Fluid Simulation of Large Scale Networks: Issues and Tradeoffs [*]

Benyuan Liu, Yang Guo, Jim Kurose, Don Towsley, Weibo Gong
University of Massachusetts
Amherst, MA, U.S.A.

CMPSCI Technical Report TR 99-38

December 22, 1999

Abstract

Traditional discrete-event packet-level approaches to simulating computer networks become computationally infeasible as the number of network nodes or their complexity increases. An alternative approach, in which packet-level traffic sources are replaced by fluid sources, has been proposed to address this challenge. In this paper we compare the amount of computational effort needed to simulate a network using a packet-level approach versus a fluid-based approach. We quantitatively characterize the amount of computational effort needed by each approach using the notion of a simulation's event rate, and derive expressions for the event rate of a packet and fluid flow at both the input and output sides of a queue. We show that fluid simulation can require less computational effort for simple networks. However, as the network size and complexity grow, the so-called "ripple effect" can result in fluid simulations becoming more expensive than their packet-level counterparts. This suggests that time-driven (approximate) fluid simulation techniques may be needed to efficiently simulate large scale networks.

Keywords: discrete event simulation, fluid simulation, performance analysis, large scale network simulation

[*]This research was been supported in part by NSF under awards ANI-980933 and CDA-9502639.

1 Introduction

Data communication networks have been experiencing tremendous growth over the last decade, in terms of both size and complexity. Consequently, evaluating the performance of such networks is becoming an increasing difficult problem. The traditional packet-level approach to simulating a network is to simulate the arrival, queueing, processing and departure of each individual packet at the various queues along a packet's path through the network. However, as the number of network nodes becomes large, this approach begins to become computationally infeasible. Thus, while packet-level simulation may be easy to implement and can yield accurate results for small networks, more scalable simulation methodologies and techniques are needed to simulate truly large scale networks.

Several efforts have been undertaken to address this important challenge. Scalable simulation frameworks[2] and parallel simulation techniques[3, 6] have been developed to speed up a simulation by taking advantage of the computing power of multiprocessors and distributed computer networks.

An alternative to devoting more computing resources to a traditional packet-by-packet simulation is to change the underlying simulation paradigm itself. One can approximately model a packet source in the hope of speeding up a simulation without sacrificing significant accuracy. In the packet-train method[1], speedup is achieved by coarsening the representation of network traffic from a packet-level granularity to a "train"-level granularity in which a cluster of closely-spaced packets is replaced by a single "packet train". Rather than having to simulate the arrival and departures of the many individual packets in the train at the various network queues, the simulation treats the packet train as a monolithic entity and only simulates the arrivals and departures of the train - a potential savings in computational effort.

Fluid simulation[4, 5] takes this approach one step further, using a fluid flow rather than a packet-by-packet flow to represent a traffic source. In a fluid simulation, a source's fluid flow rate may change as the packet generation rate in the system being modeled changes. Small time-scale variations in the packet arrival stream are abstracted out of the source model by having the fluid's rate remain constant between these changes in the fluid rate. As the fluid flows through the network, a fluid simulator need only keep track of the rate changes that occur as a result of queueing, multiplexing, and service of the fluids at the various queues; an equivalent packet-level simulation would need to keep track of a potentially enormous number of packets in the network. When a packet-level source is modeled as a fluid source, and the fluid's rate (and

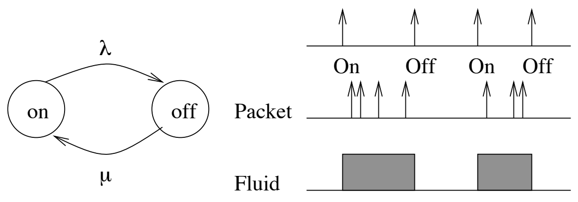

Figure 1: The Markov Modulated Packet and Fluid Sources

rate changes) is simulated and tracked exactly at each point in the network, we will refer to such a simulation as an exact fluid simulation (EFS). There are also variations and extensions to EFS that make additional approximations in the fluid simulation model, such as time-driven fluid simulation[7, 8] and time-stepped hybrid simulation, where networks can be evaluated at multiple time scales. We will discuss these techniques in our concluding remarks.

