FluNet: A Hybrid Internet Simulator for Fast Queue Regimes

Yung Yi and Sanjay Shakkottai*

July 11, 2007

Abstract

Motivated by the scale and complexity of simulating large-scale networks, recent research has focused on hybrid fluid/packet simulators, where fluid models are combined with packet models in order to reduce simulation complexity as well as to track dynamics of end-sources accurately. However, these simulators still need to track the queuing dynamics of network routers, which generate considerable simulation complexity in a large-scale network model. In this paper, we propose a new hybrid simulator – FluNet – where queueing dynamics are not tracked. The FluNet simulator is predicated on a fast-queueing regime at bottleneck routers, where the queue length fluctuates on a faster time-scale than end systems. FluNet does not track queue lengths at routers, but instead, replaces queue-based AQM schemes (such as RED or DropTail) by an equivalent rate-based model. This allows us to simulate large-scale systems, where the simulation "time step-size" is governed only by the time-scale of the end-systems, and not by that of the intermediate routers; whereas a fluid model based simulator that *tracks* queue-length would require decreasingly smaller step-sizes as the system scale size (such as the number of flows and link capacity) increases. We validate our model using a *ns-2* based implementation. Our results indicate a good match between packet systems and the associated FluNet system.

1 Introduction

1.1 Motivation

The Internet has experienced tremendous growth in both scale and speed, and the control and management of the Internet is becoming an ever more important issue. To model and understand the behavior of such networks, several widely-used discrete event-driven simulators are available (such as ns-2, GlomoSim, QualNet, PDNS, and SSFNet, see their references in [1]) in the area of simulation. However, event-driven simulation of large scale network systems with a significant number of users and flows is difficult due to simulation time complexity.

Recently, there have been significant efforts on developing fluid model based simulators to address the time complexity of discrete event simulators. These simulators can be classified into *pure fluid model based approach*, and *hybrid model based approach.* Pure fluid model based approaches include [3–6], where the authors are primarily interested in rate based fluid modeling of TCP sources, AQM algorithms, and their interactions. On the other hand, hybrid model based approaches [2, 7–10] integrate packet models along with fluid models to enable hybrid

*Y. Yi is with the Department of Electrical Engineering, Princeton University (e-mail: yyi@princeton.edu). S. Shakkottai is with the Department of Electrical and Computer Engineering, The University of Texas at Austin (email: shakkott@ece.utexas.edu). This research was supported by NSF Grants ACI-0305644, CNS-0325788 and CNS-0347400.

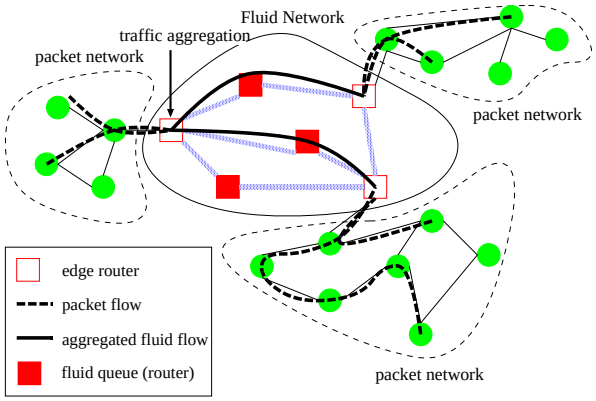


Figure 1: Hybrid Simulation Framework [2]

simulation. Hybrid simulators have both advantages of (a) accurately tracking source dynamics (as the sources in the simulator are typically modeled using packet networks), and (b) reducing simulation time by decreasing the number of generated events. This reduction is due to fluid approximation in the core network (the large system scale size permits a fluid approximation to be accurate [11]) and "dimensional collapse" due to traffic aggregations inside the core network (see Figure 1). There exist two main components that have to be modeled by fluids: *sources* and *router queues* (i.e., AQM algorithms), which are modeled by coupled differential equations (e.g., see [2, 3, 5]). In hybrid simulation, the main focus typically lies in the fluid modeling of the router queues, unless there exist sources inside the core network modeled by a fluid model. We address the issue on fluid modeling of router queues in this paper.

The model used in existing hybrid simulators, such as in [2], integrate fluid models with packet systems by measuring data rate from packet flows over short time-intervals (i.e., time-step-size) and use these rate measurements to drive a fluid simulator. The resulting fluid simulator consists of a collection of fluid queues, which evolve in discrete time. The dynamics of these fluid queues are used to "mimic" the packet system, e.g., fluid queue lengths are used to determine packet dropping and marking probabilities in the intermediate routers. From the perspective of fluid modeling of router queues, this time-step-size should be small enough to accurately track the fluid queue length dynamics, i.e., queue length variation should not be significant at successive sampling instants. We call this approach "queue-tracking fluid approach."

A potential problem with this approach is that the rate of queue length variations increases as the system scale size (i.e., the number of flows and the link capacity) increases. Note that the time-step-size is inversely proportional to "simulation event rate" that is linearly proportional to simulation time complexity [12]. Thus, increasingly large simulation event rates are required as the system scale size increases. To understand this intuitively, consider a queue of capacity nc accessed by n flows, where the number of packets generated by each flow is modeled by a Poisson process with parameter λ, and the size of each packet is fixed. Then, from standard queueing theory, the (mean) busy period (i.e., the time interval over which an empty router buffer fills up and empties again) is expressed by $\frac{1}{n(c-\lambda)}$. Note that this time step is *inversely proportional* to n, the system scale size. This implies that we need progressively smaller step-sizes (increasingly higher event rate) to capture the fluid queue dynamics accurately as the system scale (i.e., n) increases (see Figure 2 for illustration). Thus, such a queue-tracking fluid approach is not scalable with respect to the system scale size.

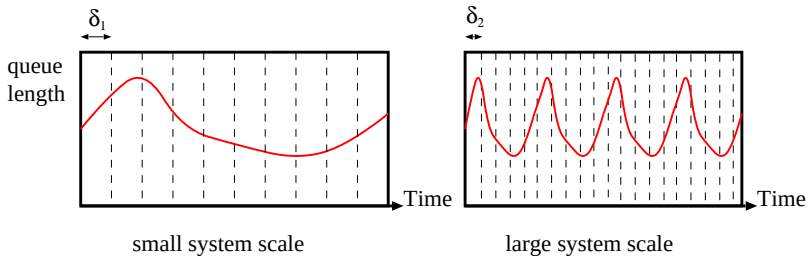


Figure 2: The queue length is sampled over a step size δ_1 in the left figure, but the step size needs to decrease to δ_2 in the right figure as the number of flows and the capacity of the router increases, in order to accurately capture the queueing dynamics.

1.2 Overview and Related Work

To address the scalability problem of queue-tracking fluid approach, we develop a new fluid simulator, **FluNet**, where the step-size is *independent* of the system scale size, but only depends on the time-scale of end-system source dynamics. The FluNet is predicated on a "fast queue regime" at bottleneck routers, and does not track queue lengths at the core-routers. Instead, it uses an *equivalent rate based marking/dropping model* for a given queue based AQM model.

The main assumptions of the fluid model in FluNet are *(i)* sufficient randomness in the system and *(ii)* large system scale size. The randomness in the Internet is generated by e.g., unresponsive flows and flow initiation/terminations. Indeed, recent studies [13, 14] show that unresponsive sources contribute to about 70% - 80% of the Internet flow counts[1]. Typical examples of such unresponsive flows include multimedia (video and audio) flows and web mice (short-lived HTTP flows). Further, the scale of the current Internet is very large, e.g., in the Sprint backbone, OC-192 links have been installed in several POPs (Points-of-Presence) with a total of approximately 11 M TCP connections being captured during one hour in those back-bone routers [15].

Under such a regime, we will have a considerable number of "cycles" in the queue dynamics of the intermediate routers even over a small interval of time, where one "cycle" corresponds to the time interval over which an empty router buffer fills up and empties again (technically, the regeneration time). In particular, the queue dynamics at the intermediate routers occur on a much *faster* time-scale than that of the end system sources[2] [16, 17]. Thus, it is reasonable to expect that queueing dynamics are not visible to the end system controller. Instead, the queueing behavior at the router affects the end system controller only through the "statistical behavior of the queue."

This allows us to simulate large-scale systems, where the simulation time-step size is governed only by the time-scale of the end-systems, and not the time-scale of queue dynamics at the intermediate routers. Note that the time-scale of the end-systems is governed by the window dynamics of the congestion controller (e.g., TCP), which is, in turn, determined by the round-trip time. This does not change with respect to the system scale size[3] resulting

[1]However, the volume of data in unresponsive flows contribute to about 10% - 20% of the total traffic volume of the Internet.

[2]In order to understand this intuitively, consider a router of capacity $n \times c$ accessed by n TCP flows and n unresponsive flows. Then, the time scale of a TCP source rate update is the order of $1/c$ (since its rate update is clocked by the ACK packets from the receiver), whereas the time scale of a router queue "cycle" is in the order of $1/(nc)$.

[3]The instantaneous round-trip time may vary w.r.t the system scale size, but the long-term mean of round-trip time does not.

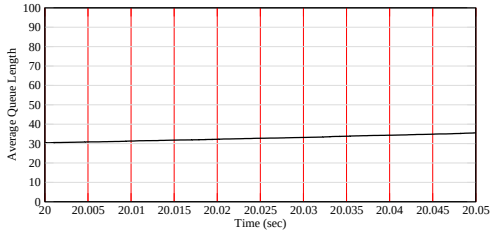
in a fixed step-size that does not scale with n.

In [17–19], the authors have formalized the above heuristic, and have derived an equivalent rate based marking model in the large n regime, which is the basis of our work in this paper. However, it is questionable whether we can apply this limiting regime toward a practical (finite-scale) simulation study. Further, from a theoretical point of view, we need new results for AQM algorithms employing a queue averaging technique (for absorbing the burstiness of incoming traffic). From an implementation point of view, we also need to consider practical issues such as the choice of time-step-size, handling dropping functionality (e.g., forced drop mode in RED, and DropTail queue), the design of the simulation framework, and importantly, extensive simulation results for validation of this approach.

