# Performance preserving topological downscaling of Internet-like networks

Fragkiskos Papadopoulos, Konstantinos Psounis, Ramesh Govindan
University of Southern California
E-mail: fpapadop, kpsounis, ramesh@usc.edu.

***Abstract*— The Internet is a large, heterogeneous system operating at very high speeds and consisting of a large number of users. Researchers use a suite of tools and techniques in order to understand the performance of complex networks like the Internet: measurements, simulations, and deployments on small to medium-scale testbeds. This work considers a novel addition to this suite: a class of methods to *scale down* the *topology* of the Internet that enables researchers to create and observe a smaller replica, and extrapolate its performance to the expected performance of the larger Internet. This is complementary to the work in [1], where the authors presented a way to scale down the Internet in *time*, by creating a slower replica of the original system.**

**The key insight that we leverage in this work is that only the congested links along the path of each flow introduce sizable queueing delays and dependencies among flows. Hence, one might hope that the network properties can be captured by a topology that consists of the congested links only. Using extensive simulations with TCP traffic and theoretical analysis, we show that it is possible to achieve this kind of performance scaling even on topologies the size of the CENIC backbone (that provides Internet access to higher-education institutions in California). We also show that simulating a scaled topology can be up to two orders of magnitude faster than simulating the original topology.**

Topology downscaling, Performance prediction, Efficient network simulation, TCP/closed-loop networks.

## I. Introduction

Networking research has taken a multi-pronged approach to understanding the performance of the Internet, and to predicting its behavior under new algorithms, protocols, architectures and load conditions. The community focuses on techniques ranging from analytic modeling [2], [3], [4], [5], [6], [7], to measurement-based performance characterizations [8], [9], [10], [11], [12] to simulation studies [13], [14], [15], [16]. This is appropriate given the overwhelming size, complexity, heterogeneity, and the speed of operation of the Internet.

This multi-pronged approach has its limitations. First, the heterogeneity and complexity of the Internet makes it very difficult and time consuming to devise realistic traffic models, and models for the network. Second, for some of the same reasons, as well as the increasingly large bandwidths in the Internet core, it is very hard to obtain accurate and representative measurements. Even when such data are available, it is very expensive to run realistic simulations at meaningful scales since the memory and CPU requirements of such simulations seem to be well beyond the reach of available hardware. Of course, there are several approaches to alleviate some of these problems. The volume of measurements can be reduced by traffic sampling, and researchers attempt to use "realistic" topology models. However, topology modeling is still in its infancy—realistic models that include notions of capacity and latency are some years away. Further, from sampled traffic, it is hard to infer the performance of the original system.

The focus of this work and of its predecessor [1], [17], [18] is the addition of a new prong—a class of *performance-preserving network downscaling* techniques that lets a designer study the behavior of new services or mechanisms in a large network using a scaled-down version of the network, where the downscaling is *designed to preserve one or more aspects of the original network*. Studying engineering artifacts by scaling them down (in addition to simulating them) has long been an established tradition in other disciplines. For example, civil engineers not only study finite-element models of large structures, they also build scale models that attempt to faithfully replicate the proportional loading on various structural elements. We attempt to bring this methodology to computer networking.

But what does it mean to design a performance-preserving network downscaling technique? Consider a large network (such as that of a backbone ISP), consisting of several hundred nodes and links, and traversed by hundreds of thousands of flows. Psounis et al. [1] have introduced a method called SHRiNK [1] that preserves some network properties by creating a *slower* downscaled version of the original network. Specifically, SHRiNK downscales link capacities (*but not network size*) such that, when a sample of the original set of flows is run on the downscaled network, a variety of performance metrics, *e.g.* the packet delay distributions, are preserved.

But is there more than one instance of a performance-preserving downscaling technique? Yes. In this work, we propose two methods to perform *topological* downscaling. In particular, starting from a complex network, we create a smaller version of it with fewer links and nodes, observe the behavior of a sample of the original traffic on this network, and extrapolate the observed performance back to the original network.

Topological downscaling has three benefits. First, by relying only on a sample of the traffic, it reduces the amount of data we need to work with. Second, by using actual traffic, it shortcuts the traffic characterization and model-building process. Finally, by reducing the number of links and nodes, it reduces the complexity of the network that we work with. This, in turn, expedites simulations, allows researchers to test new

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

architectures and algorithms in small experimental testbeds while ensuring the relevance of the results, and can help operators manage existing networks more efficiently.

