The Scaling Hypothesis: Simplifying the Prediction of Network Performance using Scaled-down Simulations

Konstantinos Psounis[†], Rong Pan[†], Balaji Prabhakar[†], Damon Wischik[‡]
[†] Stanford Univerity, [‡] Cambridge University
Email: kpsounis, rong, balaji@stanford.edu, D.J.Wischik@statslab.cam.ac.uk

ABSTRACT

As the Internet grows, so do the complexity and computational requirements of network simulations. This leads either to unrealistic, or to prohibitely expensive simulation experiments.

We explore a way to side-step this problem, by combining simulation with sampling and analysis. Our hypothesis is this: if we take a sample of the traffic, and feed it into a suitably scaled version of the system, we can extrapolate from the performance of the scaled system to that of the original.

We find that when we scale a network which is shared by TCP-like flows, and which is controlled by a variety of active queue management schemes, then performance measures such as queueing delay and the distribution of flow transfer times are left virtually unchanged. Hence, the computational requirements of network simulations and the cost of experiments can decrease dramatically.

1. INTRODUCTION

Measuring the performance of the Internet and predicting its behavior under novel protocols and architectures are important research problems. These problems are made difficult by the sheer size and heterogeneity of the Internet: it is very hard to simulate large networks and to pinpoint aspects of algorithms and protocols relevant to their behavior. This has prompted work on traffic sampling [1, 2]. Sampling certainly reduces the volume of data, although it can be hard to work backwards—to infer the performance of the original system.

A direct way to measure and predict performance is with exhaustive simulation: If we record the primitive inputs to the system, such as session arrival times and flow types, we can in principle compute the full state of the system. Further, through simulation we can test the behavior of the network under new protocols and architectures. But such large-scale simulation requires massive computing power.

Reduced-order models can go some way in reducing the burden of simulation. In some cases [3, 13] one can reduce the dimensionality of the data, for example by working with traffic matrices rather than full traces, while retaining enough information to estimate the state of the network. The trouble is that this requires careful traffic characterization and model-building. The heterogeneity of the Internet makes this time-consuming and difficult, since each scenario might potentially require a different new model.

In this paper we explore a way to reduce the computational requirements of simulations and the cost of experiments, and hence simplify network measurement and performance prediction. We do this by combining simulations with sampling and analysis. Our basic hypothesis, which we call SHRiNK[1], is this: if we take a *sample* of the traffic, and feed it into a *suitably scaled* version of the system, we can *extrapolate* from the performance of the scaled system to that of the original.

This has two benefits. First, by relying only on a sample of the traffic, SHRiNK reduces the amount of data we need to work with. Second, by using samples of actual traffic, it short-cuts the traffic characterization and model-building process while ensuring the relevance of the results.

This approach also presents challenges. At first sight, it appears optimistic. Might not the behavior of a large network with many users and higher link speeds be intrinsically different to that of a smaller network? Somewhat surprisingly we find that, in several essential ways, one can mimic a large network using a suitably scaled-down version. The key is to find suitable ways to scale down the network and extrapolate performance.

The outline of the paper is as follows: In Section 2 we study the scaling behavior of an IP-network whose traffic consists of long-lived TCP-like flows arriving in clusters. Networks with such traffic have been used in the literature to test the behavior of control algorithms and queue management schemes. Using simulations and theory we find that when such a network is suitably scaled, performance measures such as queueing delay and drop probability are left virtually unchanged. In Section 3 we study IP networks at which flows arrive at random times (i.e. unclustered) and whose sizes are heavy-tailed. Such networks are representative of the Internet. We find that a different scaling to that in Section 2 leaves the distribution of the number of active flows and of their normalized transfer times unchanged. A simple theoretical argument reveals that the method we suggest for "SHRiNKing" networks in which flows arrive at random times will be widely applicable (i.e. for a variety of topologies, flow transfer protocols, and queue management schemes). By contrast, we find that the theoretical underpinning for SHRiNKing networks at which flows arrive in clusters depends on the type of queue management scheme

[1]SHRiNK: Small-scale Hi-fidelity Reproduction of Network Kinetics

used at the routers.

