Variable Step Fluid Simulation for Communication Network

Hongjoong Kim[1] * and Junsoo Lee[2]

[1] Korea University, Seoul, Korea,
hongjoong@korea.ac.kr
[2] Sookmyung Women's University, Seoul, Korea,
jslee@sookmyung.ac.kr

Abstract. We propose a variable step fluid model for communication network in this paper. Our main goal in this research is simulation speedup of a packet-level simulator while maintaining the accuracy. The variable step fluid model not only reduces complexity but also accurately estimates simulation details such as round trip time, queue sizes, TCP windows, and packet drops. In addition, the variable step fluid model reduces event explosions, ripple effects, which have been observed in the traditional fluid models. We validate our model against `ns-2` simulation with a mixture of TCP and UDP flows under various background traffic scenarios. Our model achieves significant speedup compared to packet-level simulators. For example, the speedup of our fluid model for 20 Mb bottleneck is 40 to 70 against `ns-2`.

1 Introduction

Packet-level simulators have been widely used for performance evaluation of communication network because of their accuracy. However, packet-level simulators do not scale to large network or high bandwidth because they track every event in the system. Simulation of network traffic in a packet-level simulator such as `ns-2`[1] considers arrivals and departures of each packet at all routers and queues between a source and a destination. As the topology of the network becomes complicated and as the size of network connections increases, packet-level simulators show significant performance degrade.

Many methods have been proposed to speed up packet-level simulators. One method to overcome such an event explosion in a packet-level simulation is to use fluid models[2–7], which abstract discrete packets with a continuous fluid flow. These fluid models average out small variations in packet-level assuming that a network traffic can be considered as a continuous flow rather than discrete packets.

While traditional fluid models [3, 5] reduce complexity of packet-level simulators, they have several drawbacks. Since packet events are discrete in nature,

* This research is supported by the MIC, under the ITRC support program supervised by the IITA.

a fluid model may lose accuracy against packet model, especially if we compare short term behaviors. Secondly, when several flows go through a network connection link, a traditional fluid model [3] often increases complexity because of the event explosion called "ripple effect". A ripple effect has been introduced in [3]. Liu et. al in [3] assume flow rates are held constant between some events, but if these events occur too frequently, performance of a fluid model is worse than that of packet models due to event explosions.

In this paper, we propose a variable step fluid model for faster simulation of communication networks. The variable step fluid model provides accurate estimation of interest measures such as size of TCP congestion window, queue sizes, the round-trip time, and the throughput. Also, our fluid model reduces the ripple effect significantly by allowing variable step size integration.

A method to reduce ripple effects with fixed step size integration has been introduced in [8], but our approach improves accuracy and performance further by varying the step size. We conduct simulation experiments on TCP congestion control to validate our model. `ns-2` packet-level simulator is chosen to validate our model. In these experiments we observe our model achieves significant speedup while maintaining little error against `ns-2`. For example, the speedup of our fluid model for 20 Mb bottleneck is 40 to 70 against the `ns-2`.

The rest of the paper is organized as follows. In Sect. 2, related work is presented. In Sect. 3, we introduce the variable step fluid model algorithm. We show experimental results in Sect. 4, and conclude in Sect. 5.

2 Related works

There has been much research on the network simulation based on fluid models [3, 4, 9, 8]. Liu et al. [3] compare packet-level simulations and fluid models in terms of relative efficiency. They have observed ripple effects in detail and shown that fluid models perform worse than packet models when ripple effects start to occur. However, the accuracy of fluid models has not been fully analyzed.

[4] also studies trade off between packet-level and fluid models. The change of event rates in fluid and packet-level simulations with respect to the number of nodes or the number of flows are compared. However, they do not consider the accuracy of fluid models against packet models.

A scalable model of a network of AQM routers is presented in [9], and the transient behavior of the average queue length, packet loss probabilities, and average end-to-end latencies have been observed. It is shown that their fluid model is accurate and requires substantially less time to solve, especially when workloads and bandwidths are high. It is also shown that the computational complexity grows linearly with the size of the network, whereas the growth of the complexity for discrete event simulators is super-linear. However, this model is based on the expected values of variables and the accuracy of results only applies when there exist large number of flows. A fluid model introduced in [9] also captures average window and queue sizes, but these statistics are not very accurate compared to the statistics of packet-level simulations.