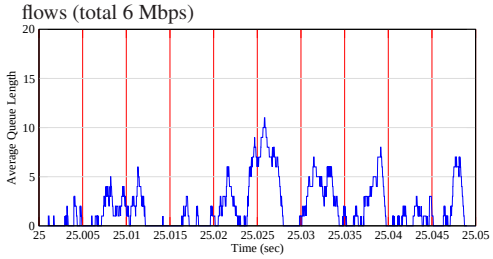
Practically, in a fluid model simulator, a queue-tracking fluid approach and our scheme should be used together, depending on the queue regime that can be seen for a given measurement interval. For a fixed measurement interval, the fluid model in FluNet works better under a fast queue regime, i.e., there are a large number of regenerative cycles, such that stationary queue length behavior is seen, but the incoming rate of TCP flows looks constant over the interval. However, under a slow queue regime, where the queue lengths do not vary much, queue-tracking fluid approach performs better, since discrete sampling in this case tracks the queue length very well. The fast queue regime becomes a slow queue regime if we decrease the measurement interval, but at the cost of increasing simulation complexity. Thus, a fast or slow queue regime is relatively determined by the chosen measurement interval, queue length threshold, and the system scale size (see Section 2 and Section 5 for more details).

We comment that another interesting approach is presented in [20], where the authors describe a procedure where a sampled version of the traffic is fed to scaled-down version of the network model, and then linearly extrapolate the results from the scaled-down system to the original large-scale system. Essentially, the authors argue that in several ways, a slowed down system mimics the larger system. However, the authors in [20] point out that correlated events (e.g., burst of packet losses, for instance) breaks the linear-scaling hypothesis, and causes the scaled simulation to deviate from the real system. Recent work in [21] applies network calculus based on the mathematical theory of Min-Plus (or Max-Plus) algebra to fluid modeling of network dynamics. We also remark that results in [21] suggest that fluid queue based simulation could perform poorly when the bottleneck buffers are not saturated. This can be understood from the fact that unsaturated buffers correspond to a system with fast queueing dynamics, where tracking queue length trajectories (i.e, a fluid queue based approach) may not be feasible. More related work on the alternate models for fast queue regimes are discussed in Section 3.3.

The work in [12, 22, 23] studied the *ripple-effect*, which increases the computational complexity in fluid simulation. The ripple-effect describes the phenomenon where changes in the data rate in one flow induces rate changes in flows which also traverse any of the links along the original flow's path. These rate changes propagate through the network, and lead to increased rate fluctuations in the network. This effect necessitates progressively smaller simulation step sizes for increasingly larger and more complex network topologies and flow configurations. We note that our approach is relatively insensitive to the ripple effect as we do not track instantaneous flow variations. It would be of interest in the future to study the ripple effect in the context of combining a queue-tracking approach and our approach.

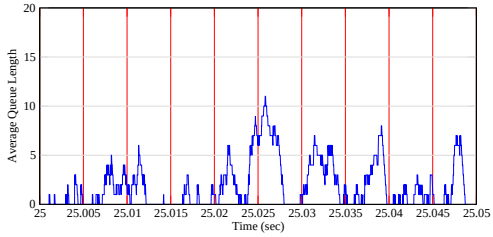


(a) 20 Mbps bottleneck bw, 20 TCP, and 20 unresponsive flows (total 6 Mbps)

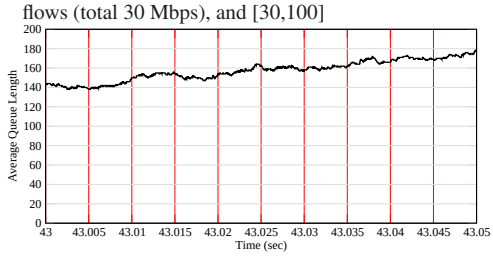


(b) 100 Mbps bottleneck bw, 100 TCP and 100 unresponsive flows (total 30 Mbps)

Figure 3: Rate of queue variations for different system scale sizes



(a) 100 Mbps bottleneck bw, 100 TCP, 100 unresponsive flows (total 30 Mbps), and [30,100]



(b) 100 Mbps bottleneck bw, 100 TCP, 100 unresponsive flows (total 30 Mbps), and [300,1000]

Figure 4: Rate of queue variations for different queue threshold parameters: [a,b] = [`min_th` , `max_th`] of RED

2 When FluNet Works: Fast and Slow Queue Regime

In this section, we present the *ns-2* based simulation results to show that the rate of queue variations indeed become faster for both larger system scales and smaller queue operating size of AQM (i.e., `min_thresh` and `max_thresh` of RED, and a physical queue size DropTail). Recall that FluNet is designed to operate at a fast queue regime, and thus these simulation results provide a practical motivation for the FluNet. The simulations are performed in a single bottleneck network (see Figure 7(a)) with RED algorithm [24], accessed by TCP and unresponsive ON-OFF sources. The end-to-end source-destination round trip time is set to be 200 msec, which is a typical end-to-end delay of trans-continental flows. In the simulation results, shown in Figures 3 and 4, we plot the queue length trajectory at the bottleneck link over a time interval of 50 msec, which can essentially be regarded as the measurement interval of a fluid simulator.

First, in Figure 3, we plot the queue length trajectories for different system scale sizes, where the system size in Figure 3(b) is scaled by a factor of 5. We observe that increasing system scale size generates faster queue dynamics, as previously discussed in Section 1.

Second, we note that fast queue variations can also occur if the mean queue length is kept small, and thus resulting in many "busy cycles" over a short interval of time. For example, with the RED algorithm, the parameters, `min_th` and `max_th`, determine the queue length thresholds at which packet marking/dropping occurs (see Section 3.2.2 for details). This in turn leads to fast queue length variations as illustrated in Figure 4(a). On th other hand, setting these parameters to be large results in slow queue length variations (see Figure 4(b)), where the queue-tracking fluid approach will perform well, for a given sampling measurement interval.

Experimental motivation and justification for fast queue regimes is also based on [25, 26], where a fast queue

regime corresponds to a small queue regime in a large-scale system. Note that the term 'small queue' corresponds to the queue length when normalized with the system capacity. Such a regime seems reasonable for large-scale systems based on arguments presented in [25, 26]. It has been argued that the required buffer size does not have to scale linearly with the system scale size [25], and in fact can be fixed, independent of the system scale size [26]. This implies that in large systems, the buffer fluctuations will be fast, because the buffer size normalized to the link capacity shrinks. Thus, the queue length *normalized* with the capacity will be small, leading to fast queue dynamics. More discussions on buffer size scaling will be provided in Section 3.3.

3 Fluid Queueing Model of FluNet

3.1 Equivalent Rate Based Model with Queue Averaging

In this subsection, we derive the fluid queueing model of FluNet in the system with a large number of TCP and unresponsive flows. The analysis here adopts similar notations and techniques used in [18], but includes an important extension for AQM algorithm with *queue averaging* which is used in various AQM algorithms (e.g., RED [24] and REM [27]) to absorb a certain level of burstiness of incoming traffic.

What we intend to study through our analysis is the queue length process over a (appropriately-scaled) short-time interval for the large system scale size, n. Even over this small time interval, we will show that the queue reaches "steady-state" behavior. This occurs due to the fact that the capacity is very large (nc), and causes the queue to "regenerate" an arbitrarily large number of times over this short interval. However, from a viewpoint of a single *source*, this corresponds to a very short interval of time, where one can expect that the end-user will only perceive the statistical "steady-state" queueing behavior. By quantifying the above heuristic argument, we develop an equivalent rate based model for a given queue based marking function.

We consider the systems with scaling parameter n. The model of the n-th system consists of a single bottleneck router (with its capacity nc) fed by n TCP flows and n unresponsive flows. For notational simplicity, we have assumed an equal number of TCP and unresponsive sources[4]. We denote the fluid rates of individual TCP flows in the n-th system by $\{x_n^i(t), i = 1, \ldots, n\}$, where $x_n^i(t)$ denotes the instantaneous arrival rate of a TCP flow i at time t. Let $x_n(t)$ be the average TCP arrival rate (over flows) at time t, i.e., $x_n(t) = \frac{1}{n}\sum_{i=1}^{n} x_n^i(t)$. For the well-defined initial conditions, we assume that $x_n^i(0) \to x^i(0)$, and $x_n(0) \to x(0)$, as $n \to \infty$. We model each unresponsive flow by means of a simple point process. We refer the readers to the Appendix for the technical details of assumptions and the complete proof of our main result, Theorem 3.1.

Associated with the router is a queue based marking function (AQM algorithm), denoted by $p_q(\bar{Q}_n(t))$, where $\bar{Q}_n(t)$ is the exponentially-moving averaged queue length. Note that our marking function is based on the actual queue lengths, not the scaled queue length. Let $Q_n(t)$ be the *instantaneous* queue-length process at the router. Then, the weight-averaged process is given by $\bar{Q}_n(t) = w_n\bar{Q}_n(t - \delta_n) + (1 - w_n)Q_n(t)$, where $0 < w_n < 1$ is the queue-averaging parameter for n-th system and $\delta_n = 1/(nc)$ corresponds to the time-scale of the queue variation at the router (see also [3]).

In [24], the authors provide a guideline on the choice of the parameter w_n. Essentially, the authors in [24] argue

[4]The results in this paper hold even if they are not the same, as long as the ratio of the number of TCP flows and the number of unresponsive flows is finite.

that w_n is chosen such that a fixed burst of packets (i.e., L back-to-back packets from a single flow) should be allowed into the router without this burst being marked. This burst tolerance is chosen to account for TCP window behavior and cumulative ACKs, which lead to a burst of packets being transmitted from a single TCP source, instead of the packets being spaced apart. However, observe that as the number of flows and capacity increases, the *normalized packet burst size* decreases (normalization with respect to link capacity).

In particular, consider a bottle-neck router with capacity nc, and fed by n independent arrivals each of which has a packet-burst of size L packets (i.e., L back-to-back packets from a single flow). Then, if the flows are independent, it is unlikely that the packet bursts from various flows will synchronize and form a single large burst of nL. This heuristic is supported by [28], where the authors show that when multiple flows are aggregated, and the individual flows have different burstiness but equal rates, the burstiness of the aggregate flow is determined by the burstiness of the individual constituent flow which has the maximum burstiness. In other words, as the number of flows and the bottle-neck link capacity increases, the burstiness of aggregate incoming flows remains constant. This implies that the queue averaging parameter w_n needs to become smaller as the system scale increases (because the normalized packet burst size decreases). Motivated by this argument, we make the following assumption:

Assumption 3.1. $w_n \overset{n\to\infty}{\longrightarrow} 0$.