This approach also presents challenges. At first sight, it appears optimistic. For example, it is possible that a group of flows share some links in the original network but not in the replica. Hence, the replica may not capture some of the correlations between these flows. However, a more careful look at the network reveals that it is only the congested links along the path of each flow that introduce dependencies among flows and sizeable queueing delays [3], [19], [9], [20], [8], [21]. Further, it has been recently shown that links with capacities large enough to carry many flows without getting congested, are in a sense, transparent to the flows that pass through them [22], [23], [24], [25]. Hence, one might hope that the network properties can be captured by a topology that consists of the congested links only. The very small number of congested links along the path of a flow makes this approach quite effective in reducing the size of the network that one works with.

We have applied our approach in the topology of the CENIC backbone [26] and have successfully predicted queue and flow statistics of the original network from the small replica with very high accuracy. The accuracy of the approach does not depend on the load conditions, the active queue management schemes used, the existence of two-way traffic, etc. The approach can be automated, that is, given the original topology and traffic, a tool can easily create the scaled network, run simulations, and extrapolate the performance of the original network from the simulation results. And simulating the scaled topology can be up to two orders of magnitude faster than simulating the original topology.

The rest of this paper is organized as follows: Section II briefly discusses prior work on scaling down networks. Section III gives a formal definition for congested links and discusses the impact of congested and uncongested parts of a network to performance. Section IV introduces the proposed methods and applies them to simple topologies, to realistic topologies where multiple congested links exist (CENIC backbone [26]), and to situations where two-way traffic is present. Extensive simulation results are presented in Section V. It is shown that a variety of metrics, including packet queueing delays, end-to-end flow delays, number of active flows, ACK delays, etc. can be predicted from the replica. Formal proofs establishing the validity of the methods are presented in Section VI. Finally, Section VII presents the savings - in terms of the time needed to run an experiment - from the usage of these methods, and Section VIII concludes the paper.

## II. Related Work

Psounis et al [1] introduced a method called SHRiNK that creates a *slower* version of the original network. The main steps in the creation of the replica are to reduce link capacities, increase propagation delays, and reduce the traffic arrival rate by sampling incoming flows. The main results of this work are the following [1], [17], [18], [27]: (i) For networks in which flows arrive at random times and whose sizes are heavy-tailed, performance measures such as the distribution of the number of active flows and of their normalized transfer times are left virtually unchanged. This is verified using extensive simulations and a simple but powerful theoretical argument. These networks are representative of the Internet. (ii) For networks which carry long-lived TCP-like flows arriving in clusters, and which are controlled by a variety of active queue management schemes, a slightly different scaling to the previous one leaves the queueing delay and drop probability unchanged as a function of time. This is verified using simulations and the differential-equation type models developed in [28]. (Such models have been widely used in designing control algorithms and for conducting control-theoretic analysis of network behavior.) These networks are widely used in simulations, e.g. [29], [30], [31]. As mentioned before, this class of work has *not* considered downscaling the size of the topology.

There have been very few studies of the question of whether one can reduce the topology of a network without losing important network properties. The most relevant to our work is the one by Eun et al. [22], [23], [24], [25]. In this line of work, the authors show analytically that in a tandem of queues where arrivals are “fluid-like” or are dictated by Poisson-point-processes, as the capacity of upstream queues and the number of flows through them increases, the performance of a downstream queue is about the same, as if all the upstream queues did not exist. However, this work consideres the network as being an “open-loop” system. In our work, we want to investigate if network downscaling is possible in the Internet, where traffic is dictated by TCP-like mechanisms, that is, we are studying “closed-loop” systems.

Other studies, e.g. [32], have used graph properties as metrics to judge the quality of network topological down-scaling. In particular, they have used metrics like the average degree of a graph, the clustering coefficient [33], or the degree exponent [34]. Our approach towards scaling down the topology of a network is very different. The goal is to be able to predict the performance of the original network from the scaled network. Like Psounis et al. [1], we want to be able to predict flow transfer times, packet delays, queue sizes, and so on.