A word about the organization of the paper: Space limitations have necessitated a selective presentation of the material. We have chosen to describe the method in detail. The theoretical complement and the validation using simulations are abbreviated. More details can be found in a longer version of the paper [10].

2. SCALING BEHAVIOR OF IP NETWORKS WITH LONG-LIVED FLOWS

In this section we explore how SHRiNK applies to IP networks used by long-lived TCP-like flows that arrive in clusters, and controlled by queue management schemes like RED.

First, we explain in general terms how we sample traffic, scale the network, and extrapolate performance.

Sampling is simple. We sample a proportion α of the flows, independently and without replacement.

We scale the network as follows: link speeds and buffer sizes are multiplied by α. The various AQM-specific parameters are also scaled, as we will explain in the following section 2.1. The network topology is unchanged during scaling. In the cases we study, performance measures such as average queueing delay are virtually the same in the scaled and the unscaled system.

Our main theoretical tool is the recent work on fluid models for TCP networks [8]. While [8] shows these models to be reasonably accurate in most scenarios, the range of their applicability is not yet fully understood. However, in some cases the SHRiNK hypothesis holds even when the fluid model is not accurate, as shown in Section 2.1.2.

2.1 RED

The key features of RED are the following two equations, which together specify the drop (or marking) probability. RED maintains a moving average q_a of the instantaneous queue size q; and q_a is updated whenever a packet arrives, according to the rule

$$q_a := (1-w)q_a + wq,$$

where w is a parameter that determines the size of the averaging window. The average queue size determines the drop probability p, according to the equation

$$p_{\text{RED}}(q_a) = \begin{cases} 0 & \text{if } q_a < min_{th} \\ p_{max}\left(\frac{q_a - min_{th}}{max_{th} - min_{th}}\right) & \text{if } min_{th} \le q_a < max_{th} \\ 1 & \text{if } q_a > max_{th} \end{cases} \quad (1)$$

We scale the parameters p_{max}, min_{th}, max_{th} and w as follows: min_{th} and max_{th} are multiplied by α; p_{max} is fixed at 10%; the averaging parameter w is multiplied by α^{-1}. The reason of choosing these parameters will become clear later in Section 2.1.1.

The Basic Setup

We consider two congested links in tandem, as shown in Figure 1. There are three routers, $R1$, $R2$ and $R3$; and three groups of flows, $grp1$, $grp2$, and $grp3$. The link speeds are 100Mbps and the buffers can hold 8000 packets. The RED parameters are $min_{th} = 1000$, $max_{th} = 2500$ and $w = 0.000005$. For the flows: $grp0$ consists of 1200 TCP flows each having a propagation delay of 150ms, $grp1$ consists of 1200 TCP flows each having a propagation delay of 200ms, and $grp2$ consists of 600 TCP flows each having a propagation delay of 250ms. Note that 75% of $grp0$ flows switch off at time 150s.

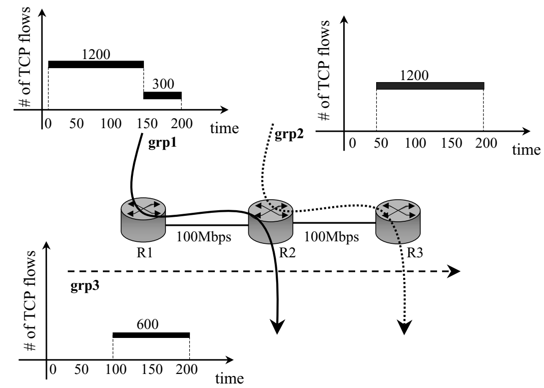

Figure 1: Basic network topology and flow information

This network is scaled-down by factors $\alpha = 0.1$ and 0.02, and the parameters are modified as described above.