[8] introduces a time-driven fluid simulation model for high speed networks. Time is partitioned into *fixed-length* intervals. [8] observes only single class fluid and it does not simulate background flows. The propagation delays between any two nodes are assumed to be zero or multiples of the discretization interval length in [8], while the delay in reality depends on the traffic conditions. [8] concerns only how much traffic is generated by each source, not the exact event arrival time. In addition, computations are performed on a very simplified scenario and backlogged fluid depends on arrived fluid only.

3 Algorithm of variable step size fluid model

3.1 Motivation

Although fixed constant time step [8] may reduce the ripple effect, simulation time and accuracy can be improved further if we use variable time step fluid model. For example, suppose that the round-trip time is τ ms and the congestion window size of TCP is α. Then α packets are emitted for τ ms. Assume that TCP sends α packets within $\tau/2$ ms and no packet is sent out for the remaining $\tau/2$ ms as in Fig. 1. If the fixed time step is smaller than the round-trip time, for example, $\tau/2$ ms, the TCP flow rate is $2\alpha/\tau$ during the first half of RTT and 0 during the second half of RTT. This introduces unnecessary rate changes. On the other hand, if constant time step is much bigger than the round-trip time, for example, multiples of τ ms, it may not accurately capture interest measures such as congestion window size of TCP because TCP varies congestion window size every RTT (τ ms).

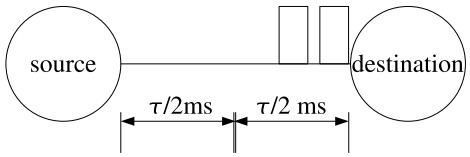

Fig. 1. Timestep

The variable step fluid model discretizes time using the round-trip time (RTT) as an interval. Since the RTT varies with the traffic condition, the moments when network variables change their values cannot be captured by a fixed time-step model if fixed-step is bigger than RTT. For example, when the network is congested (i.e. RTT is large), integration step can be large in the variable step size model. On the other hand, when the network is not congested (i.e. RTT is short), integration step need to be short to accurately capture TCP window size. The discretization in variable step fluid model also enables us to reduce the ripple effect. Ripple effects occur if flow rate is computed whenever the rate

changes. For example, flows in [3] trigger an event, even though the input amount is smaller than the size of a single packet. Since our fluid model computes the flux and other variables every round-trip time, an event explosion such as ripple effect due to frequent rate change does not occur.

3.2 Algorithm

This section describes the algorithm of our variable step size fluid model. We begin with a simple case with a router, one TCP and one UDP flow. The algorithm can be easily extended to multiple routers with many TCP and UDP flows. Our fluid model algorithm is summarized in Fig. 2.

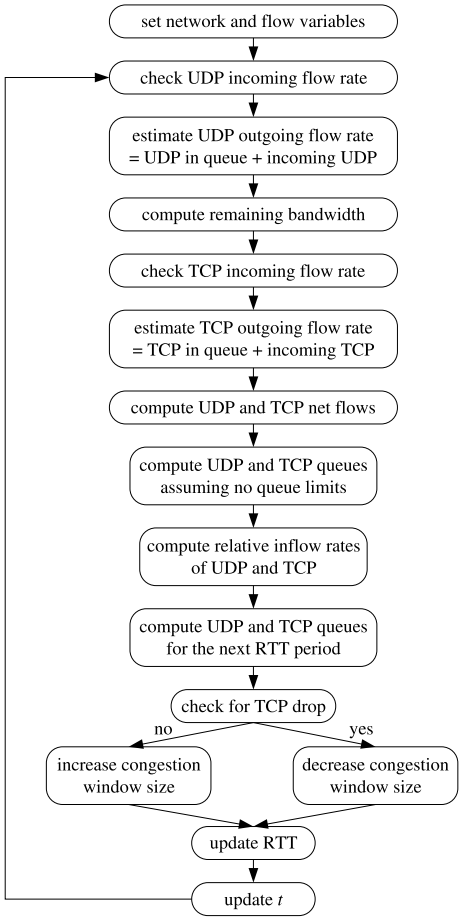

Fig. 2. Algorithm

The algorithm first defines the network topology, background and foreground flows. For example, let $b(t)$ denote the available bandwidth at time t. Then $b(t)$ is initialized to (bandwidth $\times$ RTT). Then the algorithm replaces packets during RTT as a continuous flow. Our fluid model computes UDP traffic and allocates network resources for UDP first because they do not reduce the sending rates even when packets are lost in the middle of transfer. We vary portion of background UDP flows throughput the simulation. Thus, the first step is to compute followings for UDP flows. Let $u_U(t)$ and $v_U(t)$ denote the amount of inflow and outflow of UDP flow at t, respectively. Then