In this paper we compare the amount of computational effort needed to simulate a network using a packet-level approach versus an exact fluid approach. We note that our goal here is *not* to compare the relative accuracy of packet-level versus fluid simulation of networks (see [7, 8]) but rather to compare the amount of computational effort needed by these two different approaches. We quantitatively characterize the amount of computational effort needed by each approach using the notion of a simulation's event rate, and derive expressions for the event rate of a packet and fluid flow at both the input and output sides of a queue. We show that fluid simulation can require less computational effort for simple networks, due to the high level of abstraction of the traffic source and the possibility of merging of chunks of fluid at network queues. However, as the network size and complexity grow, the so-called "ripple effect" can result in fluid simulation becoming more expensive. This suggests that time-driven (approximate) fluid simulation techniques may be needed to efficiently simulate large scale networks.

The remainder of the paper is structured as follows. Section 2 describes the packet and fluid source models and the fluid simulation of simple FIFO multiplexing. Section 3 first derives the simulation event rate of a single node under packet-level simulation and EFS. We then extend the event rate analysis to a feed-forward network and then quantitatively compare the simulation event rates for a tandem queueing system under packet-level simulation and EFS. Concluding remarks and directions for future research are presented in Section 4.

2 Source Models and FIFO Multiplexing

We consider a simple network model containing traffic sources and interconnected work-conserving FIFO queues. The dynamics of these components are described below.

2.1 Markov-Modulated Fluid Source and Packet Source

We use a stationary, continuous-time Markov chain with two states, *on* and *off*, to modulate a source's generation of packets or fluid. The transition rates from the *on* state to the *off* state, and from the *off* state to the *on* state are λ and μ, respectively, with the holding time in each state assumed to be exponentially distributed.

For a packet source, when the source is in the *on* state, packets are emitted according to a Poisson process with rate γ. Packet sizes form an iid sequence of random variables $\{x_i\}$, where $E[x_i] = \overline{x}$. No packets are emitted when the source is in the *off* state.

For a fluid source, when the source is in the *on* state, fluid is emitted at a constant rate, and when the source is in the *off* state no fluid is emitted. Since the fluid source approximates a packet source, we require that the fluid rates be $\gamma\overline{x}$ in *on* state and 0 in the *off* state, in order for the packet and fluid sources to have the same average data rate.

The packet and fluid sources are illustrated in Figure 1, and will be referred to as Markov Modulated Packet Source (MMPS) and Markov Modulated Fluid Source (MMFS), respectively.

2.2 Dynamics of work-conserving FIFO queue

A FIFO queue serves queued traffic on a First In First Out manner. This is straightforward for packet simulation, since incoming packets are queued according to their arrival times, receive service, and leave in that order.

For fluid simulation, fluid that arrived earlier in time will be served earlier. However, with fluids (unlike packets), fluid from different sources can arrive simultaneously at the queue. Let us examine this issue in more detail. Assume there are N fluid sources connected to a FIFO queue, with a service rate of c. Let $a_k(t)$ and $d_k(t)$ be the fluid arrival rate (amount of fluid arriving per unit time) and fluid departure rate of the $k-th$ flow at time t. The overall arrival rate to the queue is $a(t) = \sum_{k=1}^{N} a_k(t)$. Suppose the aggregate flow's input rate changes at times $\tau_1 < \tau_2 < \cdots < \tau_N$. Let $U(t)$ denote the backlog in the queue at time t, for $\tau_i \leq t \leq \tau_{i+1}$; this can be computed using the following recursive formula:

$$U(t) = max(0, U(\tau_i) + (a(t) - c)(t - \tau_i)) \tag{1}$$

For each rate change, there are two possible scenarios:

• $U(\tau_i) = 0$. In this case, a rate change in the input at τ_i will result in an immediate change in the output rate, since no fluid is backlogged. If $a(\tau_i) \leq c$, the output rate of each source is just its arrival rate, $d_k(\tau_i) = a_k(\tau_i)$. If the aggregate arrival rate is larger than the FIFO's bandwidth, each source receives a service rate proportional to its arrival rate, $d_k(\tau_i) = \frac{a_k(\tau_i)}{a(\tau_i)}c$.

• $U(\tau_i) > 0$. In this case, the backlogged fluid present at time τ_i must first be consumed before the input rate change at τ_i is reflected in the output rate. If the new aggregate arrival rate is greater than the FIFO's bandwidth, once the backlog $U(\tau_i)$ is consumed, the departure rate will be $d_k(\tau_i + \frac{U(\tau_i)}{c}) = \frac{a_k(\tau_i)}{a(\tau_i)}c$.

If $a(\tau_i) \leq c$, during the time when the FIFO serves the backlogged fluid $U(\tau_i)$, fluids which arrive after would be accumulated in the queue. So the corresponding departure rate of source k would be, $d_k(\tau_i + \frac{U(\tau_i)}{c}) = \frac{a_k(\tau_i)}{a(\tau_i)}c$. Note that $d_k(\tau_i + \frac{U(\tau_i)}{c}) > a_k(\tau_i)$, i.e., that the departure rate of the fluid that arrived at τ_i is actually greater than the arrival rate at τ_i. Note also that if the rates haven't changed when the FIFO queue consumes all the newly backlogged fluids, i.e., $\tau_{i+1} > \tau_i + \frac{U(\tau_i)}{c} + \frac{a(\tau_i)\frac{U(\tau_i)}{c}}{c-a(\tau_i)}$, the departure rate for each source will drop to its instantaneous arrival rate at that moment.

3 Analysis of Fluid Simulation vs. Packet Simulation

Having described the *on-off* packet and fluid sources, as well as the functionality of the FIFO queue, we are now in the position to study the simulation event rates associated with a variety of fluid and packet queues.

3.1 Single node case

Let us first consider a system consisting of a single source and a single FIFO queue. Packets (in the case of a packet simulation) or fluid chunks (in the case of a fluid simulation) - a continuous flow with a constant rate - are fed into the FIFO server, receive service and then leave the system. The server is assumed to have an infinite size buffer; no packet or fluid loss will be considered. We also assume that the packet or fluid arrival rate (when the source is *on*) is greater than the FIFO's service rate, as otherwise there will be no queueing.

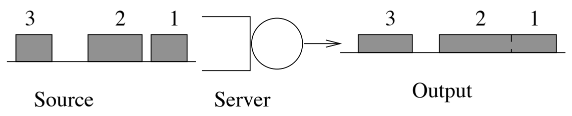

Figure 2: The aggregation of the fluid chunks

The simulation of the *on-off* source produces two types of events, *source on* and *source off*, corresponding to transitions from *off* to *on* and *off* to *on*. We define an *on-off* cycle to be the time between two successive *source on* events. It has an average length of $(\lambda + \mu)/\lambda\mu$. Each cycle consists of exactly two events, *source on* and *source off*, and thus the event rate associated with the *on-off* source transitions is $2\lambda\mu/(\lambda + \mu)$ events per unit time.