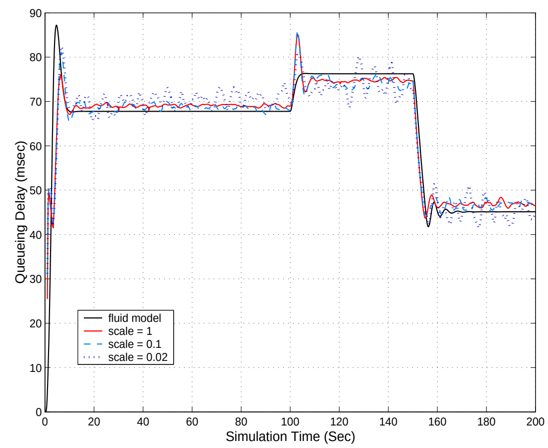

Figure 2: Basic Setup: Average Queueing Delay at Q1

We plot the average queueing delay at Q1 in Figure 2. The drop probability at Q1 is shown in Figure 3. Due to a lack of space, we omit the plot of the average queueing delay and drop probability for Q2 whose behavior is similar to those of Q1. We see that *the queueing delay and the drop probabilities are almost identical at different scales.* We draw attention to two features which we shall comment upon later: (i) The transient behaviors (e.g. the overshoots and undershoots) are quite well-mimicked at the smaller scales, and (ii) the variability increases as the scale reduces.

Since the queueing dynamics and drop probabilities essentially remain the same, the dynamics of the TCP flows are also unchanged. In other words, an individual flow which survives the sampling process essentially cannot tell whether it is in the scaled or the unscaled system.

We have also tested the case where the scale is $\alpha = 0.01$. In this case, only three flows in $grp1$ are present during the period 150s to 200s. Hence the sample is too meager to reproduce the queueing dynamics. Certainly, 1% is not the limit of the scaling hypothesis *in general.* Further study needs to be conducted to find out theoretically where this limit is.

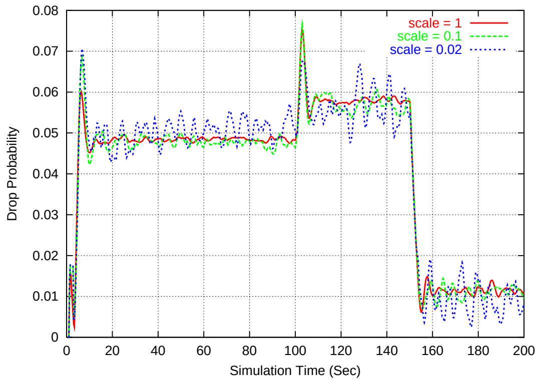

Figure 3: Basic Setup: Drop Probability at Q1

2.1.1 With Faster Links

Suppose we alter the basic setup, by increasing the link speeds to 500Mbps, while keeping all other parameters the same. Figure 4 (zoomed in to emphasize the point) illustrates that, once again, scaling the network does not alter the queueing delay at Q1 (Q2 shows the same scaling behavior). Note that high link speeds cause the queue to oscillate. There have been various proposals for stabilizing RED [5, 9]. We are not concerned with stabilizing RED here: we mention this case to show that SHRiNK can work whether or not the queue oscillates.

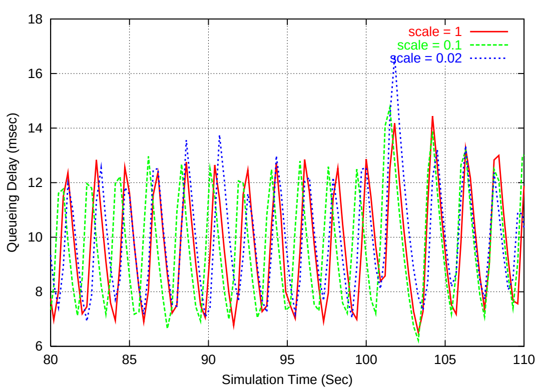

Figure 4: With faster links: Average queueing delay at Q1 (zoomed in)

Theory

We now show that these simulation results are supported by the fluid model of TCP/RED [8].

Consider N flows sharing a link of capacity C. Let $W_i(t)$ and $R_i(t)$ be the window size and round-trip time of flow i at time t. Here $R_i(t) = T_i + q(t)/C$, where T_i is the propagation delay for flow i and $q(t)$ is the queue size at time t. Let $p(t)$ be the drop probability and $q_a(t)$ the average queue size at time t.