$$\begin{aligned} u_U(t) &= (\text{inflow rate}) \times \text{RTT} \\ v_U(t) &= \min\{q_U(t) + u_U(t), b(t)\} \\ b(t) &= b(t) - v_U(t) \end{aligned}$$

where $q_U(t)$ is the amount of UDP flow in the queue at t. Then the remaining resources will be shared by TCP flows. Let $u_T(t)$ and $v_T(t)$ denote the amount of inflow and outflow of TCP flow at t, respectively. Then,

$$\begin{aligned} u_T(t) &= cwnd(t) \times (\text{packet size}) \\ v_T(t) &= \min\{q_T(t) + u_T(t), b(t)\} \\ b(t) &= b(t) - v_T(t) \end{aligned}$$

where $cwnd(t)$ is the congestion window size for TCP at t and $q_T(t)$ is the amount of TCP flow in the queue at t. The next step computes *net flows* between inflows and outflows for UDP and TCP flows:

$$n_i(t) = u_i(t) - v_i(t), \quad i = U, T$$

Let q_U^∞ and q_T^∞ be queue sizes when it is *temporarily* assumed there are no queue limits. Then,

$$q_i^\infty(t) = q_i(t) + n_i(t), \quad i = U, T$$

Let $q^\infty(t) = q_U^\infty(t) + q_T^\infty(t)$. Let us denote relative inflow rates of UDP and TCP by $r_i(t) = u_i(t)/u(t), \quad i = U, T$, where $u(t) = u_U(t) + u_T(t)$. When there are not sufficient resources for TCP, TCP flows will compete for limited resources and reduce sending rates accordingly if flow loss occurs. Flow loss (for UDP or TCP) occurs when the queue exceeds its maximum size, denoted by $q_{\max}$. The increase or decrease of the queue depends on net flows and there are four cases:

Case I. ($n_U > 0$ and $n_T > 0$)

If $q^\infty(t) > q_{\max}$, define

$$q_i^0 = q_i(t) + (q_{\max} - q(t))r_i(t), \quad i = U, T$$

where $q(t)$ is the queue size at t. Otherwise, define

$$q_i^0(t) = q_i^\infty(t), i = U, T$$

q_U^0 and q_T^0 are queue lengths for UDP and TCP flows at (t + *new* RTT). Since *new* RTT is not known yet, q_U^0 and q_T^0 are used temporarily now and stored to $q_U(t+new\ \text{RTT})$ and $q_T(t+new\ \text{RTT})$ when *new* RTT is obtained.

Case II. ($n_U < 0$ and $n_T < 0$)
Define

$$q_i^0(t) = \max\{0, q_i(t) + n_i(t)\}, \quad i = U, T$$

Case III. ($n_U > 0$ and $n_T < 0$)
Define $q_T^0(t) = \max\{0, q_T^\infty(t)\}$. If $q_U^\infty(t) + q_T^0(t) > q_{\max}$, define

$$q_U^0(t) = q_{\max} - q_T^0(t)$$

Otherwise, define

$$q_U^0(t) = q_U^\infty(t)$$

Case IV. ($n_U < 0$ and $n_T > 0$)
Define $q_U^0(t) = \max\{0, q_U^\infty(t)\}$. If $q_T^\infty(t) + q_U^0(t) > q_{\max}$, define

$$q_T^0(t) = q_{\max} - q_U^0(t)$$

Otherwise, define

$$q_T^0(t) = q_T^\infty(t)$$

Then, RTT is updated with the sum of the transmission delay, the queueing delay and the propagation delay. If TCP flow loss occurs in Case I or IV, the inflow rate is reduced by halving $cwnd(t)$ and threshold, $h(t)$. When TCP flow loss does not occur, $cwnd(t)$ and $h(t)$ increase exponentially up to the receiver window in the slow-start phase and linearly after that. If the window size reaches the receiver window, the slow-start phase changes to the congestion avoidance phase. After setting $q_U(t+\text{RTT}) = q_U^0(t)$, $q_T(t+\text{RTT}) = q_T^0(t)$ and $q(t+\text{RTT}) = q_U(t+\text{RTT}) + q_T(t+\text{RTT})$, t can be updated to $t+\text{RTT}$. Note also that there is no trade-off of adjusting the step size every RTT because RTT itself is the step size.