The above analysis applies to both packet and fluid simulations. For packet simulation, we identify two additional events, packet arrivals and packet departures at a queue, that must be handled by the simulation. Recall that when the source is in the *on* state, the packet arrival rate is γ. For each packet, the simulation must process an associated arrival and eventual departure event at the FIFO queue. Therefore, the corresponding event rate is $2P(on)\gamma = 2\mu\gamma/(\lambda + \mu)$, where $P(on)$ is the probability that the source is in *on* state. We use e^p to denote the total event rate of the packet-level simulation. Given the above discussion, we have

$$e^p = \frac{2\lambda\mu}{\lambda + \mu} + \frac{2\mu\gamma}{\lambda + \mu} \tag{2}$$

In fluid simulation, the *source on* event of the information source indicates an arrival of a fluid chunk and the corresponding *source off* event marks the end of the fluid chunk. We observe that a fluid arrival always coincides with a *source on* event. Thus, in this case, we only need one additional event - a fluid chunk departure event. When we process the *source off* event, we schedule the next *source on* event, and we compute whether the departure event of the current fluid chunk will happen before the arrival of the next fluid chunk. If at the arrival of the next fluid chunk, the current fluid chunk has left the server, then the departure event of the fluid chunk is scheduled and inserted in the event list. Otherwise, the two adjacent fluid chunks are aggregated into one fluid chunk, and we wait until the *source off* event of the next chunk to determine whether we need to schedule a departure event for this merged chunk.

Figure 2 illustrates this chunk aggregation process. Fluid chunk 2 arrives before fluid chunk 1 leaves the server, and hence no departure event is scheduled for chunk 1. Instead fluid chunk 2 joins chunk 1 to form a merged chunk and the departure event is delayed. When fluid chunk 2 turns off, the computation shows that the service of the merged chunk will be completed before the arrival of fluid chunk 3. A departure event is thus scheduled.

We have observed that not every fluid chunk will have a corresponding departure event. Thus, the event rate associated with the departure of the fluid chunks has an upper bound of $\lambda\mu/(\lambda+\mu)$. The total event rate from this simple single-source single-queue fluid model, denoted by e^f, is the sum of the event rate for the *on-off* source and the event rate for the departure of the fluid chunks,

$$e^f = \frac{2\lambda\mu}{\lambda+\mu} + \alpha\frac{\lambda\mu}{\lambda+\mu} \tag{3}$$

where $\alpha \in [0,1]$, represents the probability that a fluid chunk departs without being merged with the next fluid chunk, i.e., it completes service before the arrival of the next fluid chunk.

Comparing equations 2 and 3, we see that the first term in the packet and fluid event rates (respectively) are identical, as they come from the simulation of the two-state Markov chain. In the worst case (i.e., the case that maximizes work needed under EFS), the coefficient α in equation 3 can reach 1, in which case there is no aggregation of the fluid chunks and each chunk has its own departure event. In this worst case, if $\gamma > \frac{\lambda}{2}$, the event rate of the fluid simulation is smaller; otherwise, packet-level simulation has fewer events. Note that as the arrival rate approaches the queue's capacity, more and more fluid chunks will be merged, and the computational advantages of fluid simulation over packet simulation will increase.

We observe that two factors reduce the event rate of fluid simulation over that of a corresponding packet-level simulation. First, a single fluid chunk represents all the packets emitted in the *on* state, which can save a large number of events when the packet emission rate is high. Secondly, the number of events in fluid simulation may be further reduced by the merging of adjacent fluid chunks.

3.2 Network case

In this section we extend our analysis of the event rate of fluid and packet simulation to more complicated networks. We will observe that the performance (measured in terms of the event rate) of fluid simulation begins to suffer from the so called *ripple effect* (first observed in [4]) when we introduce multiple sources into system.

