simulate a FIFO node with the same input flows. This occurs

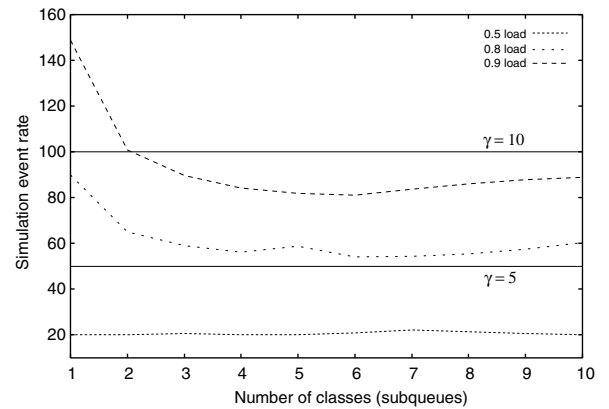


Fig. 19. Simulation event rate as a function of the number of classes in a WFQ node.

because the service rate redistribution process generates a larger number of events when the number of classes increases.

As expected, when there is no queuing in the system (which occurs when the service rate is greater than or equal to the sum of the peak rate of all the sources), the number of classes has no impact on the fluid simulation event rate as shown in Fig. 19 for a load of 0.5. We note that the inherent performance isolation among classes in WFQ does not change the packet-level simulation event rate from the FIFO case. In the above model, the packet-level simulation event rate is given by Eq. (6).

The relative efficiency of fluid- and packet-level simulation depends, as always, on the specific model parameter values, especially γ , the packet rate at the sources. To illustrate, Fig. 19 shows the packet-level event rate for $\gamma = 5, 10$. Note that fluid simulation can outperform packet-level simulation in some regions of the parameter space. A more detailed analysis of the WFQ fluid simulation event rate is available in [10], which also discusses the efficiency regions.

9. Conclusions

In this paper, we have analyzed the amount of computational effort required by fluid- and packet-level simulation by exploring several different network models and using the simulation event rate as a measure of computational cost. In fluid-level simulation, flow interference was found to be a fundamentally important property, inducing a “ripple effect” that leads to larger simulation event rates.

This phenomena is pronounced when considering large-scale models with high utilizations. In the case of packet-level simulation, the source's packet generation rates were shown to be the parameter of fundamental importance. The simulation event rate can be easily determined analytically and depends linearly on the number of packets generated. Thus, the relative efficiency between the two simulation approaches depends critically on the ripple effect and the packet generation rates. We showed that flow aggregation is an effective technique to reduce the amount of flow interference, reducing the impact of the ripple effect and resulting in improved efficiency for fluid simulation. Different scheduling policies, such as WFQ, can also have an impact on the ripple effect and can limit flow interference, consequently, improving the simulation event rate.

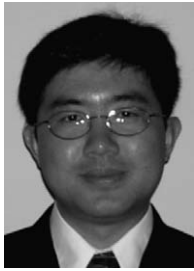
We also investigated the accuracy and adequacy of the simulation event rate as a measure of the computational cost of a simulation. In all cases investigated, the trends shown by the simulation event rate matched the trends shown by the execution time. However, for large-scale network models at high loads, the simulation event rate does not always accurately reflect the computational cost of the fluid simulation. This discrepancy is due to the inherent variability in the execution time of a fluid event.

References

- [1] J.S. Ahn, P.B. Danzig, Packet network simulation: speedup and accuracy versus timing granularity, *IEEE/ACM Transactions on Networking* 4 (5) (1996) 743–757.
- [2] D. Anick, D. Mitra, M.M. Sondhi, Stochastic theory of a data-handling system with multiple sources, *The Bell System Technical Journal* 61 (8) (1982) 1871–1894.
- [3] S. Bohacek, J.P. Hespanha, J. Lee, K. Obraczka, A hybrid systems modeling framework for fast and accurate simulation of data communication networks, in: *Proceedings of the ACM SIGMETRICS 03*, Jun 2003.
- [4] R. Brown, Calendar queues: a fast $O(1)$ priority queue implementation for the simulation event set problem, *Communications of the ACM* 31 (10) (1988) 1220–1227.
- [5] S3 Consortium, Scalable simulation framework API reference manual, Documentation Draft, January 1999.
- [6] J. Cowie, D. Nicol, A. Ogielski, Modeling the global Internet, *Computing in Science & Engineering* (1999) 30–38.
- [7] A. Demers, S. Keshav, S. Shenkar, Analysis and simulation of a fair queuing algorithm, *Internetworking Research and Experience* 1 (1990).
- [8] D.R. Figueiredo, B. Liu, Y. Guo, J. Kurose, D. Towsley, On the efficiency of fluid simulation of networks, Technical Report CMPSCI TR 04-82, Department of Computer Science, University of Massachusetts, Amherst, 2004.
- [9] Y. Gu, Y. Liu, D. Towsley, On integrating fluid model with packet simulation, in: *Proceedings of the IEEE INFOCOM 04*, March 2004.
- [10] Y. Guo, On fluid modeling of networks and queues, Ph.D. thesis, University of Massachusetts, Amherst, MA, 2000.
- [11] P. Heidelberger, Fast simulation of rare events in queuing and reliability models, *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 5 (1) (1995) 43–85.
- [12] G. Kesidis, A. Singh, D. Cheung, W.W. Kwok, Feasibility of fluid event-driven simulation for ATM networks, in: *Proceedings of the IEEE GLOBECOM 96*, 3 November 1996, pp. 2013–2017.
- [13] B. Liu, D.R. Figueiredo, Y. Guo, J. Kurose, D. Towsley, A study of networks simulation efficiency: fluid simulation vs. packet-level simulation, in: *Proceedings of the IEEE INFOCOM 01*, April 2001.
- [14] Y. Liu, F.L. Presti, V. Misra, D. Towsley, Y. Gu, Scalable fluid models and simulations for large-scale IP networks, *ACM Transactions on Modeling and Computer Simulation* 14 (3) (2004).
- [15] V. Misra, W.-B. Gong, D. Towsley, A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to red, in: *Proceedings of the ACM SIGCOMM 00*, September 2000.
- [16] D. Nicol, M. Goldsby, M. Johnson, Fluid-based simulation of communication networks using SSF, in: *Proceedings of the 1999 European Simulation Symposium*, Erlangen-Nuremberg, Germany, 1999.
- [17] A.K. Parekh, R.G. Gallager, A generalized processor sharing approach to flow control in integrated services network: the single-node case, *IEEE/ACM Transactions on Networking* 1 (3) (1993) 344–357.
- [18] D. Ros, R. Marie, Loss characterization in high-speed networks through simulation of fluid models, *Telecommunication Systems* 16 (2001) 73–101.
- [19] R. Rönngren, J. Riboe, R. Ayani, Lazy queue: an efficient implementation of the pending-event set, in: *Proceedings of the 24th Annual Symposium on Simulation*, 1991, pp. 194–204.
- [20] Mischa Schwartz, *Broadband and Integrated Networks*, Prentice-Hall, 1996.
- [21] M. Villén-Altamirano, J. Villén-Altamirano, RESTART: a straightforward method for fast simulation of rare events, in: *Proceedings of the 1994 Winter Simulation Conference*, Lake Buena Vista, FL, USA, 1994, pp. 282–289.