4 Simulation Experiments

In order to validate our model, we use the parking-lot topology in Fig. 3. This topology has multiple bottleneck links, one between nodes R2 and R4, and another between nodes R6 and R8. Foreground flows are sent from R1 to R10.

Then, the first set of background traffic is sent from R3 to R5. The second set of background traffic is sent from R7 to R9. A similar type of topology was also considered in [9].

We consider several network scenarios by assigning different bandwidths from 5 Mb to 20 Mb to these bottlenecks. Drop Tail Queues (FIFO) are used for each router. Different types of queues such as AQM can be easily considered and are postponed for future work. We simulated extensive scenarios on this network topology, but we will show two example scenarios in this paper. Mainly, we simulate three types of flows on this topology, which are foreground TCP, foreground UDP and background UDP flows. TCP flows go from node R1 to node R10 via nodes R2, R4, R6 and R8. Foreground UDP flows use the same path as TCP flows. Background UDP flows propagate from node R3 to node R5 passing through nodes R2 and R4, and from node R7 to node R9 passing through nodes R6 and R8. Background UDP flows represent background traffic on the network which is composed of short lived TCP flows, whereas foreground UDP assumes UDP packets are generated by protocols such as RTP[10].

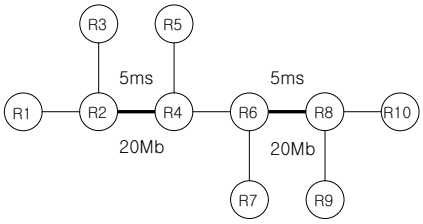

Fig. 3. Parking Lot Topology

4.1 TCP flows without background traffic

We first consider a flow generated by a single TCP connection without any background traffic. TCP starts at 0.5 second and is observed for 20 seconds. Once a TCP connection is established, a source sends a file of size 50 MB to the destination. We run the same scenario on `ns-2` to compare results between our model against packet-level simulations. For packet simulations in `ns-2`, the packet size is set to 1500 bytes, and the queue may hold up to 50 packets. The initial threshold of the congestion window is set to infinite. Fig. 4 shows that RTT, congestion window size, queue size, and throughput of our variable step size model captures those statistics of `ns-2` with little error when the bottleneck bandwidth is 20 Mb. As RTT increases, the congestion window size increases exponentially and the queue becomes filled up with the surplus from the inflow. When the queue becomes full, flow loss occurs and this leads to the halving of

the congestion window size. Correct estimation of RTT change in the variable step fluid model allows us to detect the change of the congestion window size and the queue precisely as shown in Fig. 4. The throughput of our model also captures the result of `ns-2`.

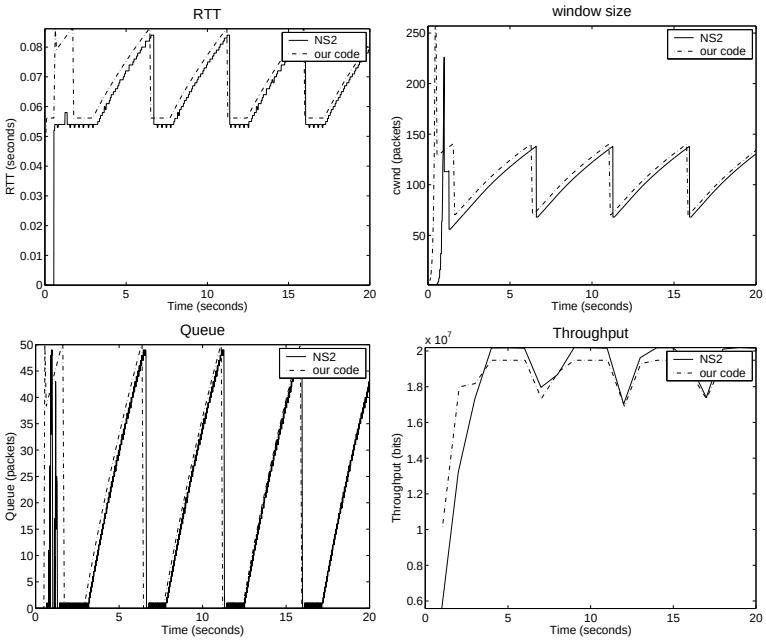

Fig. 4. Round-trip time (top left), congestion window size (top right), Queue size (bottom left) and throughput (bottom right) from 1 TCP flow when bottleneck bandwidth is 20 Mb