The fluid model describes how these quantities evolve; or rather, since these quantities are random, the fluid model describes how their expected values evolve. Let $\bar{X}$ be the expected value of random variable X. Then the fluid model equations are:

$$\frac{d\bar{W}_i(t)}{dt} = \frac{1}{R_i(\bar{q}(t))} - \frac{\bar{W}_i(t)\bar{W}_i(t-\tau_i)}{1.5R_i(\bar{q}(t-\tau_i))}\bar{p}(t-\tau_i) \tag{2}$$

$$\frac{d\bar{q}(t)}{dt} = \sum_{i=1}^{N} \frac{\bar{W}_i(t)}{R_i(\bar{q}(t-\tau_i))} - C \tag{3}$$

$$\frac{d\bar{q}_a(t)}{dt} = \frac{\log(1-w)}{\delta}\bar{q}_a(t) - \frac{\log(1-w)}{\delta}\bar{q}(t) \tag{4}$$

$$\bar{p}(t) = p_{\text{RED}}(\bar{q}_a(t)) \tag{5}$$

where $\tau_i = \tau_i(t)$ solves $\tau_i(t) = R_i(\bar{q}(t-\tau_i(t)))$, δ is the average packet inter-arrival time, and p_{RED} is as in (1) [2]. Suppose we have a solution to these equations

$$\big(\bar{W}_i(\cdot), \bar{q}(\cdot), \bar{q}_a(\cdot), \bar{p}(\cdot)\big).$$

Now, suppose the network is scaled and denote by C', N', etc., the parameters of the scaled system. When the network is scaled, the fluid model equations change, and so the solution changes. Let $\big(\bar{W}'_i(\cdot), \bar{q}'(\cdot), \bar{q}'_a(\cdot), \bar{p}'(\cdot)\big)$ be the solution of the scaled system. It can be theoretically verified (but we do not do this here due to lack of space) that

$$\big(\bar{W}'_i(\cdot), \bar{q}'(\cdot), \bar{q}'_a(\cdot), \bar{p}'(\cdot)\big) = \big(\bar{W}_i(\cdot), \alpha\bar{q}(\cdot), \alpha\bar{q}_a(\cdot), \bar{p}(\cdot)\big),$$

which means the queueing delay $\bar{q}'/C' = \alpha\bar{q}/\alpha C$ is identical to that in the unscaled system. The drop probability is also the same in each case, i.e. $\bar{p}(t) = \bar{p}'(t)$. Thus, we will have theoretical support for the observations in the previous section.

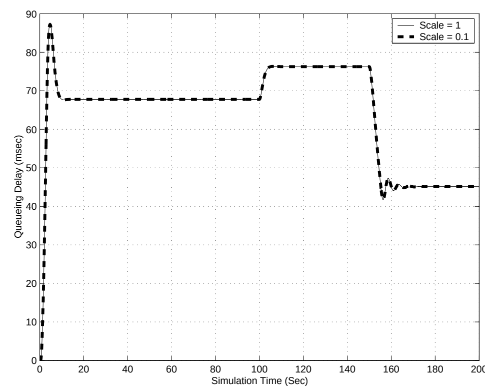

Figure 5: Fluid model predicts scaling behavior

Figure 5 presents the solution of the fluid model for the queueing delay at Q1 under the scenario of Figure 1 for the scale parameters $\alpha = 1$ and 0.1. As can be seen, both the solutions are virtually identical, illustrating the scaling property of the differential equations mentioned above.

2.1.2 When the theory is not appropriate

Suppose we alter the basic setup, by decreasing the link speeds to 50Mbps, while keeping all other parameters the same. Once again, scaling the network does not alter the queueing delay. Due to limitations of space we omit the corresponding plot. For such a simulation scenario, especially in the time frame 100sec-150sec, the fluid model is not a good fit as shown

[2] We have the constant 1.5 in (2), not 2 as in [8]. This change improves the accuracy of the fluid model; due to limited space we omit the derivation.

in [12] and verified by us via simulations: actual window and queue sizes are integer-valued whereas fluid solutions are real-valued; rounding errors are non-negligible when window sizes are small as is the case here. The range of applicability of the fluid model is not our primary concern in this paper: we mention this case to show that SHRiNK can work whether or not the fluid model is appropriate.