We define a queue length process $q(t)$ to be the queue length process of M/D/1 queue with Poisson arrival rate λ and capacity $c - x(0)$. Then, we have the following main result on the fluid queueing model:

Theorem 3.1. *For a fixed* $T < \infty$,

$$\frac{1}{T/n}\int_0^{\frac{T}{n}} x_n^i(y)p_q(\bar{Q}_n(y))\,dy \overset{n\to\infty}{\longrightarrow} x^i(0)\frac{1}{T}\int_0^T p_q(q(y))\,dy \tag{1}$$

Theorem 3.1 states that in the large n regime, the time-average volume of marks experienced by i-th TCP flow over the interval $[0, T/n]$ (LHS) can be well approximated by the marked volume at the M/D/1 queue with Poisson arrival rate λ and capacity $c - x(0)$ (RHS). *Note that the original unresponsive arrival process is not necessarily a Poisson process.* In other words, for n large enough, and fixed T, the interaction between the router queueing process and the congestion controller at a fixed user occurs only through $\frac{1}{T}\int_0^T p_q(q(y))\,dy$ (the marking probability in the limiting system).

Thus, the system with aggregate unresponsive rate of $n\lambda$, and with aggregate TCP rate of nx can be represented by M/D/1 system of fixed service rate of $c - x$, and an arrival process that is Poisson with parameter λ (even if the actual system does not have Poisson arrivals). Further, we observe that $q(t)$ is a regenerative process when $\frac{\lambda}{c-x} < 1$, and $x < c$. Thus, from the ergodic theorem for a regenerative process for large enough T, the following definition on the equivalent rate based marking function, denoted by $p_r(x)$, is a good approximation of the marked volume of data at the router:

$$p_r(x) = \begin{cases} E_{\pi_\lambda^{c-x}}[p_q(Q)] & \text{if } \frac{\lambda}{c-x} < 1 \text{ and } x < c, \\ 1 & \text{if } x \geq c \text{ or } \frac{\lambda}{c-x} \geq 1, \end{cases} \tag{2}$$

where Q is the stationary queue length random variable and π_λ^{c-x} is the stationary distribution of an M/D/1 queue with capacity $c - x$ and arrival rate λ.

In the next section, we describe how our theoretical model of this section is practically implemented in the FluNet.

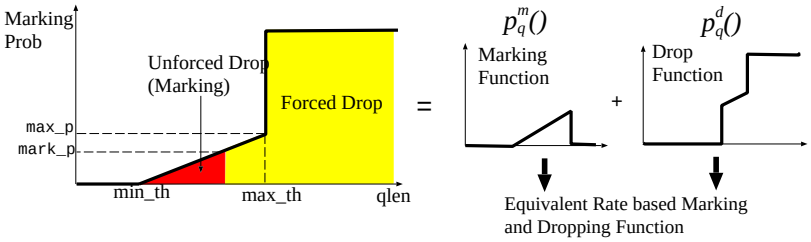


Figure 5: Marking and dropping in RED with marking mode

3.2 Implementation Issues

3.2.1 Choice of Measurement Interval and Simulation Step Size

Implementing an approximate marking function based on (2) requires measurement of x (TCP arrival rate) and λ (unresponsive arrival rate). In our implementation, this is done by choosing a small measurement interval $\bar{M}$, which satisfies *(i)* that the arrival rate from TCP flows does not vary significantly over $\bar{M}$, and *(ii)* that there exists a sufficient number of regenerative cycles in the router queue over $\bar{M}$, enabling us to see the statistical stationary behavior (i.e., the Poisson approximation in Theorem 3.1 holds for the chosen value of $\bar{M}$). In our experiments, we have chosen one (minimum over end-to-end flows) round-trip time for $\bar{M}$, which is independent of the system scale size, n. This choice seems to be reasonable, since the transmission rate of a TCP flow is clocked by the received ACK feedbacks, and we can also see a significant queue variations over one round-trip time (a large number of regenerative cycles) in large scale systems (e.g., in the plots of Figures 3 and 4, even over $\frac{1}{4}$ of a round-trip time in the router with 100 flows, we can see more than 20 cycles of queue regeneration). Note that since we focus on the flows passing through a WAN, the minimum round trip-time over flows will be large enough to see the stationary behaviors at the core-routers.

Additionally, we split a measurement interval $\bar{M}$ into the W sub-intervals (i.e., $\delta = \frac{\bar{M}}{W}$), over which the system states evolves. If this "splitting" of a measurement interval is not done, ACKs will aggregate over a measurement interval at a source, and lead to spurious bursts of packets, resulting in incorrect end-system behaviors. This time-step-size is significantly smaller than the time-step-size in a queue-tracking fluid simulator (e.g., [2]), which requires at least one sampling of queue length over one regenerative cycle of the queue dynamics, leading to simulation time complexity reduction in FluNet. Note that $\bar{M}$ and W do not depend on the system scale size, n. W depends only on the number of packets per flow in a round-trip time (i.e., congestion window size), so that the packets can be "clocked out" without spurious bursts[5]. Thus, as W and $\bar{M}$ are independent of n, δ is also independent of n. This is the key observation that leads to the reduced simulation time complexity for large scale systems.

3.2.2 Marking and Dropping in AQM

A realistic fluid model of a AQM algorithm should incorporate packet dropping as well. This is due to the fact that *(i)* a queue has a finite queue limit size, and *(ii)* some AQM algorithms drop the incoming packet when the current

[5]Note that this *does not* mean that "valid" bursts are not allowed. If a burst of ACKs arrive at a TCP source, the TCP source will in fact send a burst of packets by ACK-clocking.

(averaged) queue length is larger than a threshold. For instance, the RED [24] algorithm marks packets below a queue threshold, but drops packets if the current (average) queue length exceeds this threshold (see unforced and forced drop region in Figure 5). Packets are also dropped if the packet buffer overflows.

This dropping functionality can be approximated by and incorporated into the equivalent rate based fluid model simply by decomposing the queue based marking-dropping function into a pure drop and marking function ($p_q^m(\cdot)$ and $p_q^d(\cdot)$), as shown in Figure 5. Each of these two functions are then used in (2) to compute the equivalent rate based marking and dropping probability, denoted by $p_r^m(\cdot)$ and $p_r^d(\cdot)$, respectively. Note that this is an approximation in the sense that the arrival rate is not "thinned" when computing the dropping probability. However, we do take into account flow "thinning" along a path of routers, see Section 4.2.2. We observe that this approximation is good as long as the long-term drop rates are small (see Section 5.6 for numerical results). We comment that a simple Drop-Tail queue without marking functionality can also be approximated by setting its marking function to be always zero and its drop function to be a step function (1 if qsize $\geq$ qlimit, and 0 otherwise).

3.2.3 Implementation of $E[p_q^m(Q)]$ and $E[p_q^d(Q)]$

As discussed in Section 3.1 and 3.2.2, we have to compute $E[p_q^m(Q)]$ and $E[p_q^d(Q)]$ to determine the marking/dropping probability for a given fluid input rate. However, we may not always find a closed form expression to compute $E[f(Q)]$, where Q is the stationary queue length in an M/D/1 queue. Thus, our implementation uses numerically pre-computed values of the marking probability for each arrival rate (discretized suitably). This computation is performed using results in [29], where the authors have shown that the unfinished work U for a Poisson process with arrival rate λ, and with service rate μ has a steady state distribution of the form

$$\Pr(U > x) = 1 - (1-\rho)e^{\rho x} H_{\lfloor x \rfloor}(x - \lfloor x \rfloor), \tag{3}$$

where $H_n(x)$ are polynomial functions (which can be computed recursively as shown in [29]), and $\rho = \lambda/\mu$. Then, by denoting the stationary workload of an M/D/1 queue with capacity c by V, we have $V = Q/c$, and $U = \mu V = (c-x)V$. Thus,

$$\Pr(Q > q) \quad = \quad \Pr\left(U > \frac{q(c-x)}{c}\right) \tag{4}$$

This is used to off-line pre-compute a table of marking/dropping probabilities for a large set of input traffic parameters (λ, μ), and AQM parameters, and use table lookups during run-time.

3.3 On the Alternate Models and Discussion

We conclude Section 3 by providing comparative explanation on the alternate models for completeness. The alternate models are derived based on the different assumptions on (i) buffer size scaling and (ii) randomness in the network.

The buffer size scaling has been classified into three regimes [19]: (a) *small buffer regime* ($B = \Theta(1)$, i.e., independent of the system scale size n), (b) *intermediate buffer regime* ($B = \Theta(n^\alpha),\ 0 < \alpha < 1$), and (c) *large buffer regime* ($B = \Theta(n)$), where B is the buffer size at the bottleneck router. Traditionally, network design (at the back-bone routers) has been predicated on the large buffer regime (i.e., linearly proportional to n) [30]. However, recent experimental studies on the buffer size scaling show that we can achieve sufficient statistical multiplexing

gain and high network utilization in small buffer regime [26] as well as intermediate buffer regime [25]. Further, analytical results [31, 32] show that arbitrary small loss probability and high throughput are guaranteed even with small buffer regime. The small buffer regime is based on the intuition that a large number of flows multiplexed at the large capacity router and *randomness* (which leads to de-synchronization among flows) in the network enables sufficient statistical multiplexing without large buffer size.

The major sources of randomness in the network are generated by both unresponsive flows and TCP flows, i.e., real-time or short-lived flows, random session arrivals/departures and flow initiations/terminations. Different models can be derived, depending on the different assumptions on the randomness by these two types of flows, and their relative magnitude and time-scale of variability.

The authors in [17, 19] assume that the time-scales of randomness by both TCP and unresponsive flows are in the same order. Thus, aggregate TCP and unresponsive flows form a Poisson process with parameter $(\lambda + x)$, where λ and x are the mean arrival rates of unresponsive and TCP flows, respectively. Thus, the limiting system is approximated by an M/D/1 system with capacity c and Poisson input with parameter $\lambda + x$. On the other hand, our work in this paper and [18] assumes that the randomness of TCP flows happens at much slower time-scale than that of unresponsive flows, i.e., over a small interval of time, the mean arrival rate of TCP flows looks constant. This assumption leads to a M/D/1 system, but with capacity $c - x$ and Poisson input with parameter λ.