Figure 3 illustrates the ripple effect in a single queue with three sources. As noted in the previous section, for a work-conserving FIFO queue, a rate change of one source may cause the

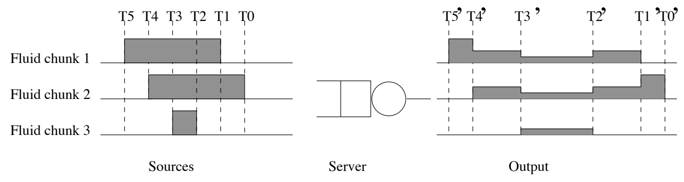

Figure 3: An example of ripple effect

output rate of the other sources feeding the same queue to change. In Figure 3, for example, the arrival of fluid chunk 1 at time $T1$ causes fluid chunks 1 and 2 to share the server capacity during the interval $[T1', T2']$. Similarly, the arrival of a *single* fluid chunk 3 at time $T2$ causes a change in the output rate for all *three* chunks at time $T2'$. Thus, as a flow progresses into the network, it can be characterized by a event rate that is much *larger* than that of the original source.

In the following we first analyze a single queue with multiple arrival processes. We then consider a tandem queue example to illustrate how the ripple effect can dramatically increase the event rate of EFS. Let us begin with some terminology and notation.

- event: a rate change;
- $E_i(t)$: # events occurring in $[0, t]$ for arrival process i;
- e_i: the event rate of arrival process i. $e_i = \lim_{t\to\infty}(E_i(t)/t)$;
- s_i: the number of distinct rates for arrival process i;
- e_i^d: the event rate of the departure process i;
- s_i^d: the number of distinct rates for departure process i;

Figure 4 depicts a snapshot of a FIFO queue with two arrival processes. We define an aggregate chunk of fluid to contain chunks of fluid from sources 1 and 2, with a rate vector $[r_1, r_2]$ that does not change over the length of the aggregate fluid chunk. We assume that processes 1 and 2 never simultaneously change their rates. Thus, every event in process 1 and process 2 leads to a new chunk. In Figure 4, different shading represents queued aggregate chunks of fluid with different rate vectors. We number the aggregate chunks in the order of their creation. Note aggregate chunks 0, 4, and 8 contain no fluid and are called "empty chunks". We denote other aggregate chunks as "non-empty chunks".

Using the analysis presented in Section 2, we know a "non-empty chunk" will lead to at

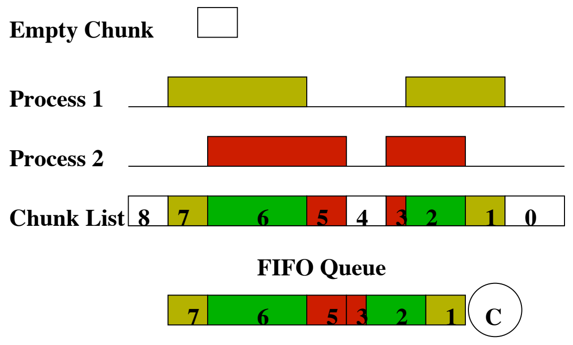

Figure 4: A Two Arrival Process FIFO Queue

least one event (rate change) in the departure process in the future. Under certain conditions an aggregate chunk can actually lead to *two* events in the departure process. We denote such a aggregate "chunk" as a "special aggregate chunk". The conditions that produce a "special aggregate chunk" are as follows: 1) the aggregate arrival rate must be less than the service rate; 2) the queue must not be empty when a "special aggregate chunk" is formed at the queue's input; 3) there are no rate changes until the server completely empties the queue (at which point the fluid departure rates equal the fluid arrival rates). More precisely, during the time when the server serves the backlog found by an arriving special aggregate chunk, the fluid of the special chunk accumulates in the queue. The server will then eventually serve the accumulated fluid from the special chunk at full capacity (even though the aggregate arrival rate of the special chunk was less than the service rate.) When the server completely empties the queue, the fluid departure rate then changes from the full rate of the server to the aggregate arrival rate of the input. In such cases, there are thus *two* rate changes in the departure process of a special aggregate chunk - one occurs when the special chunk begins service; the other occurs when the server empties the queue and aggregate departure rate equals the aggregate arrival rate.

For the "empty chunk", there are the following two scenarios:

• Upon the arrival of non-empty chunk following the empty chunk, the queue is empty. In this case, "empty chunk" will lead to one event in the departure process.