Daniel Ratto Figueiredo received the B.S. degree and the M.Sc. degree in Computer Science from the Federal University of Rio de Janeiro, Brazil in 1996 and 1999, respectively. He recently received the M.Sc. degree and the Ph.D. degree from the Department of Computer Science at the University of Massachusetts, Amherst (in 2005). His research interests include modeling and analysis of computer networks, and in particular, mechanisms used in the Internet.



Benyuan Liu received the B.S. degree in Physics from the University of Science and Technology of China (USTC) in 1994, and the M.S. degree in Physics from Yale University in 1998. He also received the Ph.D. degree in Computer Science from the University of Massachusetts Amherst in 2003. He is currently Assistant Professor at University of Massachusetts Lowell. His research

interests include the design and performance analysis of communication and computer networks.



Yang Guo received his B.A. degree from Shanghai Jiaotong University, and his Ph.D. degree in electrical and computer engineering from University of Massachusetts, Amherst in 2000. He is currently a member of technical staff at Thomson Corporate Research, Princeton, NJ. His research interests include

peer-to-peer networking and content distribution, real-time system and its scheduling, on-demand video streaming, and network modeling and performance evaluation. He is a member of ACM.



Jim Kurose received a B.A. degree in physics from Wesleyan University and his Ph.D. degree in computer science from Columbia University. He is a Professor (and past chairman) in the Department of Computer Science at the University of Massachusetts, where he is also Associate Director of the NSF Engineering Research Center for Collaborative Adaptive Sensing of the Atmosphere (CASA). Professor Kurose

has been a Visiting Scientist at IBM Research, INRIA, Institut EURECOM, and the University of Paris, LIP6.

His research interests include network protocols and architecture, network measurement, sensor networks, multimedia communication, and modeling and performance evaluation. He has served as Editor-in-Chief of the IEEE Transactions on Communications and was the founding Editor-in-Chief of the IEEE/ACM Transactions on Networking. He has served as Technical Program Co-Chair for IEEE Infocom, ACM SIGCOMM, ACM SIGMETRICS, and the ACM Internet Measurement conferences.

He has received a number of awards for his teaching and educational activities, including the IEEE Taylor Booth Education Medal. He is a Fellow of the IEEE and the ACM.

With Keith Ross, he is the co-author of the textbook, Computer Networking, a Top Down Approach Featuring the Internet, published by Addison-Wesley, Longman.

Don Towsley holds a B.A. in Physics (1971) and a Ph.D. in Computer Science (1975) from University of Texas. From 1976 to 1985 he was a member of the faculty of the Department of Electrical and Computer Engineering at the University of Massachusetts, Amherst. He is currently a distinguished Professor at the University of Massachusetts in the Department of Computer Science. He has held visiting positions at IBM T.J. Watson Research Center, Yorktown Heights, NY; Laboratoire MASI, Paris, France; INRIA, Sophia-Antipolis, France; AT&T Labs—Research, Florham Park, NJ and Microsoft Research Lab, Cambridge, UK. His research interests include networks and performance evaluation.

He currently serves as Editor-in-Chief of IEEE/ACM Transactions on Networking and on the editorial boards of Journal of the ACM, and IEEE Journal on Selected Areas in Communications and has previously served on numerous other editorial boards. He was the Program Co-chair of the joint ACM SIGMETRICS and PERFORMANCE'92 conference and the Performance 2002 conference. He is a member of ACM and ORSA, Chair of IFIP Working Group 7.3, and Chair of the Computer Performance Foundation.

He has received the 1998 IEEE Communications Society William Bennett Best Paper Award and numerous best conference/workshop paper awards. Last, he has been elected Fellow of both the ACM and IEEE.