The assumptions and analytical models in [17,19] can be viewed as a "conservative" interpretation of the Internet. However, the models in [18] and this paper analyze the Internet on a "optimistic" assumption, i.e., the randomness of TCP flows due to inter-packet jitter over a short time-interval is ignored, where the randomness of unresponsive flows is dominant. Intuitively, in a system with the both TCP packet-jitter and unresponsive flows, more marking/dropping of packets occurs. FluNet can be implemented based on the model in either [17, 19] or that described in this paper. Both models are easily implementable using the method in Section 3.2.3. However, due to space limitations, we have provided simulation results using only the model described in this paper (our simulation experiments suggest that the results due to both models are very close with 100 or more flows).

4 FluNet Architecture

4.1 Architecture Summary

The entire FluNet framework consists of four components: *(i)* ingress fluid interface, *(ii)* fluid routers, *(iii)* egress fluid interface, and *(iv)* packet queue pool, as shown in Figure 6. The ingress and egress fluid interfaces reside at edge of the FluNet-core (a pair of ingress and egress fluid interfaces forms a single interface node), and there are multiple fluid routers (depending on the topology of simulated network) inside the FluNet-core, where a fluid router corresponds to a packet router in the Internet-core.

To illustrate, let us consider the two packet streams between the end-systems of (ES-1↔ES-2) and (ES-1↔ES-3). The packets from ES-1 (destined to a node in ES-2 or ES-3) are first transmitted to the ingress fluid interface, IFI-1, which classifies them into two *classes* depending on the destination end-systems, ES-2 and ES-3. IFI-1 records per-class packet transmission *rate information* over successive time steps. The IFI-1 separately stores the received real packets at a queue (identified by a tuple (ingress fluid interface, egress fluid interface)) belonging to the packet queue pool, and forwards only rate information (not real packets) into the FluNet-core. The forwarded rate

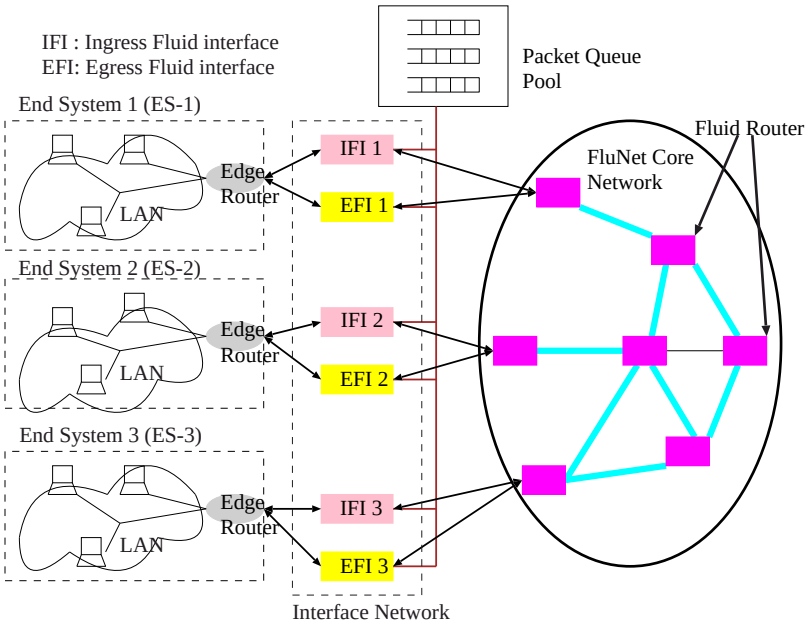


Figure 6: FluNet Architecture

information is processed by the FluNet-core, and the associated marked/dropped rate is computed at each fluid router using the equivalent rate based marking/dropping model in Section 3. The updated marked/dropped rate information is finally transferred to the egress fluid interfaces, EFI-2 and EFI-3, which marks/drops the real packets (which are fetched from the packet queue pool) physically based on the rate information computed at the fluid routers in the routing path. The EFI-2 and EFI-3 forwards the real packets to the destination networks in ES-2 and ES-3. We will describe the detailed procedure and issues of each component in Section 4.2.

The important thing that we note is that the real packets do not traverse the FluNet-core. The real packets are stored at the associated queue (corresponding to the ingress-egress pair) in the packet queue pool. Since only aggregate flow information between end-systems is transfered inside the FluNet-core, the maximum number of (aggregate) flows present within the FluNet-core is just $I \times (I-1)$ (e.g., $I = 3$ in Figure 6).

In this paper, we focus on the data traffic injected only by end-packet systems, and the queue regimes at the intermediate routers at the FluNet-core. We can easily extend our analysis and implementation to more general scenario, where there exists data flows originated from and destined to the nodes inside the FluNet-core. The fluid representation of such flows inside the FluNet-core can be easily made by the techniques in [3,5], and apply the fluid rates to the input of our rate based marking/dropping models.

4.2 Description of Components

Inside the FluNet-core, we model the network as a directed graph. We denote this graph by $G = (V, L)$, where V is a set of nodes (corresponding to routers in the real network) and L is a set of links. Associated with each link $l \in L$ is a propagation delay of γ_l. Let us denote $I \subset V$ to be a set of (ingress/egress) interface nodes, and we also denote by I the number of interface nodes to abuse the notation. We define a class to be a single fluid-flow within

the FluNet-core which experiences the same route inside the FluNet-core[6]. Denote by L_k a sequence of links of the class-k path, and again we use L_k to refer to the number of links of the path to abuse the notation, for simplicity.

4.2.1 Ingress Fluid Interface

At each ingress fluid interface, a step size δ is chosen (based on the criterion discussed in Section 3.2.1), and the total number of bytes that arrive from the end-system edge routers over this interval for each class is recorded over successive time step intervals. This recorded rate information is transferred to the first link inside the FluNet-core in the routing path of the corresponding traffic class. The incoming real packets are stored in a packet queue (indexed by its class) of the packet queue pool.

At the end of each time step $s = 1, 2, \ldots$, a rate information vector which includes the information on the received volume of (1) total, (2) marked, (3), dropped, (4) TCP, and (5) unresponsive data, is generated corresponding to the aggregate data volume over the s-th time step for class k. Note that the sum of TCP and unresponsive data volume equals the total data volume, and the sum of marked and dropped data volume equals the TCP data volume. At an ingress fluid interface, checking the IP packet header CE and ECT field enables us to know whether a packet is TCP or unresponsive, or marked or unmarked.

4.2.2 Fluid Router

The main task of a fluid router is to update the marked/dropped volume of data in the rate vector transferred from an ingress fluid interface or the previous fluid router in the routing path by applying the rate based fluid model in Section 3, and to finally transfer those vectors to the associated egress fluid interface.

We now describe the traffic interactions inside the FluNet. We first denote by $c_l[s]$, $u_l[s]$, $t_l[s]$, $m_l[s]$, and $d_l[s]$, the received volume of TCP, unresponsive, total, marked, and dropped data [7] of a class k at link $l \in L_k$ over the s-th time step, respectively. We denote by l' the previous link of link l in the path of L_k. Denote by s' the time step right before the data is transmitted over link l', i.e., $s' = s - \lceil \gamma_{l'}/\delta \rceil$.

Associated with every link l are the (rate based) marking and dropping probabilities, $p_l^m[s]$ and $p_l^d[s]$, which are computed by (2) in Section 3 [8]. Then, we have

$$c_l[s] = \Big(1 - p_{l'}^d[s']\Big) c_{l'}[s'] \tag{5}$$

$$m_l[s] = \Big(t_{l'}[s'] - m_{l'}[\tilde{s}]\Big) p_{l'}^m[s'] + m_{l'}[s']. \tag{6}$$

Note that mean queueing delays can be easily incorporated by adding it to the link propagation delays. However, in sufficiently large scale systems, queueing delay is negligible, compared to link propagation delay [9]. Analogous equations to (5) and (6) hold for the unresponsive arrival rate ($u_k^{l_{k,i}}[s]$) and the drop rate ($d_k^{l_{k,i}}[s]$), respectively. Finally, the total arrival rate of each class k is computed by the sum of TCP and unresponsive arrival rate, i.e.,

$$t_l[s] = c_l[s] + u_l[s]. \tag{7}$$

[6] A class corresponds to multiple packet flows over the same path in the FluNet-core.

[7] Since our explanation can be generally applied to any class, we have omitted the class index k here for notational simplicity.

[8] We eliminate the subscript 'r' in (2) for notational simplicity.

[9] This is also justified by [25], where the authors show that buffers need to scale only as $\sqrt{n}$, whereas the capacity scales with n, which implies that the queueing delay is $O(\frac{1}{\sqrt{n}})$, while the propagation delay is $\Theta(1)$.

Notice that update of $c_l[s]$ and $u_l[s]$ due to the dropping probability is needed, since it affects the TCP and unresponsive arrival rate at *the next link* (and thus, affects the computation of marking/dropping probability of next link). As discussed in Section 3.2.1, in computing $p_l^m[s]$ and $p_l^d[s]$, we need the TCP and unresponsive rate information over $\bar{M}$ interval as their input. In our implementation, we use the TCP/unresponsive volume of data averaged over past W time steps, i.e., $\bar{c}_l[s] = \sum_{j=1}^{W} \frac{c_l[s-j+1]}{W}$, and similarly, $\bar{u}_l[s]$.

4.2.3 Egress Fluid Interface

At the egress of the FluNet-core is the egress fluid interface connected to the destination end-system. When a rate vector, corresponding to a particular class leaves the last fluid router in its routing path, this vector enters the egress fluid interface of the corresponding destination end-system. At each time-step s, the egress fluid interface $i \in I$ maintains a collection of vectors of *byte-counters* (not necessarily integers) for every class (which has the node i as its egress inside the FluNet-core), where byte-counters for, say, class k consist of: *(i)* total counter ($T[s]$), *(ii)* mark counter ($M[s]$), and *(iii)* drop counter ($D[s]$). These byte counters keep track of the amount of total data, the amount of marks, and the amount of drops received for the class k at each time step. Note that the egress fluid interface does not need the information of TCP or unresponsive data rate, since they are used only in computing marking/dropping probability at the fluid routers. At the end of each time-step, when a vector, (t_l, m_l, d_l) is transferred to the egress fluid interface, the interface increments three counters by the incoming quantities.