• When the non-empty chunk following the empty chunk arrives, there is backlogged traffic in the queue. In this case the empty chunk "disappears" in the queue (since the newly arriving chunk will be queued directly behind the last backlogged aggregate chunk) and will not cause an event in the departure process. Moreover, if the rate vector of the arriving chunk is the same as that of the chunk at the end of queue, these chunks will merge and the arrival will not produce an event in the departure process.

To simplify the analysis, we assume that every empty chunk will introduce a single event and do not consider the merging of chunks.

Consider the system with two arrival processes as shown in Figure 4. Every arrival event of process 1 creates a new aggregate chunk and, thus, a future rate change in its departure process. A rate change in process 2 has no impact on process 1 when process 1 is in the *off* state. However, when process 1 is *on*, an event of process 2 changes the state vector of the newly generated aggregate chunk and thus will affect the state of process 1's departure process in the future. Moreover, if the newly generated chunk is a special chunk, it will introduce yet a second event in the departure process, as discussed above.

We can reduce a system with N incoming flows to a system with 2 flow problem by aggregating flows. We assume that the incoming flow's event rate, number of states, and α_i are known, where α_i is the probability that process i is not *off,* observed at moments of rate changes for all aggregated flows except flow i. We summarize the above discussion and have the following Proposition:

Proposition 1 *Consider N flows feeding a fluid queue. Then:*

$$
\begin{aligned}
e_i^d &= e_i + \alpha_i \cdot \sum_{j \neq i} e_j + \psi_i, &(4)\\
s_i^d &\leq 1 + (s_i - 1) \cdot \prod_{j \neq i} s_j. &(5)
\end{aligned}
$$

for all $i \in \{1, 2, \cdots, N\}$.

where ψ_i is the extra event rate caused by special chunks.

Note that we have made the assumptions that 1) an empty chunk always leads to a event; and 2) no chunks are merged. These assumptions result in an overcount of the number of simulation events and our computed event rates are consequently an upper bound. Assumption (1) is violated only when an event results in all flows having a zero flow arrival rate, and the following non-empty aggregate chunk finding a non-empty queue. The probability of this happening may be non-negligible when the the number of flows is small, but will be negligible when the number of incoming flows is large.

Our assumptions also ignore the merging of chunks. We argue that the opportunity to merge two chunks occurs infrequently. Two successive aggregate chunks can be merged only when they

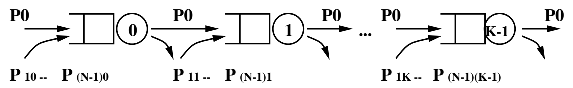

Figure 5: A Tandem Queue

having the same state vector. This scenario can only occur when an event generates a chunk, the next event results in all flows having a zero flow arrival rate, and a third event generates a new aggregate chunk that has the same rate vector as the previous one. As discussed above, the probability of this happening is likely to be small.

For EFS from Proposition 1 we observe that the number of states increases exponentially as the number of sources increases, and the event rate for each session will increase in proportion to the event rate of aggregated traffic other than itself. As the scale of the network grows, such an effect could be overwhelming. The event rate for fluid simulation can easily exceed that of packet level simulation. To illustrate this, we present the following tandem queue example.

Tandem queue example:

Consider a tandem network as showed in Figure 5. For each node in the network there are N input processes. Process 0, a MMFS at node 0, traverses through all the nodes. The other $(N-1)$ processes, which are all MMFS sources, leave the system after passing through one node.

We assume that all of the sources have the same transition rates λ and μ. We use an additional subscript i to represent the node id. For example, we use α_{0i} to denote the *not-off* probability at node i for process 0. Thus:

$$\alpha_{00} = \frac{\mu}{\lambda + \mu}.$$

The output process of queue 0 is likely to be quite complicated, making the calculation of α_{01} difficult. Thus we use the value of α_{00} to approximate α_{0i}, for $i > 0$.