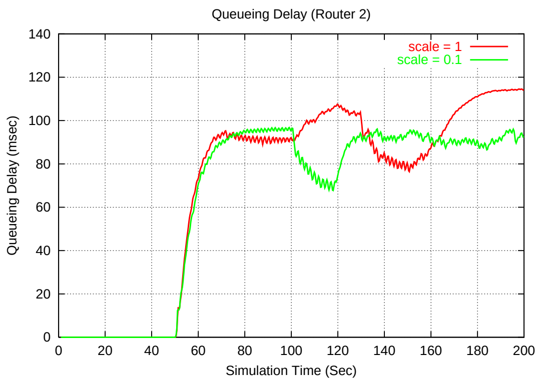

Figure 6: DropTail: Average queueing delay at Q2

2.2 DropTail

While all the simulations above show the validity of SHRiNK, the scaling behavior does not hold when we change the queue management scheme to DropTail. Figure 6 shows the average queueing delay at Q2. Clearly, the queueing delays for different scale do not match. This scheme drops all the packets that arrive at a full buffer. As a result, it could cause a number of consecutive packets to be lost. These bursty drops underlie the reason the scaling hypothesis fails in this case [11]. Besides, when packet drops are bursty and correlated, the assumption that packet drops occur as a Poisson process (see [8]) is violated and the differential equations become invalid.

2.3 Summary

Besides the examples we have studied in this section, we have also validated SHRiNK with heterogeneous end-systems (TCP, general AIMD and MIMD protocols, UDP, HTTP), with a variety of active queue management policies such as the PI controller [6] and AVQ [7], with a range of system parameters, and with a variety of network topologies (tandems, stars, meshes). We have found that, in cases where TCP-like flows are long-lived and drops are not bursty, basic performance measures such as queueing delay are left unchanged, when we sample the input traffic and scale the network parameters in proportion.

3. SCALING BEHAVIOR OF IP NETWORKS WITH SHORT AND LONG FLOWS

It has been shown that the size distribution of flows on the Internet is heavy-tailed [14]. Hence, Internet traffic consists of a large fraction of short and and a small fraction of long flows. It has been observed that sessions arrive as a Poisson process[3]. In this section we take these observations into account and study the scaling behavior of IP networks carrying heavy-tail distributed, Poisson flows. Our finding is that with a somewhat different scaling than in the previous section, the distributions of a large number of performance measures, such as the number of active flows and the delay of flows, remain the same.

3.1 Simulations

We perform simulations using ns-2 for the same topology as in Figure 1. There are three routers, $R1$, $R2$ and $R3$, two links in tandem, and three groups of flows, $grp1$, $grp2$, and $grp3$. The link speeds are 10Mbps. We present simulations with both RED and DropTail. The RED parameters are $min_{th} = 100$, $max_{th} = 250$ and $w = 0.00005$. When using DropTail, the buffer can hold 200 packets.

Within each group flows arrive as a Poisson process with some rate λ. We vary λ to study both uncongested and congested scenarios. (We use the ns-2 built-in routines to generate sessions consisting of a single object each. This is what we call a flow.) Each flow consists of a Pareto-distributed number of packets with average size 12 packets and shape parameter equal to 1.2. The packet size is set to 1000 bytes. The propagation delay of each flow of $grp0$, $grp1$, and $grp2$, is 50ms, 100ms, and 150ms respectively.

Sampling and Scaling

The heavy-tailed nature of the traffic makes sampling a bit more involved than before, because a small number of very large flows has a large impact on congestion. To guarantee that we sample the correct number of these flows, we separate flows into large (elephants) and small (mice) and sample exactly a proportion α of each.

Scaling the system is slightly different from Section 2. As before, we multiply by α the link speeds. However, we do not scale the buffer sizes or the RED thresholds. Further, we multiply by α^{-1} the propagation delay of each flow[4]. We will elaborate on the intuition and theory behind these choices after we present the simulation results.