At each time-step s, if $T[s]$ is larger than or equal to the size of a packet in the class-dependent packet queue in the packet queue pool, the head-of-line (HOL) packet from the packet queue is transferred to the corresponding *destination egress fluid interface*. Then, the egress fluid interface determines how to handle this real packet: drop, forward with marking, or forward without marking. First, the fetched real packet is dropped with probability $D[s]/T[s]$. Next, if the packet is not dropped, the value of the mark counter is now checked, and the ECN bit of the real packet is set to the value '1' with a probability $M[s]/T[s]$. This is simply an implementation of probabilistic marking corresponding to the fluid mark/drop rate. This packet is now forwarded to the edge-router of the destination end-system, which will suitably forward the packet to its final destination.

Once the packet has been forwarded or dropped, the byte counters are updated by the following:

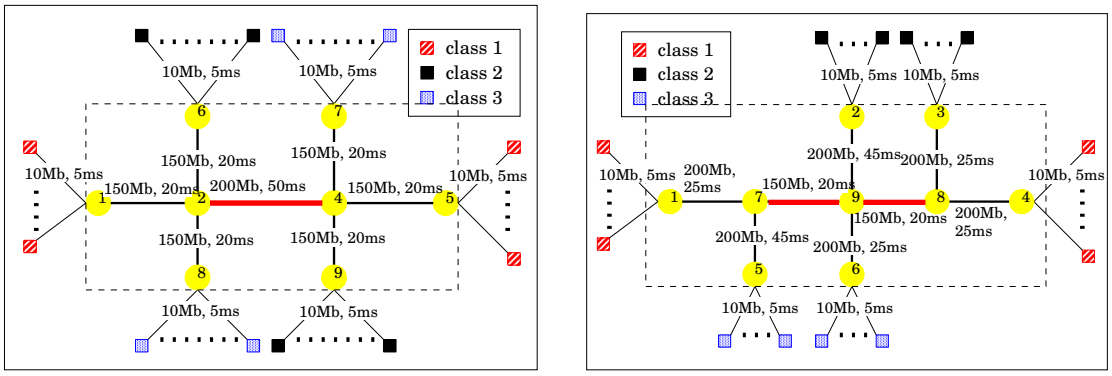
$$M[s] = M[s](1 - \text{psize}/T[s]), \quad D[s] = D[s](1 - \text{psize}/T[s]), \quad T[s] = T[s] - \text{psize},$$

where "psize" is the size of the packet that is fetched and processed. The same procedure is repeated unless $T[s]$ is smaller than the HOL packet in the associated per-class packet queue.

5 Simulation Results

In this section, we provide extensive simulation results to validate the performance of FluNet. We have implemented FluNet in *ns-2*. To implement FluNet, we have modified the fluid network implementation in *ns-2* developed by [2], where their fluid queueing model (router) is completely replaced by our rate based (fluid) model with consideration of practical issues in discussed Section 3.2.

Our objective here is to compare a queue-tracking fluid simulation (QFM) in [2] with FluNet, and determine the regimes for which each is suitable. Our base-line system for comparison is a "packet" system, where no fluid approximations/models are used.



(a) a single bottleneck network

(b) dumbbell network

Figure 7: One bottleneck network and dumbbell network

5.1 Simulation Environment

In the simulation results that use RED [24] as an AQM algorithm, the parameters in RED are set to: `w_q` = 0.002, `gentle_` = false, `max_p` = 0.02, `mark_p` = 1.0, and `adaptive_` = false, most of which are the default values in *ns-2*. The main parameters that we vary are the queue threshold values (i.e., `min_th` and `max_th` in RED, and the physical queue length in DropTail), which determines the fast or slow queueing regime for a given time-step-size. Other network environments will be described in each set of simulations.

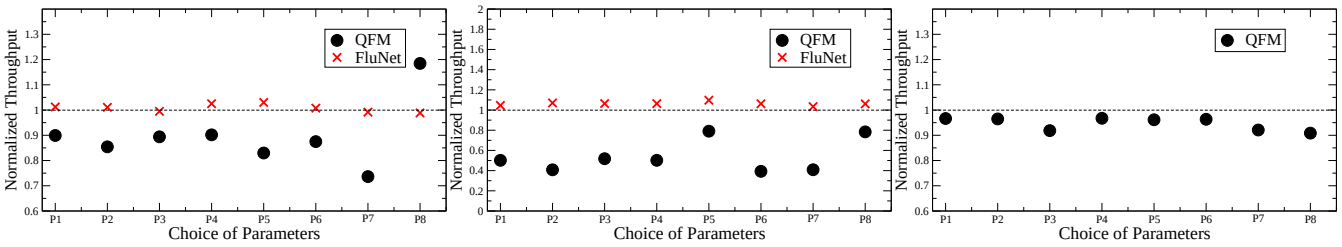
As suggested in [33], in our simulations, we use TCP Sack for end-system elastic data flows, and ON-OFF processes for unresponsive flows, respectively. In each ON-OFF process, the burst size of ON and OFF periods are exponentially distributed with mean 50 msec. The packet size for both TCP and the unresponsive flows are set to be 1000 bytes. In the simulation results, we present three statistics to investigate both steady-state and transient behavior: *(i)* normalized average throughput (over flows and time) w.r.t the packet system, *(ii)* CWND (congestion WiNDow) traces and instantaneous throughput averaged over flows, and *(iii)* average CWCR (Congestion Window Cut Ratio) over flows and time[10].

According to the selection rule of measurement interval and step size, we use $\bar{M} = 200$ msec, and $W = 40$, resulting in the step-size being set to 5 msec (i.e., $\delta = \frac{200}{40}$), unless explicitly specified. The value of $W = 40$ is sufficient to avoid spurious bursts of transmitted TCP packets, since the steady-state congestion window size less than 25 packets, as in Figure 9.

5.2 A single bottleneck network and a dumbbell network

In this experiment, we present simulation results with four varying parameters: `min_th` , `max_th` , number of unresponsive flows, and total volume of unresponsive flows in a single bottleneck (SINGLE) and a dumbbell (DUMBBELL) network, where there are three (TCP, unresponsive) traffic classes depending on the source-destination access network pair. Each choice of these parameters are denoted as P1-P8 (see Table 1), for different values of queue threshold values and the per-class number/volume of total unresponsive flows. Further, for each traffic class, we

[10]The CWCR represents the number of window cut events divided by the number of total transmitted packets, where window cut events include triple duplicate ACKs, retransmission timeouts, and ECN responses.



(a) a single bottleneck: step size = 5 ms

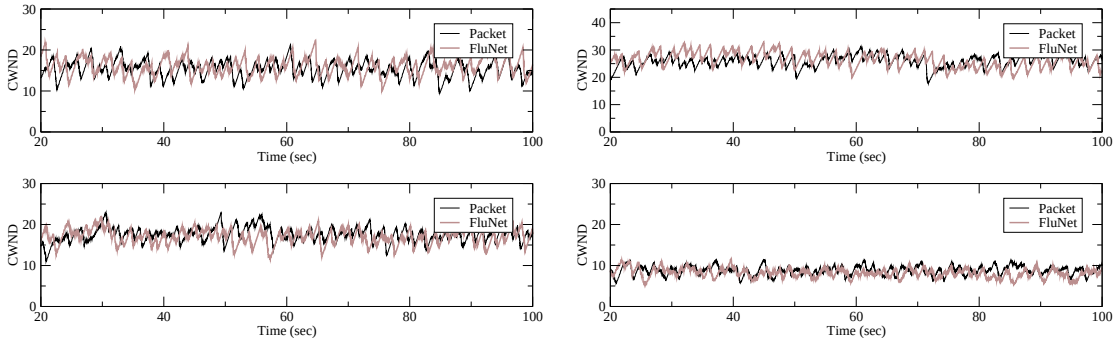
(b) a dumbbell network: step size = 5 ms

(c) a single bottleneck with 10 × queue thresholds as those in (a), and step size = 1 msec

Figure 8: Normalized average throughput of QFM and FluNet

Table 1: Average CWCR for SINGLE and DUMBBELL. (a,b,c,d) = (`min_th` , `max_th` , num of unresp flows, vol of unresp flows).

Name	Parameters	L1 ($\times 10^{-3}$)			L2 ($\times 10^{-3}$)		
		Pkt	FlN	QFM	Pkt	FlN	QFM
P1	(10,100,30,30Mb)	4.83	4.69	8.18	4.94	4.93	18.7
P2	(30,100,30,30Mb)	4.84	4.67	9.22	4.81	5.11	25.9
P3	(10,50, 30,30Mb)	5.06	4.95	9.05	5.32	4.95	17.9
P4	(10,150,30,30Mb)	4.67	4.54	7.72	4.69	4.45	7.22
P5	(30,100,10,30Mb)	4.91	4.66	9.34	5.28	4.74	28.1
P6	(30,100,90,30Mb)	5.15	4.90	9.76	5.22	4.93	26.6
P7	(30,100,30,10Mb)	2.45	2.29	6.36	2.72	2.62	4.63
P8	(30,100,30,50Mb)	14.4	14.4	15.2	12.7	11.3	17.8



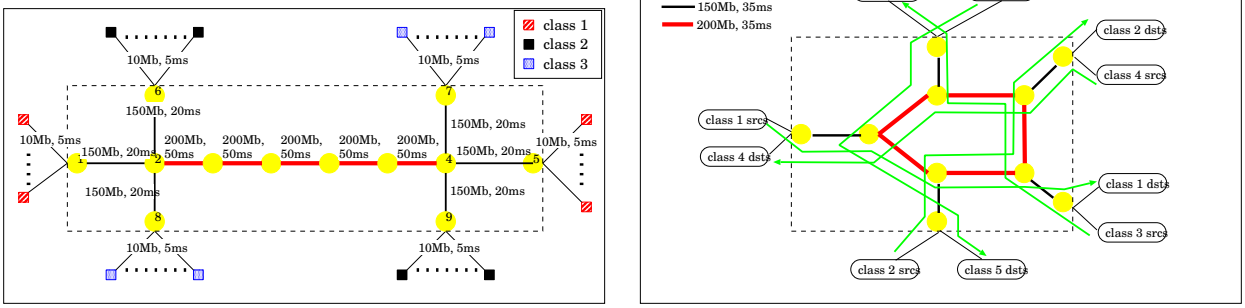
(a) # of UFs (10/90)

(b) vol of UFs (10/50) Mb

Figure 9: CWND traces for SINGLE topology: `min_th` = 30, `max_th` = 100.

run 50 TCP sources. We show only the performance results of class 1 TCP sources due to space limitation (similar results are seen for other classes).