Another line of research that is relevant to our work attempts to replace time consuming packet-level simulations by modelling. Bohacek et al. [35] propose a hybrid modelling framework that uses averaging of discrete variables over short time intervals, and can accurately predict network behavior faster than packet-level simulations. Liu et al. [2] extend the fluid models introduced in [28] to be topology-aware, take into account congested links only, and accurately predict network dynamics by solving the fluid models. They show that this takes less time than packet-level simulations, especially when workloads and bandwidths are high. Our work does not attempt to replace packet-level simulations with faster methods, but rather to run them in smaller networks with fewer traffic, which results in faster execution times.

## III. Congested Links: Definition and Impact

The widely used term “congested link” has various meanings in the literature today, see, for example, [10], [36], [37].

Typically, it is defined to be a link with either high utilization, low available bandwidth, high loss rate, long queueing delay, or a combination thereof. [2] These definitions are not necessarily equivalent. For example, a link connected to a small buffer may have a high loss rate but low queueing delay. Further, queueing delays on links with the same utilization may differ significantly, if the link capacities are different. Finally, the link with the smallest available (or residual) bandwidth along a path, only determines the end-to-end throughput of that path. This doesn't necessarily mean that the link is highly utilized or that the link queueing delays are large.

A common property of most types of "congested" links is that they alter the packet arrival process as packets go through the link. For example, this is clearly the case when drops occur, or when queueing delays are long and change over time. Taking into consideration our goal to create small, performance-preserving network replicas by ignoring "uncongested" links, it is natural to define the latter as precisely those links that do not alter the packet arrival process. For simulation purposes, we consider a link to be "uncongested" if no drops occur and the average queueing delay incurred to a packet of any source-destination pair by this link is one order of magnitude smaller than the total end-to-end queueing delay of the packet. [3]

With a specific definition at hand, we want to assess what is the impact of congested and uncongested links on performance. In this work we argue that ignoring uncongested links does not result in a loss of performance-related information about the original network. [4] This is based on the following two observations: (i) the packet arrival process is almost the same before and after an uncongested link, and (ii) queueing delays due to uncongested links do not have a significant contribution to end-to-end packet delays. These observations are used in [21], where a scheme was proposed for computing the end-to-end queueing delay through a backbone network, and are validated in detail in [20]. These are also the observations made in [3], where a method for modelling the backbone traffic at the flow level was introduced. Similar results have been derived in [5] for traffic which observes the Large Deviations Principle, and were used in [38] to justify that the effective bandwidth of a flow remains unchanged throughout a network.

Another line of work in support of our claims is [22], [23], [24], [25]. As briefly explained in the previous section, in this work the authors show analytically that when arrivals are "fluid-like" or are dictated by Poisson-point-processes, if the internal links of a network (or upstream queues) have large enough capacities and are serving (multiplexing) many flows, it is then safe to remove those links from consideration, and the queueing behavior of the other network links (or downstream queues) remains unchanged. In the same study, simulation results suggest that an upstream queue does not significantly contribute to the end-to-end queueing delay of a packet, and that a similar behavior is observed when packet arrival times are dictated in an open-loop manner by real ethernet traces.

**Remark:** In a network where paths are very long, a large number of uncongested links might have collectively a significant impact on the total end-to-end queueing delay of a packet. We ignore this situation based on the well-known fact that Internet paths are quite short; they are typically less than fourteen hops long, and rarely more than eighteen hops long [39], [40].

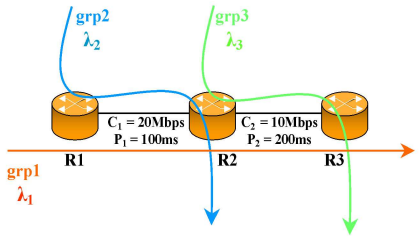



Fig. 1. Original network topology.

## IV. Scaling down network topology

In this section we present two methods for scaling down the topology of a network while preserving its performance. Our main assumption is that ignoring uncongested links does not result in a loss of information about the performance of the original network.

The proposed methods operate on a given topology with known traffic. (There is a large number of efficient tools that can estimate source/destination pairs, the corresponding rates, the paths followed by network flows, and the links that are congested, see, for example, [41], [42].) Under both methods, the downscaled network consists of congested links from the original topology. The first method, called DSCALEd (downscale using delays), accounts for the missing uncongested links by adding appropriate fixed delays to all packets. The second method, called DSCALEs (downscale using sampling), accounts for the missing links by sampling flows and properly adjusting the capacities and propagation delays of the scaled (or replica) network.