Since we sample flows which arrive at random times and have random sizes, quantities like the queueing delay cannot be expected to scale as functions of time. However, simulations and theory show that we can exhaustively compare the *distributions* of related quantities.

We run the experiments for scale factors $\alpha = 1$ and 0.1, and compare the distribution of the number of active flows as well as the histogram of the normalized delays of the flows in the original and the scaled system. (The normalized delays are the flow transfer times multiplied by α.) We will also compare more detailed performance measures such as the distribution of active flows that are less than some size and belong to a particular group. Due to limitations of space we will not present results when the links are uncongested, but only compare distributions for the more interesting case where drops occur. (The performance of uncongested networks also scale.) The flow arrival rate is set to be 60 flows/sec within each group. The results don't depend on whether the rates are larger or smaller.

Simulation Results

We will start by comparing distributions when RED is used.

[3]Further, for certain models of TCP bandwidth sharing, the equilibrium distribution of the number of flows in progress is *as if* flows arrive as a Poisson process, not just sessions [4].

[4]One should also multiply by α^{-1} the various protocol timeouts. In practice, since it is very rare for a timeout to expire, leaving timeouts unscaled does not affect the results.

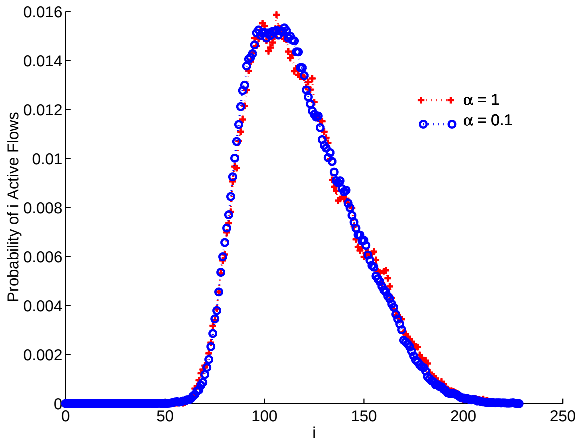

Figure 7: Distribution of number of active flows on the first link.

Figure 7 plots the distribution of the number of active flows in the first link between routers $R1$ and $R2$. It is evident from the plot that the two distributions match. A similar scaling holds for the second link.

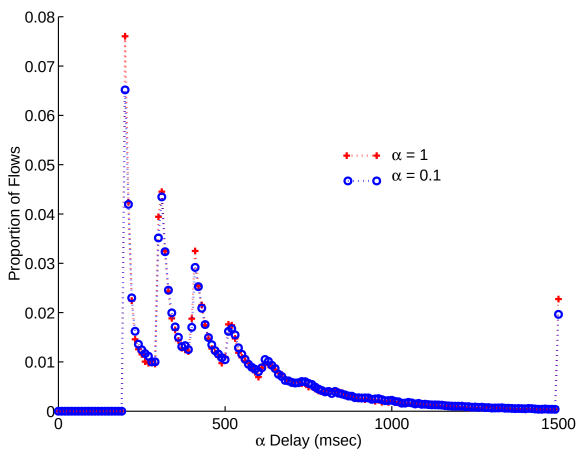

Figure 8: Histogram of normalized delays of *grp*0 flows.

Figure 8 plots the histogram of the flow transfer times (delays) of the flows of *grp*0 multiplied by α. To generate the histogram, we use delay chunks of $\frac{10}{\alpha}$ms each. There are 150 such delay chunks in the plot, corresponding to flows having a delay of 0 to $\frac{10}{\alpha}$ms, $\frac{10}{\alpha}$ms to $\frac{20}{\alpha}$ms, and so on. The last delay chunk is for flows that have a delay of at least $\frac{1500}{\alpha}$ms. It is evident from the plot that the distribution of the normalized delays match. The results for the other two groups of flows are also the same. The peaks in the delay plot are due to the TCP slow-start mechanism. The left-most peak corresponds to flows which send only one packet that face no congestion, the portion of the curve between the first and second peaks corresponds to flows which send only one packet but face congestion (but no drops), the next peak corresponds to flows which send two packets and face no congestion, and so on.

We will now investigate if distributions scale when DropTail is used.