4.2 Foreground TCP and UDP flows with background UDP flows

Now suppose that there are 3 foreground TCP, 20 foreground UDP and 50 background UDP flows. The simulation is performed for 50 seconds, and three TCP connections share the same bottleneck. Each TCP connection is initiated at 0.5 second and transfers a file of size 50 MB. In this scenario 10 foreground UDP flows are sent from R3 to R5 and R7 to R9 at 10 seconds. Each UDP flow sends out 240 Kb of data per second. At 20 seconds in simulation time, 10 more foreground UDP flows are sent from R3 to R5 and R7 to R9 with 240Kb each. In 30 seconds, the entire foreground UDP flows are disconnected. Since each UDP source sends 12Kb of packet for every 0.05 second, UDP flows occupy about 2.4 Mb of the bottleneck bandwidth for 10 seconds, and 4.8 Mb

of the bottleneck bandwidth for the next 10 seconds. In addition, background UDP traffic is injected for the last 10 seconds. The sending rate of background UDP traffic is 48 Kb and its on and off time is 500 ms, respectively. Thus each background On-Off source generates 24 Kb per second on average.

A traditional fluid model may consider fluid chunks using packets averaged out during on-time. Our fluid model introduces larger scale to define a single UDP flow. Since a packet is of size 12 Kb, the number of packets sent by each UDP flow follows the exponential distribution with mean 2. Thus 50 background UDP flows generate about 1.2 Mb per second on average in `ns-2`. Similarly, the background UDP flow in our fluid model sends out a flow at a constant rate of 1.2 Mb every second from its source to the destination. TCP flows merge with one UDP flow between nodes R2 and R4 and merge with another UDP flow from node R6 through node R8 to node R9.

Fig. 5 shows the results from `ns-2` and the fluid model when bottleneck bandwidth is 20 Mb. Round-trip time, congestion window size, and throughput are all captured with little error. The adjustment of the congestion window size due to dynamic injection of UDP flows are matched closely between `ns-2` and the fluid model. Fig. 5 also shows that our model captures RTT and throughput computed from `ns-2`.

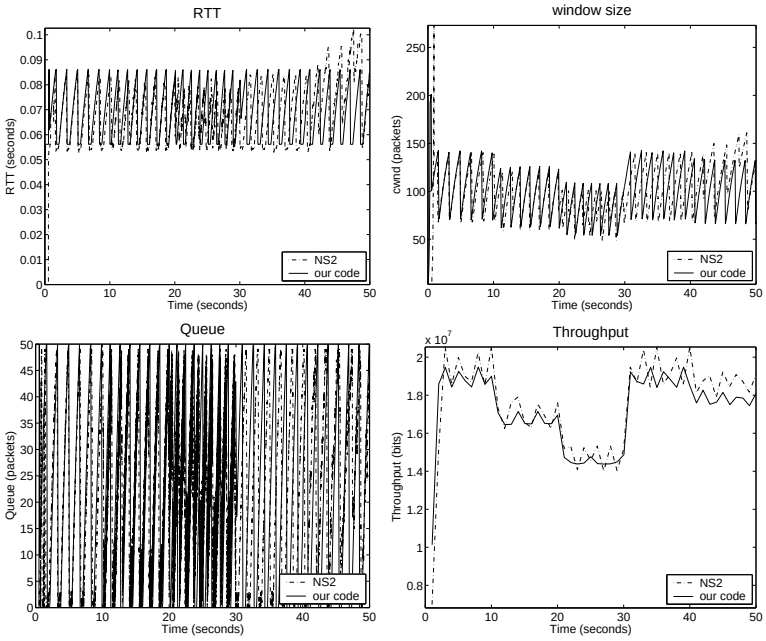

Fig. 5. RTT (top left), congestion window size (top right), Queue size (bottom left) and throughput (bottom right) from 3 TCP, 20 UDP and 50 background UDP flows when bottleneck bandwidth is 20 Mb

Now we consider the speedup of variable step fluid model compared to `ns-2` packet-level simulation. Left of Fig. 6 compares computation times between `ns-2` and our fluid model simulations, the case study in Sect. 4.1, when the bandwidth of bottlenecks changes. Note that the CPU time is plotted in log-scale. This shows that our fluid model reduces the computational cost significantly. In this comparison, the fluid model takes about 0.2 second to run. `ns-2`, on the other hand, takes about 2.8 up to 8.4 second depending on the bandwidth. When the bottleneck bandwidth is 20 Mb, Running time in `ns-2` is 42 times more than that of the fluid model. As the bandwidth increases, more packets are generated in the packet level simulator, thus `ns-2` would require more computation time. Right of Fig. 6 compares the CPU time in the second scenario in Sect. 4.2, where 3 TCP, 20 UDP and 50 background UDP flows exist. In this scenario, the CPU time for the fluid model is about 0.4 second, whereas `ns-2` takes 13.3 to 28.2 seconds depending on the bandwidths of the bottleneck link. When the bottleneck bandwidth is 20 Mb, the speed up of fluid model over `ns-2` is around 70. Thus, our variable step fluid model achieves significant speedup if we compare CPU time against that of packet-level simulations.