We now describe a simple topology used to introduce the methods. Consider two 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*. A group of flows comprises all the flows that follow the same path from the source to the destination. [5] The link *R1-R2* has speed $C_1 = 20$Mbps and propagation delay $P_1 = 100$ms. The link *R2-R3* has speed $C_2 = 10$Mbps and propagation delay $P_2 = 200$ms. Within each group, flows arrive with some rate $\lambda_i$, $i = 1, 2, 3$. We vary $\lambda_i$'s so that the link *R2-R3* becomes congested whereas the link *R1-R2* remains uncongested. (For simulation purposes we asume the following. Buffers on the routers can hold 300 packets, they use either DropTail or RED, and the RED's parameters are $min_{th} = 100$, $max_{th} = 250$ and averaging parameter $w = 0.00005$.)

[2]Every link is fed with packets from a buffer, e.g. the buffer of a router linecard. The properties of this buffer, e.g. queueing delay, are also considered properties of the corresponding link.

[3]Note that if two packets belong to different flows that follow different paths, it might be the case that the delay due to a "congested" link comprises a large proportion of the end-to-end delay for one but not for the other packet.

[4]Clearly, one loses information about the network topology, but as we shall see, this does not hinder ones ability to preserve the performance-related information of the original network.

[5]Notice that in accordance with the usual practice [43], [44], [8], packets are said to belong to the same flow if they have the same source and destination IP address, and source and destination port number. A flow is "on" if its packets arrive more frequently than a timeout of some seconds. This timeout is usually set to something less than 60 seconds.

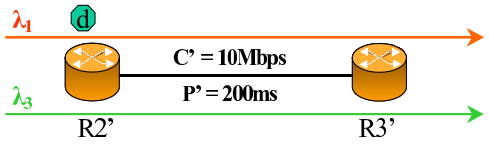



Fig. 2. Scaled system when using DSCALEd.

Given the network setup shown in Figure 1, we are interested in creating a replica and predicting various performance measures on link *R2-R3*, which is the bottleneck link. A detailed description of the proposed methods that accomplish this task follows.

### A. DSCALEd: Downscaling using delays

The first method is quite intuitive and can be summarized as follows:

1) Ignore uncongested links and retain all congested links.
2) Groups of flows that traverse congested links in the original network, will traverse the corresponding bottleneck links in the scaled system.
3) Groups of flows that do not traverse bottleneck links are ignored.
4) Assign to the links of the scaled system, speed and propagation delay values equal to those they have in the original network.
5) For every group of flows that had in their original paths at least one link that is not included in the scaled system, add a constant delay factor such that the end-to-end propagation and transmission delays for each group of flows is equal to that in the original system.

Let's apply the method to the network in Figure 1. Since link *R1-R2* is highly uncongested and no drops occur, there isn't any interdependence between *grp1* and *grp2* flows. Moreover, the rate by which *grp1* packets enter the link *R2-R3* is not affected by the queueing on link *R1-R2*. Finally, the only link that has an impact on the total queueing delay of *grp1* flows is the congested link *R2-R3*. Having the above comments in mind, we remove link *R1-R2* from the original topology, and consequently, ignore *grp2* flows. Hence, the scaled system will consist of two groups of flows (*grp1*, *grp3*) and one link, denoted by *R2'-R3'*, as shown in Figure 2.

To compensate for the absence of link *R1-R2* from the scaled system, a fixed delay $d$ is introduced to each *grp1* packet. In this example, the value of $d$ equals the sum of the propagation and transmission delay of link *R1-R2*. That is, $d = P_1 + L/C_1 = 100.416$ms, where $L = 1040$bytes is the packet size. Notice that a fixed delay like this can be very easily added to the round trip time of a packet by a simulator.
**Remarks:** When the original topology and the number of source/destination pairs are large, adding the fixed delays at exactly the point where uncongested network segments existed in the original network may require significant bookkeeping. It would simplify simulations if one could add the fixed delays at the source of each packet, irrespectively of where along its original path these delays occur. This is the approach that we follow in the simulations. Since this approach causes different time-shifts to packets that share a link but have different end-to-end paths, it is not clear whether the approach preserves the performance of the original network. In Section VI we prove that this is the case.