Proposition 1 yields the following for the event rate of departure process 0 at node 0:

$$e_{00}^{d} = \frac{2\lambda\mu}{\lambda + \mu} + (N-1)\alpha_{00}\frac{2\lambda\mu}{\lambda + \mu} + \psi_{00}.$$

For departure process 0 at node $i \geq 1$, we have the following recursive equation by applying

Proposition 1:

$$
\begin{aligned}
e^d_{0i} &= e^d_{0(i-1)} + \alpha_{0i} \cdot \sum_{j=1}^{N-1} e_{ji} + \psi_{0i} \\
&= \frac{2\lambda\mu}{\lambda+\mu}(1 + (N-1)\alpha_{00} \cdot (i+1)) + \sum_{j=0}^{i} \psi_{0j}. \qquad (6)
\end{aligned}
$$

Let $R^f(K)$ and $R^p(K)$ denote the simulation event rates for a K node tandem system using fluid simulation method and packet level simulation method, respectively. We exclude the event rate simulating the source since the cost for both methods is the same. Let $r^f(i)$ denote the event rate to simulate ith node in the fluid simulation. Then:

$$r^f(0) = N\frac{2\lambda\mu}{\lambda+\mu} \qquad (7)$$

and for all $i \geq 1$:

$$r^f(i) = e^d_{0(i-1)} + (N-1)\frac{2\lambda\mu}{\lambda+\mu} \qquad (8)$$

Thus the event rate for fluid simulation is:

$$R^f(K) = \sum_{i=0}^{K-1} r^f(i) \qquad (9)$$

Recall that γ denotes the packet emission rate in the *source on* period. Since the packet simulator schedules two events for each packet (a packet arrival and a packet departure), the event rate for simulating one node in a packet level simulation is $N \cdot \frac{2\mu\gamma}{\lambda+\mu}$. Thus the event rate to simulate the K node tandem queue is:

$$R^p(K) = NK \cdot \frac{2\mu\gamma}{\lambda+\mu}. \qquad (10)$$

Figure 6 depicts $R^p(K)$ and $R^f(K)$ with γ equal to 10, $\lambda = \mu = 1$, and $N = 10$. So:

$$R^f(K) = 2.25K^2 + 7.75K + \sum_{i=0}^{K-1}\sum_{j=0}^{i-1} \psi_{0j}. \qquad (11)$$

and

$$R^p(K) = 10K\gamma. \qquad (12)$$

We ignore the effects of special aggregate chunks, i.e., $\psi_{0i} = 0$ for all i. $R^p(K)$ is thus a linear function with slope of 10γ. $R^f(K)$ is a parabola. When K is small, fluid simulation has a smaller

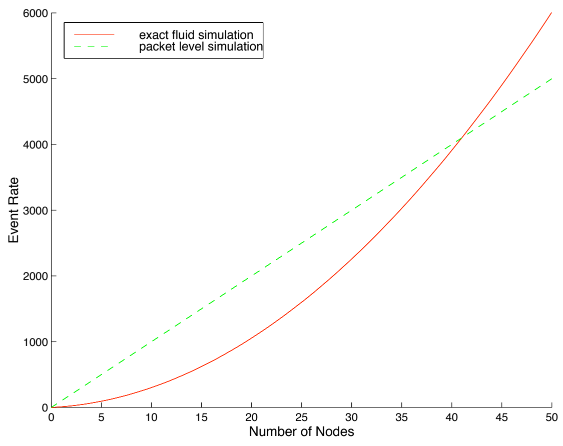

Figure 6: Event Rate For K-Node Tandem Queue Simulation

event rate than packet-level simulation, due to the fact that one fluid chunk represents multiple packets. However as K increases, the ripple effect eventually (at around 41 nodes) causes the fluid simulation to have more events that its packet-level counterpart. The intersection point can serve as a guideline to select the proper simulation methodology for the simulation of different sized systems.