Figure 8 and Table 1 summarize the simulation results of normalized average throughput and CWCR. Further, Figure 9 shows the associated CWND traces for selected experiments. As the experiments in Section 2 indicate, the parameter choices of queue thresholds and step-size in Figure 8 and Table 1 induce fast queue regimes. The results



(a) a long-route network

(b) a circular network with multiple bottlenecks

Figure 10: A network with long-route connections and circular interactions

of FluNet match those of a pure packet network within an error of 5% for all cases considered. On the other hand, QFM has up to 45% throughput error, compared to the packet simulation.

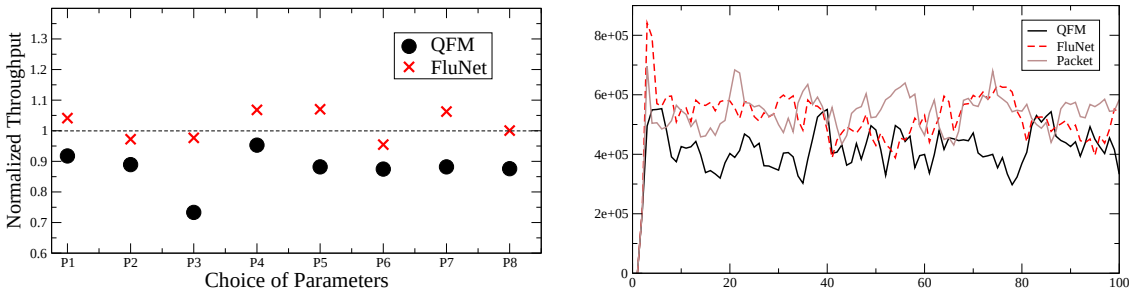
However, as shown in Figure 8(c), by decreasing the step-size to 1 msec and increasing the queue thresholds (ten times as those in (a)), *the results of QFM have a good match with those of packet systems.* This is because with a step-size of 1 msec and with the large mean queue size, queue fluctuations are not severe, enabling QFM to accurately track queue dynamics. This simulation results support the fact that QFM and FluNet should be incorporated together depending on the queue regimes. However, the simulation results in this subsection also shows that FluNet can achieve the comparable performance with low cost (i.e., larger step size).

FluNet is based on asymptotic models for router queues, where the number of flows are large. Indeed, FluNet simulation results match packet simulations closely when there are a large number of flows. For example, in a system with 4,200 flows and the topologies of Figure 7, the difference in average throughput between the packet system and FluNet is less than 4%. Importantly, in this section, we have shown that FluNet performs well even in a moderately scaled system, with only a few hundred flows. We skip the details for the large system (with many thousand flows) due to space constraints.

5.3 Network with multiple bottlenecks and long-route connections

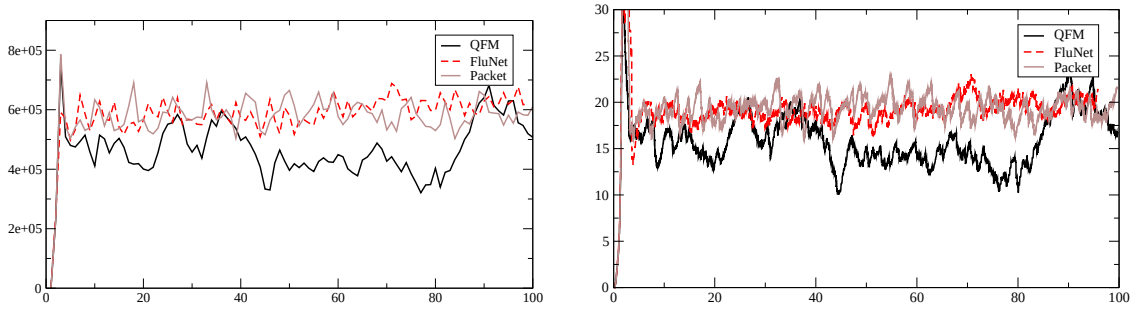
Now, we compare the performance of FluNet in a network with long-route connections, shown in Figure 10(a), the total number of hops in of end-to-end path is nine. We also carry out the simulation for the network with circular interactions with multiple bottlenecks, as shown in Figure 12. Again, Figures 11 and 12 show that FluNet has a good match with the packet system, whereas the performance of QFM deviates from that of the packet system more than that of FluNet.

We observe that the deviation of QFM manifests as lower throughput, and this happens across many simulation results that we have carried out throughout this section. A possible explanation of this is that QFM fails to adequately track the queue dynamics in a fast queue regime, thus leading to an overestimation of the queue lengths.



(a) Normalized average throughput (b) Instantaneous throughput (class-1 TCP flows)

Figure 11: Comparison of FluNet and QFM for long-route connections, where `min_th` $= 10$, `max_th` $= 50$, the total per-class number/volume of on-off flows = 30 and 30 Mbps, respectively. (a) shows the normalized average throughput for eight scenarios described in Table 1. (b) shows the instantaneous throughput measured over every second,



(a) Instantaneous throughput (class-1 TCP flows) (b) CWND traces (class-1 TCP flows)

Figure 12: Comparison of FluNet and QFM for circular interactions with multiple bottlenecks, where `min_th` $= 30$, `max_th` $= 60$, the total per-class number/volume of on-off flows = 80 and 50 Mbps, respectively.

5.4 Connections with different round-trip times

Now, we move on to a heterogeneous setup, where we have connections with different round-trip times. In the single bottleneck topology in Figure 7(a), we have changed the delay of links, such that classes 1, 2, and 3 experience 60 msec, 100 msec, and 200 msec round-trip times, respectively. Figure 13 shows that FluNet again has a good match with the packet system for different configuration of queue thresholds and unresponsive flows. Note that in a network with connections of heterogeneous round-trip times seem to generate more randomness due to asynchronous interactions among flows. This intuition is shown in the simulation results, where QFM has much worse performance than the homogeneous setups in earlier subsections.

5.5 Dynamic connections

In this experiment, we investigate the response of FluNet when flows dynamically enter and leave the system, where we have tested the cases when TCP and unresponsive flows are dynamic, in the topology of Figure 7(a). In the TCP-dynamic scenario, we have used the following scenario, where the number and volume of per-class total

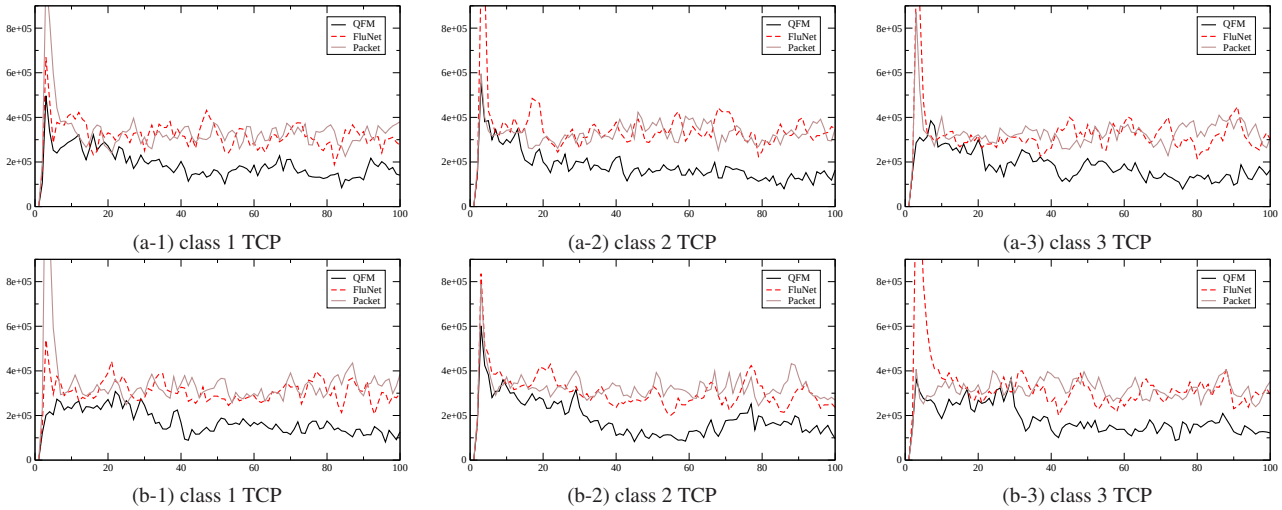
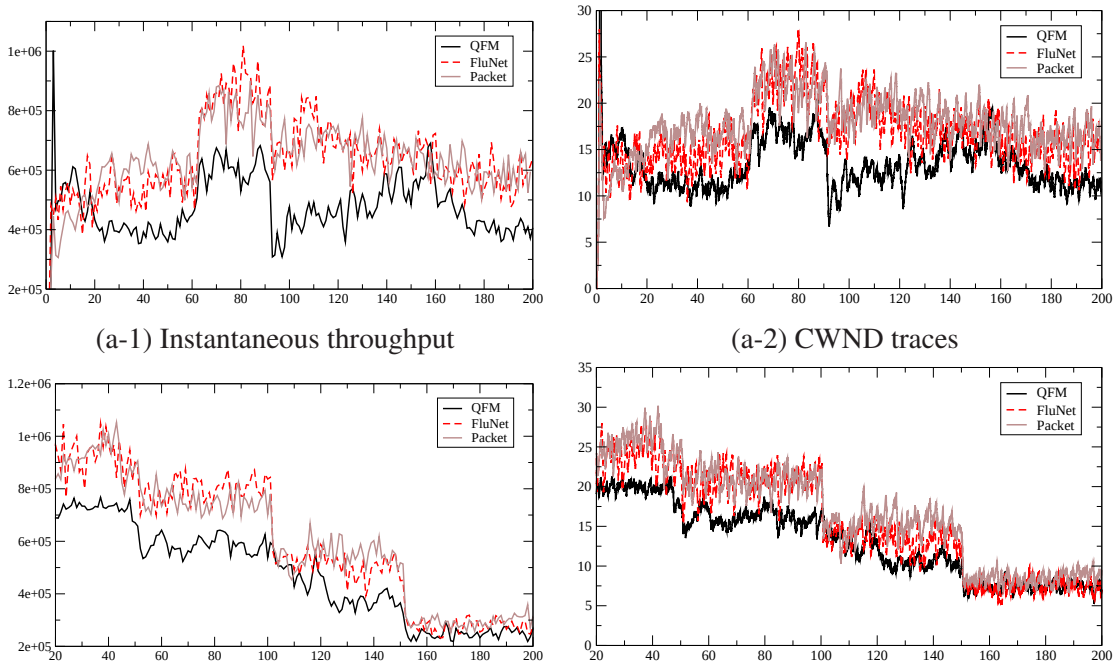