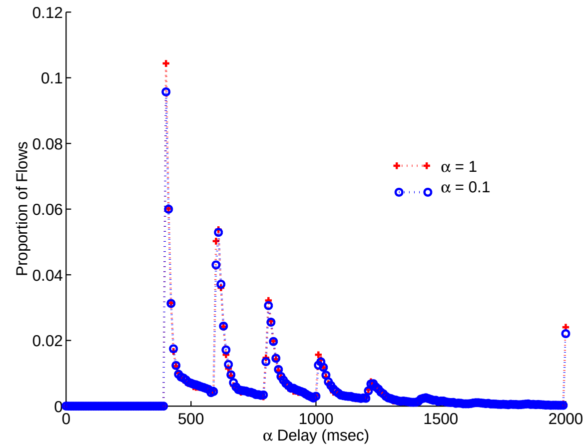

Figure 9: Histogram of normalized delays of *grp*1 flows when DropTail is used.

Figure 9 plots the histogram of the flow transfer times of the flows of *grp*1 multiplied by α, when routers employ DropTail. The distributions of the normalized delays match as before. Although not shown here, the distribution of the number of active flows also scales under DropTail.

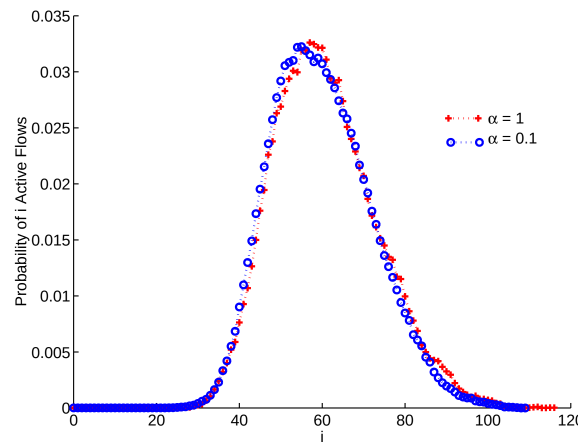

Figure 10: Distribution of number of active *grp*2 flows with size less than 12 packets.

What about more detailed performance measures? As an example, we compare the distribution of active flows belonging to *grp*2 that are less than 12 packets long. The AQM scheme used at the routers is RED (DropTail behaves similarly). Figure 10 compares the two distributions from the original and scaled system. Again, the plots match.

3.2 Theoretical support

The above results can be theoretically supported. First, consider a simplified model: suppose that flows arrive as a Poisson process, and that the service time for each flow is independent and drawn from some common distribution (perhaps heavy-tailed). This is known in queueing theory as an M/GI model.

Suppose also that the service capacity of the link is shared between currently active flows according to the classic equation for TCP throughput. (We will shortly extend the argument to allow for a detailed packet-level model of the link.) This is the sort of flow-level model used in [4].

Let $J(t)$ be the number of jobs in this system at time t. Now scale the process of arriving flows, by multiplying flow interarrival times by $1/\alpha$. Also, multiply the service capacity of the link by α. It is not hard to see that the scaled system looks exactly like the original system, watched in slow motion. Specifically, if $\tilde{J}(t)$ is the number of jobs in the scaled system at time t, then $\tilde{J}(t) = J(\alpha t)$.

Suppose that instead of stretching the flow arrival process in time we had sampled it, retaining each flow independently with probability α. It is a simple but far-reaching property of the Poisson process that these two processes have exactly the same distribution. In particular, if $\hat{J}(t)$ is the number of jobs in the system which is scaled by sampling, then for each t (assuming the queues are in equilibrium), $\hat{J}(t)$ and $\tilde{J}(t)$ have the same distribution, which is the distribution of $J(t)$. That is, the marginal distribution of the number of jobs is the same in the two systems.

This argument does not in fact depend on how the link shares its bandwidth. It could be first-come-first-served, or use priorities. More interestingly, let us instead model the behavior of the link as a discrete event system. Suppose that flows arrive as a Poisson process as before, and that they arrive with a certain number of packets to send, independent and with a common distribution. Consider a time-line showing the evolution in time of the discrete event system. How could we scale the parameters of the system, in order to stretch out the time-line by a factor $1/\alpha$?