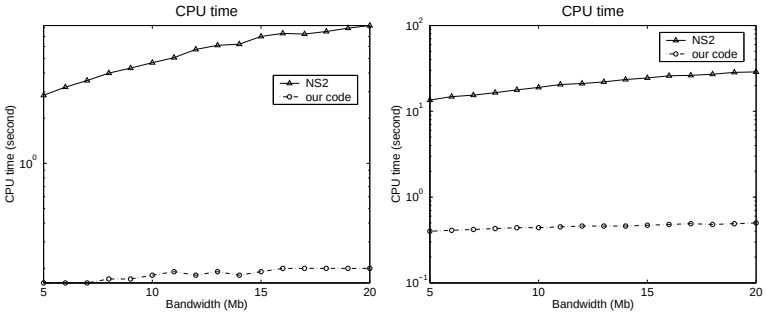

Fig. 6. CPU time for 1 TCP flow(left), CPU time for 3 TCP, 20 UDP and 50 background UDP flows(right)

5 Conclusion

In this paper, a variable step fluid model has been introduced to simulate network traffic in the communication network. Our model replaces discrete packet-level events with a propagation of continuous fluid flow. The variable step fluid model estimates the variation of queues accurately and captures the round-trip time, the congestion window size, and the throughput of TCP connections. With the compensation of negligible error, the variable step fluid model reduces the computational load. To validate our fluid model against `ns-2`, two network traffic scenarios are considered. When single TCP flow is simulated, we showed that

our fluid model saves up to 97.5% of CPU time compared to `ns-2` packet-level simulator. If there are 3 TCP flows, 20 foreground UDP flows, and 50 background UDP flows, we save up to 99% of CPU time. We consider general TCP and UDP model in this paper. Modeling of different TCP implementations such as Reno, NewReno, SACK, etc or modeling of flows other than TCP or UDP such as TERC flows can be implemented into the current fluid model as a traffic source module. The current fluid model assumes identical pair of a source and a destination for each TCP flow. When there are multiple pairs of TCP flows, the computation time increases linearly and it is reserved for future research.

References

1. The VINT Project, a collaboratoin between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC: The ns Manual (formerly ns Notes and Documentation). (2000) Available at *http://www.isi.edu/nsnam/ns/ns-documentation.html*.
2. Ahn, J.S., Danzig, P.B.: Packet network simulation: speedup and accuracy versus timing granularity. IEEE/ACM Trans. on Networking **4**(5) (1996) 743–757
3. Liu, B., Figueiredo, D.R., Yang Guo, J.K., Towsley, D.: A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation. In: Proc. of the IEEE INFOCOM. Volume 3. (2001) 1244–1253
4. Liu, B., Guo, Y., Kurose, J., Towsley, D., Gong, W.: Fluid simulation of large scale networks: Issues and tradeoffs. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. Volume IV. (1999) 2136–2142
5. Guo, Y., Gong, W., Towsley, D.: Time-stepped hybrid simulation (TSHS) for large scale networks. In: Proc. of the IEEE INFOCOM. (2000)
6. Kumaran, K., Mitra, D.: Performance and fluid simulations of a novel shared buffer management system. In: Proc. of the IEEE INFOCOM. (1998) 1449–1461
7. Liu, B., Figueiredo, D.R., Yang Guo, J.K., Towsley, D.: A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation. In: Proc. of the IEEE INFOCOM. Volume 3. (2001) 1244–1253
8. Yan, A., Gong, W.B.: Time-driven fluid simulation for high-speed networks. IEEE Transactions on Information Theory **45**(5) (1999) 1588–1599
9. Liu, Y., Presti, F.L., Misra, V., Towsley, D., gu, Y.: Fluid models and solutions for large-scale ip networks. In: ACM SIGMETRICS. (2003)
10. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard) (2003)