Another subtle point is the following: While performance preservation is intuitive under the assumption that uncongested links are transparent to packets, it still remains to be seen if, in practice, removing a large number of uncongested links won't change the dynamics of the network.

Finally, it is easy to use this method in conjunction with scaling in time [1], [17] to produce a smaller and slower network replica. In particular, once the smaller replica is produced, one can employ scaling in time independently. Scaling in time has the benefit of reducing the amount of traffic going though the network (via traffic sampling), which further expedites simulations.

### B. DSCALEs: Downscaling using sampling

It is interesting to investigate if one can seamlessly combine space and time downscaling in a single method. The resulting scaled network would consist of fewer links, and due to traffic sampling associated with time downscaling, would be traversed by fewer packets per unit of time. This, in turn, would result in faster simulations as compared to plain DSCALEd that does not use scaling in time. Another reason to look for more methods to downscale a network is to investigate if it is possible to avoid adding a fixed delay to each packet, which is a requirement of DSCALEd.

With the above in mind, we propose a method that preserves performance by sampling flows and carefully choosing the capacities and propagation delays of the links of the replica network. The method can be summarized as follows:

1) Ignore uncongested links and retain all congested links.
2) Groups of flows that traverse congested links in the original network, will traverse the corresponding bottleneck links in the scaled system.
3) Groups of flows that do not traverse bottleneck links are ignored.
4) Sample each group of flows with different probabilities. (details described below)
5) Compute the capacities and propagation delays of the links of the scaled network such that (i) the round trip times of each group of flows remain unchanged (except by a constant multiplicative factor), and (ii) the traffic intensities of the links in the original and scaled network are equal.

Before describing the method in detail, we make the assumption that the *total end-to-end* queueing (and tranmission) delay is negligible in comparison to the *total* end-to-end delay. This assumption is primarily made for ease of exposition. DSCALEs can accurately predict a large number of metrics, but not all, even when queueing delays are large, as we show via theory and simulations in Section VI and V respectively. Nevertheless, it is interesting to point out that the negligible queueing delay assumption is not unrealistic for IP backbone networks, see, for example, [19]. In particular, in [9] it is reported that the average measured queueing delay on an operational OC-3 link for two data sets was as low as 70$\mu$s

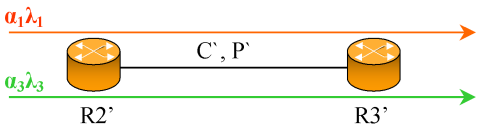



Fig. 3. Scaled system when using DSCALEs.

and 33$\mu$s respectively. Further, in [20], [8], [21] it is observed that packets not only experience insignificant queueing inside a backbone, but also that the jitter is insignificant and that the link utilizations can reach $80\% - 90\%$ before queueing delays begin to exceed several ms. Also, in [21], it is shown that a minor excess bandwidth is needed to support end-to-end queueing delay requirements as low as 4ms. Hence, although a packet may experience significant queueing in some links compared to others (e.g. at network access points v.s. links at the core), it's total end-to-end delay is mainly dominated by the propagation delay.

Now, let's describe the method in detail. In the original topology (Figure 1), the link *R2-R3* is traversed by two groups of flows: *grp1* and *grp3*. A packet of a flow that belongs to *grp1* will experience a (one-way) average end-to-end latency, equal to the sum of its queueing, transmission, and propagation delays. Thus, the total average latency from the source to the destination of a *grp1* packet equals

$$\frac{q_1 L}{C_1} + \frac{L}{C_1} + P_1 + \frac{q_2 L}{C_2} + \frac{L}{C_2} + P_2, \tag{1}$$

where $q_1$ and $q_2$ are the average queue sizes (measured in number of packets, not counting the packet being transmitted) for the queues *R1-R2* and *R2-R3* respectively, and $L$ is the packet size. Similarly, the end-to-end average latency of a *grp3* packet equals $\frac{q_2 L}{C_2} + \frac{L}{C_2} + P_2$. Assuming that the queueing and transmission delays are negligible in comparison to the propagation delays, the round-trip time for *grp1* flows is approximately equal to:

$$RTT_1 = 2\left(P_1 + P_2\right). \tag{2}$$

Similarly, for *grp3* flows we have:

$$RTT_3 = 2P_2. \tag{3}$$

For the same reasons as before, we remove link *R1-R2* from the original topology, and consequently, ignore *grp2* flows. The scaled system will consist of two groups of flows (*grp1*, *grp3*) and one link, denoted by *R2'-R3'*. Let $C'$ be the capacity and $P'$ be the propagation delay of this link.