To be concrete, suppose that $\alpha = 0.1$, and that a flow with 12 packets takes 1 second to transfer in the original system. We would like to ensure that its transfer time in the scaled system is 10 seconds. We can do this by making sure that each of the 12 packets takes 10 times as long to transfer in the scaled system. Now, the transmission time of a packet is the sum of its queueing delay and propagation delay. We can multiply the queueing delay by 10 by reducing the link speed by a factor of 10; we should also multiply the propagation delay by 10.

In general, we should multiply propagation times by $1/\alpha$, and the service times by the same factor, which means multiplying the service rate by a factor α. As before we would multiply flow interarrival times by $1/\alpha$, which has the same effect as sampling with probability α. Note that this model takes account of retransmissions due to drops (assuming either there are no timeouts, or that the timeout clock is also scaled). Note also that the packet buffer at the link is *not* scaled. The conclusion holds just as before: the marginal distribution of the number of jobs in the system is unchanged.

4. CONCLUSION

In this paper we have presented a method, SHRiNK, to reduce the complexity of network simulations and performance prediction. Our main finding is that when a sample of the network traffic is fed to a suitably scaled replica of the network, performance measures of the original network are accurately predicted by the smaller scale replica. In particular, (i) when long-lived flows arrive in clusters, queueing delays and drop probabilities in the two networks are the same as a function of time in many interesting scenarios, and (ii) when flows arrive at random times and their size is heavy-tailed, the distribution of performance measures under *any* network topology, active queue management mechanism, and transport protocol remains unchanged. We have shown these results using simulations and theory.

Further work consists of validating SHRiNK in large experimental testbeds, obtaining a better understanding of the theory, and trying to extend the approach to web-server farms and to wireless networks.

5. REFERENCES

[1] K. Claffy, G. Polyzos, and H.-W. Braun. Applications of sampling methodologies to network traffic characterization. In *Proceedings of SIGCOMM*, 1993.

[2] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, 2001.

[3] Fluid models for large, heterogeneous networks. `http://www-net.cs.umass.edu/fluid/`, accessed January 2002.

[4] S. Ben Fredj, T. Bonalds, A. Prutiere, G. Gegnie, and J. Roberts. Statistical bandwidth sharing: a study of congestion at flow level. In *Proceedings of SIGCOMM*, 2001.

[5] C.V. Hollot, V. Misra, D. Towlsey, and W. Gong. A control theoretic analysis of RED. In *Proceedings of INFOCOM*, 2001.

[6] C.V. Hollot, V. Misra, D. Towlsey, and W. Gong. On designing improved controllers for AQM routers supporting TCP flow. In *Proceedings of INFOCOM*, 2001.

[7] S. Kunniyur and R. Srikant. Analysis and design of an Adaptive Virtual Queue (AVQ) algorithm for active queue management. In *Proceedings of SIGCOMM*, 2001.

[8] V. Misra, W. Gong, and D. Towsley. A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proceedings of SIGCOMM*, 2000.

[9] T. Ott, T. Lakshman, and L. Wong. SRED: Stabilized RED. In *Proceedings of INFOCOM*, 1999.

[10] R. Pan, B. Prabhakar, K. Psounis, and D. Wischik. Shrink: A method for scalable performance prediction and efficient network simulation. `http://www.stanford.edu/~kpsounis/scale1.html`, accessed October 2002.

[11] R. Pan, K. Psounis, B. Prabhakar, and M. Sharma. A study of the applicability of the scaling-hypothesis. In *Proceedings of ASCC*, 2002.

[12] S. Shakkottai and R. Srikant. How good are deterministic fluid models of internet congestion control. In *Proceedings of Infocom 2002, to appear*, 2002.

[13] J. Walrand. A transaction-level tool for predicting TCP performance and for network engineering. In *MASCOTS*, 2000. `http://walrandpc.eecs.berkeley.edu/Papers/mascots1.pdf`.

[14] W. Willinger, M.S. Taqqu, R. Sherman, and D.V. Wilson. Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, 1997.