The multi-queue system we have studied is a simple tandem queueing system. We conjecture that the ripple effect will be even more pronounced in a network with downstream branching and multiple input nodes. In such networks, a change in the output event rate at one queue will be propagated to all downstream nodes, generating changes in the fluid service rates of flows at those nodes. These rate changes in these flows, in turn, will effect all downstream flows with which they interact.

4 Conclusion and future work

In this paper, we have seen that the high degree of abstraction of fluid model over packet model and the possible aggregation of adjacent fluid chunks can greatly speedup a fluid simulation over its packet-level counterpart. However, we have also seen that the ripple effect can also play a significant negative role in fluid simulations, particularly in more complex models. These two competing factors will thus determine the relative event rates of fluid and packet-level simulation. For simple networks and high source output rates, fluid simulation is likely to outperform packet-level simulation in terms of event rate. For large scale networks, the number of rate changes caused by the ripple effect may become so large that the packet-level simulation has fewer simulation events.

One possible direction for future research is to compare fluid and packet-level simulation under scheduling disciplines other than FIFO. We believe that weighted fair queueing may be particularly well-suited for fluid simulation, since the isolation among traffic flows provided by WFQ will avoid introducing ripple effects.

A second direction for future research is to explore approximate fluid simulation methods. The idea of fluid simulation is to model the network traffic at a coarser resolution than packet-level simulation. In this sense, the Markov on-off fluid source is a coarse grain model of the discrete packet arrival process. When modeling a source as on-off fluid, we have already committed to a certain degree of approximation. The fluid simulation approach discussed in this paper, which we referred to as the *exact* fluid simulation, exactly simulates the event interactions among the on-off fluid processes at the queues in the network. Indeed, this was precisely the cause of the difficulties created by the ripple effect. One promising solution, then, is a time-driven fluid simulation scheme, in which the continuous flow is discretized into fixed length of fluid chunks, with each chunk having a constant rate and representing many packets. This is done not only at the source, but at *all* queues in the network. This helps reduce the number of rate changes resulting from the ripple effect. Also, since the discretization interval (the time-step) for all the nodes is the same, the simulation is time-driven and parallelism is much easier. The time-steps can be adjusted to accommodate different resolution requirements in modeling. This scheme is undergoing both theoretical and experimental investigation.

References

[1] Jong Suk Ahn and Peter B. Danzig "Packet Network Simulation: Speedup and Accuracy Versus Timing Granularity", *IEEE/ACM Transactions on Networking* pp. 743-757, **4**, No. 5, Oct., 1996.

[2] J. Cowie, D. Nicol and A. Ogielski "Modeling the Global Internet", *Computing in Science & Engineering* , pp. 30-38, 1999.

[3] R.M. Fujimoto, "Parallel Discrete Event Simulation", *Communications of the ACM*, pp. 30-53, **33**, No. 10, Oct., 1990.

[4] G. Kesidis, A. Singh, D. Cheung and W.W. Kwok "Feasibility of Fluid-Driven Simulation for ATM Network" *IEEE GLOBECOM*, pp. 2013-2017, **3**, November, 1996.

[5] K. Kumaran and D. Mitra "Performance and Fluid Simulations of a Novel Shared Buffer Management System", *Proc. IEEE INFOCOM 98*, March 1998

[6] D. Nicol and P. Heidelberger "Parallel Execution for Serial Simulators", *ACM Trans. on Modeling and Computer Simulation*, **6**, no. 3, pp.210-242, July 1996.

[7] A. Yan and W.B. Gong "Fluid Simulation for High Speed Networks", To appear in *IEEE Trans. on INformation Theory*, June, 1999.

[8] A. Yan and W.B. Gong "Time-Driven Fluid Simulation For High-Speed Networks With Flow-Based Routing", *Proc. of the Applied Telecommunications Symposium '98*, pp. 153-158, Boston, MA, April, 1998