Figure 13: Comparison of FluNet and QFM for connections of different round-trip times. (a) `min_th` $=$ 10, `max_th` $=$ 100, the total per-class number/volume of on-off flows = 30 and 30 Mbps and (b) `min_th` $=$ 30, `max_th` $=$ 100, the total per-class number/volume of on-off flows = 90 and 30 Mbps.

unresponsive flows are 30 and 30 Mbps: At time 0, we add 50 TCP flows per each class, at time 60, we remove the half of the class-1 and class-3 TCP flows, at time 90, we add 25 class-2 TCP flows, at time 120, we add 25 class-1 TCP flows, and finally, at time 150, we add 25 class-3 TCP flows and remove 25 class-2 TCP flows. In the UDP-dynamic scenario, for a fixed number of 50 TCP flows in each class and a fixed 500 Kbps mean rate of a unresponsive flow, we increasingly add the number of unresponsive flows over time. In each class, we start the system with 20 unresponsive flows, and then we add 30 unresponsive flows in every 50 secs. Since all the unresponsive flows of three class shares the same bottleneck link, this dynamic scenario corresponds to the total unresponsive flow load ranging from 15% ($= \frac{20\times3\times0.5}{200}$) to 82.5% ($=\frac{110\times3\times0.5}{200}$). We observe that FluNet reacts to both of these changes in a similar manner to that in the packet system, whereas the performance of QFM is lower than that of the packet system in most cases.

5.6 DropTail Queues

In this experiment, we provide the simulation results for DropTail queues. First, we comment that our results include only comparison of FluNet with the packet system, since the QFM code which is available in public does not support DropTail. One can emulate DropTail queue by setting the RED parameters (e.g., `min_th` =`max_th` =qlimit and $w_q = 1$, where w_q is the weight assigned to the instantaneous queue length in the moving average). However, our simulation experiments suggest that with such an emulation, the throughput with QFM throughput drops close to zero, which is due to $w_q = 1$. We conjecture that this phenomenon happens due to the current version of the fluid model in QFM, where the fluid (differential equation) for average queue length is not well-defined for this condition (the RHS in [5, eqn (2)] becomes indeterminate for this case, which corresponds to $\alpha = 1$ in [5, eqn (2)]). Table 2 shows the normalized average throughput and CWCR values of class-2 flows, and Figure 15 shows an instantaneous average throughput for two different physical queue limit values. These results also show that FluNet's ability to match with the packet system. However, we comment that if we use a large value of the queue limit, then FluNet's



(a-1) Instantaneous throughput

(a-2) CWND traces

(b-1) Instantaneous

(b-2) CWND traces

Figure 14: Comparison of FluNet and QFM for the dynamics of TCP and unresponsive flows, `min_th` $= 10$, and `max_th` $= 50$. Both (a) and (b) correspond to the performance of class-2 flows.

performance become worse, again, corresponding to fast and slow queue regimes.

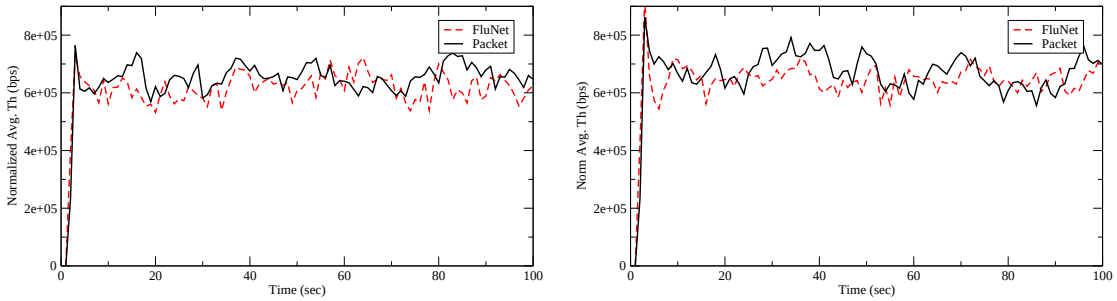
Table 2: Normalized Average Throughput and Average CWCR values

Physical Queue Limit	Packet		FluNet	
	Norm. Avg. Thput (kbps)	CWCR ($\times 10^{-3}$)	Norm. Avg. Throughput	CWCR
20	572	4.94	613	5.21
50	610	4.34	653	4.88
80	635	4.21	662	4.63
110	643	4.06	675	4.52

6 Concluding Remarks

In this paper, we have presented an approach to hybrid packet/fluid simulation using the equivalent rate based models of router queues. Under a fast queue regime, our approach enables us to simulate large-scale systems, where the simulation step-size is governed only by the time-scale of the end-systems, and not that of the queueing dynamics at the intermediate routers. This gives us a significant reduction of both simulation time complexity, compared with queue tracking based hybrid simulation.

Further, we have implemented the FluNet-core (the fluid model part of FluNet) using multiple processes in the



(a) Instantaneous Throughput (qlimit=50) (b) Instantaneous Throughput (qlimit=80)

Figure 15: FluNet's performance for DropTail queue

user-level space in a Linux PC. Network interface cards (Fast Ethernet cards) along with appropriate packet capture libraries [34] have been used to connect external PCs (TCP sources/sinks) with the FluNet-core, and thus has enabled hybrid fluid/packet emulation (as opposed to ns-2 based simulation). Measurements based on this system have shown that we can enable real-time hybrid emulation using this approach.

However, it is possible that we can be in a regime where fast queue regimes do not occur (e.g., by having large queue threshold parameters or in a small-scale system), where FluNet does not seem to perform well. In a real network, depending on the number of traversing flows and capacity at intermediate routers, fast and slow queue regimes can occur at different times and routers. In this case, queue-tracking fluid simulation and rate model based simulation should be chosen appropriately. A simple approach to differentiate between both regimes is to measure the number of regenerative cycles in a chosen step-size, and apply one of both fluid simulation models to mark/drop the packet at the intermediate routers.

Further, queueing delays could be significant with such a large queue operating size, which are ignored in the current implementation of FluNet (note, however, that FluNet can incorporate mean queueing delays, simply by adding an extra parameter to the round-trip propagation delay). In this case, a more accurate approach is to implement fluid queues to track the queue variation, and determine marking/dropping decisions based on the queue length (at the cost of having state in the simulator to keep track of the queue lengths). Thus, we believe that a good approach to hybrid simulation is to use both queue based method (such as in QFM [2]) as well as rate based method (such as FluNet), depending on the type of the system under study.

Note that there are some common disadvantages of hybrid simulation scheme (either with or without fluid queues). For example, algorithms that dynamically operate on a per-packet basis (such as per-packet routing) cannot be directly studied. With FluNet, only statistical queue behavior can be measured, whereas QFM can capture a finer granularity of queue dynamics (even if its granularity is larger than that of the pure packet system). An approach to tackling this class of problems is to couple analytic models of the aggregate behavior of per packet algorithms at the fluid time-scale (a time-scale decomposition model), and use these models in the fluid network. Our future work will focus on these techniques.

Appendix

6.1 Assumptions and Proof of Theorem 3.1

Let

$$\begin{aligned} X_n(t) &= \sum_{i=1}^{n} x_n^i(t), \\ Y_n(t) &= \int_0^t X_n(z)dz = n\int_0^t x_n(z)dz, \\ A_n(t) &= \sum_{i=1}^{n} A_n^i(t), \end{aligned}$$

where $X_n(t)$ is the *total arrival rate* at time t across all n TCP flows, and $Y_n(t)$ is the total cumulative volume of arrivals of the TCP flows until time t. $A_n(t)$ represents the total cumulative volume of arrivals of the unresponsive flows.

For TCP flows, recall that for the well-defined initial conditions, we assume that $x_n^i(0) \to x^i(0)$, and $x_n(0) \to x(0)$, as $n \to \infty$. We assume that $\dot{x}_n^i(\cdot)$ is bounded by some constant L (i.e., the transmission rate is Lipschitz). This in-turn implies that $x_n^i(\cdot)$ is Lipschitz continuous with some parameter $M < \infty$ [11]. Further, we assume that $p_m^q(\cdot)$ is also Lipschitz continuous.

For unresponsive flows, we assume that each $A_n^i(t)$ has the same distribution as a simple stationary point process A that satisfies the following assumptions [31, 35]:

Assumption 6.1.

(i) $\exists\, \lambda > 0$, *s.t.* $E[A(t)] = \lambda t$, $t \in [0, \infty)$.

(ii) $\exists\, \theta_0 > 0$ *and* $K < \infty$, *s.t.* $\lim_{t\to 0^+} E[e^{\theta_0 A(t)} \boldsymbol{I}_{\{A(t)>K\}}] = 0$. [11]

(iii) $\liminf_{t\to\infty} \frac{t\Lambda(x,t)}{\log t} > 0$, *where* $\Lambda(x,t) = \sup_{\theta\in\mathcal{R}}[\theta x - \frac{1}{t}\log E[e^{\theta A(t)}]]$.

Assumption 6.1 states that each unresponsive arrival process satisfies the properties that *(i)* multiple packets from a single unresponsive source do not arrive at the same time, *(ii)* all arriving packets are of the same size and *(iii)* the unresponsive arrival process has a finite intensity (see [31] for further details).

We define $q(t)$ to be a queue length process $q(t)$ to be the queue length process of M/D/1 queue with Poisson arrival rate λ and capacity $c - x(0)$, i.e.,

$$q(t) = \sup_{r\in[0,t]} [a(t) - a(r) - (c - x(0))(t - r) + q(r)]. \tag{8}$$

For a fixed $T < \infty$, consider the time interval $[0, T/n]$. For any $s \in [0, T/n]$, the *instantaneous* queue length process is given by:

$$Q_n(s) = \sup_{r\in[0,s]} [A_n(s) - A_n(r) + Y_n(s) - Y_n(r) - nc(s - r) + Q_n(r)]$$

[11] $\mathbf{1}_B = 1$ if the event B is true, and 0 otherwise.

$$= \sup_{r\in[0,s]}\left[A_n(s) - A_n(r) + n\int_r^s x_n(z)\,dz - nc(s-r) + Q_n(r)\right].$$

Recall that $\bar{Q}_n(t) = w_n\bar{Q}_n(t-\delta_n) + (1-w_n)Q_n(t)$, where $0 < w_n < 1$ and $\delta_n = 1/nc$.

Now, let us study the processes $(X_n, Y_n, A_n, Q_n, \bar{Q}_n)$ over a *slowed-down* time-scale, i.e., for $t \in [0,T]$, let $q_n(t) = Q_n\left(\frac{t}{n}\right)$, $\bar{q}_n(t) = \bar{Q}_n\left(\frac{t}{n}\right)$, $a_n(t) = A_n\left(\frac{t}{n}\right)$, and $y_n(t) = Y_n\left(\frac{t}{n}\right)$. Then, from the definition of $Q_n(t)$ and $\bar{Q}_n(t)$, we have for any $t \in [0,T]$,

$$\begin{aligned} q_n(t) &= \sup_{r\in[0,t]}[a_n(t) - a_n(r) + y_n(t) - y_n(r) - c(t-r) + q_n(r)] \\ \bar{q}_n(t) &= w_n\bar{q}_n(t-1/c) + (1-w_n)q_n(t). \end{aligned} \tag{9}$$

We will show that

$$\bar{q}_n(t) \xrightarrow[w]{n\to\infty} q(t), \quad t\in[0,T] \quad \text{over} \quad \mathcal{D}([0,T]:\mathcal{R}^+), \tag{10}$$

where $\mathcal{D}([0,T] : \mathcal{R}^+)$ is the space of RCLL (Right Continuous with Left Limit) trajectories over the interval $[0,T]$, and $\xrightarrow[w]{n\to\infty}$ corresponds to the weak convergence (i.e., convergence in the distribution functions). Then, from Theorem 3.2 in [18], and using the Lipschitz continuity of $p_m^q(\cdot)$ and $x_n^i(t)$, the result follows. Thus, we focus on proving (10) in the rest of this proof.

First, we will prove that for any fixed $\epsilon > 0$, we can find N_1 such that $\forall n > N_1$,

$$||\bar{q}_n(t) - q_n(t)|| < \epsilon \quad \text{over} \quad \mathcal{D}([0,T]:\mathcal{R}^+), \tag{11}$$

where $||\cdot||$ is the Skorohod metric.

¿From (9), we have

$$||\bar{q}_n(t) - q_n(t)|| = ||w_n\bar{q}_n(t-\frac{1}{c}) + (1-w_n)q_n(t) - q_n(t)|| = w_n||\bar{q}_n(t-\frac{1}{c}) - q_n(t)|| \le w_n \sup_{t\in[0,T]}|q_n(t)|.$$

Then, (11) follows from that fact that $\sup_{t\in[0,T]}|q_n(t)|$ is almost surely finite, and from Assumption 3.1.

Next, from Theorem 3.2 in [18], we have

$$q_n(t) \xrightarrow[w]{n\to\infty} q(t) \quad \text{over} \quad \mathcal{D}([0,T]:\mathcal{R}^+).$$

Then, from the Skorohod representation theorem [36], we can find processes $q_n'(t)$ and $q'(t)$ in $\mathcal{D}([0,T]:\mathcal{R}^+)$ such that $q_n(t) \stackrel{\text{dist}}{=} q_n'(t)$, and $q(t) \stackrel{\text{dist}}{=} q'(t)$, where $\stackrel{\text{dist}}{=}$ means "equivalence in distribution." Then, for the same ϵ in (11), we can find N_2 such that $\forall n > N_2$,

$$||q_n'(t) - q'(t)|| < \epsilon \quad \text{over} \quad \mathcal{D}([0,T]:\mathcal{R}^+). \tag{12}$$

Let $\bar{q}_n'$ be the weighted averaged process of q_n'. By the triangle inequality of Skorohod norm, for the same ϵ in (11) and (12), $\forall n > N = max(N_1, N_2)$, we have

$$||\bar{q}_n' - q'(t)|| < ||\bar{q}_n' - q_n'(t)|| + ||q_n'(t) - q'(t)|| < \epsilon + \epsilon = 2\epsilon.$$

Since ϵ is arbitrary, this completes the proof.

References

[1] Network Simulators, http://www.icir.org/models/simulators.html.

[2] Y. Gu, Y. Liu, and D. Towsley, "On Integrating Fluid Models with Packet Simulation," in *Proceedings of IEEE INFOCOM*, March 2004.

[3] V. Misra, W.-B. Gong, and D.Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proceedings of ACM SIGCOMM*, 2000.

[4] D. M. Nicol and G. Yan, "Discrete event fluid modeling of background TCP traffic," *ACM Transactions on Modeling and Computer Simulation*, vol. 14, no. 3, July 2004.

[5] Y. Liu, F. L. Presti, V. Misra, D. Towsley, and Y. Gu, "Fluid models and solutions for large-scale IP networks," in *Proceedings of ACM SIGMETRICS*, June 2003.

[6] F. Baccelli and D. Hong, "Flow level simulation of large IP networks," in *Proceedings of INFOCOM*, San Francisco, CA, April 2003.

[7] T. Yung, J. Martin, M. Takai, and R. Bagrodia, "Integration of fluidbased analytical model with packet-level simulation for analysis of computer networks," in *Proceedings of SPIE*, 2001.

[8] B. Melamed, S. Pan, and Y. Wardi, "Hybrid discrete-continuous fluid-flow simulation," in *Proceedings of ITCOM, Scalability and Traffic Control in IP Networks*, August 2001.

[9] G. Riley, R. Fujimoto, M. Ammar, K. Permula, and D. Xu, "Distributed network simulations using the dynamic simulation backplane," in *Proceedings of International Conference of Distributed Computing Systems*, 2001.

[10] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka, "A hybrid systems modeling framework for fast and accurate simulation of data communication networks," in *Proceedings of ACM SIGMETRICS*, 2003.

[11] S. Shakkottai and R. Srikant, "Mean FDE models for Internet congestion control under a many-flows regime," *IEEE Transactions on Information Theory*, vol. 50, no. 6, June 2004.

[12] D. R. Figueiredo, B. Liu, Y. Guo, J. Kurose, and D. Towsley, "On the efficiency of fluid simulation of networks," *Computer Networks*, vol. 50, no. 12, pp. 1974–1994, 2006.

[13] CAIDA, http://www.caida.org.

[14] C. V. Hollot, Y. Liu, V. Misra, and D. Towsley, "Unresponsive flows and AQM performance," in *Proceedings of INFOCOM*, 2003.

[15] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Inferring TCP connection characteristics through passive measurements," in *Proceeding of INFOCOM*, March 2004.

[16] F. P. Kelly, "Models for a self-managed Internet," *Philosophical Transactions of the Royal Society*, vol. A358, pp. 2335–2348, 2000.

[17] S. Deb and R. Srikant, "Rate-based versus Queue-based models of congestion control," in *Proceedings of ACM SIGMETRICS*, 2004.

[18] Y. Yi, S. Deb, and S. Shakkottai, "Time-scale decomposition and rate-based marking," *IEEE/ACM Trasactions on Networking*, vol. 14, no. 5, pp. 938–950, 2006.

[19] G. Raina and D. Wischik, "Buffer sizes for large multiplexers: TCP queueing theory and instability analysis," in *Proceedings of Next Generation Internet Networks*, 2005.

[20] R. Pan, B. Prabhakar, K. Psounis, and D. Wischik, "Shrink: A method for scaleable performance prediction and efficient network simulation," in *Proceedings of IEEE INFOCOM*, San Francisco, CA, 2003.

[21] H. Kim and J. C. Hou, "Network Calculus Based Simulation for TCP Congestion Control: Theorems, Implementation and Evaluation," in *Proceedings of IEEE INFOCOM*, March 2004.

[22] B. L. D. R. Figueiredo, Y. Guo, J. Kurose, and D. Towsley, "A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation," in *Proceedings of infocom*, 2001.

[23] G. Kesidis, A. Singh, D. Cheung, and W. Kwok, "Feasibility of fluid-event-driven simulation for atm networks," in *Proceedings of globecom*, 1996.

[24] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[25] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Proceedings of ACM SIGCOMM*, 2004.

[26] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part iii: Routers with very small buffers," *ACM/SIGCOMM Computer Communication Review*, 2005.

[27] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, May/June 2001.

[28] H. Jiang and C. Dovrolis, "Why is the internet traffic bursty in short time scales?" in *Proceedings of ACM SIGMETRICS*, 2005.

[29] U. M. J. Roberts and J. Virtamo, *Broadband Network Teletraffic, Final Report of Action COST 242*. Boston: Birkhauser, 1992.

[30] C. Villamizar and C. Song, "High performance TCP in ANSNET," *ACM SIGCOMM Computer Communications Review*, vol. 24, no. 5, pp. 45–60, 1994.

[31] J. Cao and K. Ramanan, "A poisson limit for buffer overflow probabilities," in *Proceedings of IEEE INFOCOM*, New York, NY, June 2002.

[32] M. Mandjes and J. H. Kim, "Large deviations for small buffers: an insensitivity result," *Queueing Systems*, vol. 37, pp. 349–362, 2001.

[33] S. Floyd and E. Kohler, "Internet research needs better models," in *HotNets-I*, October 2002.

[34] P. Wood, "A libpcap version whith supports mmap mode," http://public.lanl.gov/cpw/.

[35] D. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes*. New York, NY: Springer-Verlag, 1988.

[36] P. Billingsley, *Convergence of Probability Measures*. Wiley-Interscience, 1999.