On the scaled system (shown in Figure 3), the average one-way end-to-end latency of a packet belonging to either *grp1* or *grp3* equals $\frac{q' L}{C'} + \frac{L}{C'} + P'$, where $q'$ is the average queue size on the scaled system. Assuming that the queueing and transmission delays are negligible (comparing to the end-to-end delays) in the scaled system as well, the round-trip time on this system is $RTT' = 2P'$.

Notice that the round-trip time on the scaled system is the same for both groups of flows. To capture the delay dynamics of both groups, we introduce two constants, $a_1$ and $a_3$, and require that the following two equations hold simultaneously:

$$\left(P_1 + P_2\right) = a_1 P', \tag{4}$$

$$P_2 = a_3 P'. \tag{5}$$

In other words, the round-trip time of *grp1* flows is scaled by a factor of $\frac{1}{a_1}$ and the round-trip time of *grp3* of flows is scaled by a factor of $\frac{1}{a_3}$.

This is reminiscent of the situation in [1], [17], where the authors prove the following downscaling law: If network flows are sampled with some factor $\alpha$ and fed into a network replica whose link speeds are multiplied by $\alpha$ and propagation delays by $1/\alpha$, then performance extrapolation is possible. The main point of the law is to reduce the arrival rate and hence increase interarrival times via sampling by some amount, and slow down the network such that the round-trip time for every flow is increased by the same amount.

The difference here is that the round-trip times are scaled with different factors depending on the group of flows. Hence, we sample flows that belong to *grp1* with a factor $\alpha_1$ and flows that belong to *grp3* with a factor $\alpha_3$, as shown in Figure 3. Note that sampling only dictates whether a particular flow is present at the replica network or not. Once a flow is sampled, its packets traverse the network according to the TCP and network dynamics of the original or scaled system.

Let $a_1 + a_3 = \delta$ for some constant $\delta$ that we choose such that $a_1 \leq 1$ and $a_3 \leq 1$. (The specific value of this constant is not important.) Then, this equation together with Equations (4) and (5) can be solved to find $a_1$, $a_3$ and $P'$.

To find $C'$ we require the traffic intensity in the congested link under study to be the same in the original and the scaled network. The traffic intensity is proportional to the ratio of the flow arrival rate over the link capacity. The total arrival rate of flows on link *R2-R3* is $\lambda_1 + \lambda_3$. Due to sampling, the total arrival rate of flows on the scaled system is $\alpha_1\lambda_1 + \alpha_3\lambda_3$. Hence, for traffic intensities to match,

$$C' = s \cdot C_2, \quad \text{where} \quad s = \frac{\alpha_1\lambda_1 + \alpha_3\lambda_3}{\lambda_1 + \lambda_3}. \tag{6}$$

Applying the method with $\delta = 1$ we get $\alpha_1 = 0.6$, $\alpha_3 = 0.4$, $C' = 5$Mbps ($s = 0.5$) and $P' = 500$ms.

**Remarks:** It is straightforward to study uncongested links using either of the methods. To do so, one only needs to add to the scaled system the uncongested links of interest, together with the groups of flows that traverse them.

Further, the methods do not depend on the potential location of the congested links. These can be anywhere, i.e inside the core, at public exchange points, at the edges of the network, etc. It is also possible that the location of the congested links changes over time, e.g. due to changes in load conditions. In such situations our method would merely need to be reapplied to the altered topology and traffic matrix.

### C. Multiple bottlenecks

To demonstrate the applicability of our methods to larger topologies with multiple bottlenecks, we consider the CENIC backbone [26]. The topology of the CENIC backbone along with link information is shown in Figure 4.

The CENIC maps do not include information about the propagation delays of the links and the traffic that traverses them. We estimate the propagation delay of a link by dividing the length of the link over the propagation velocity of the

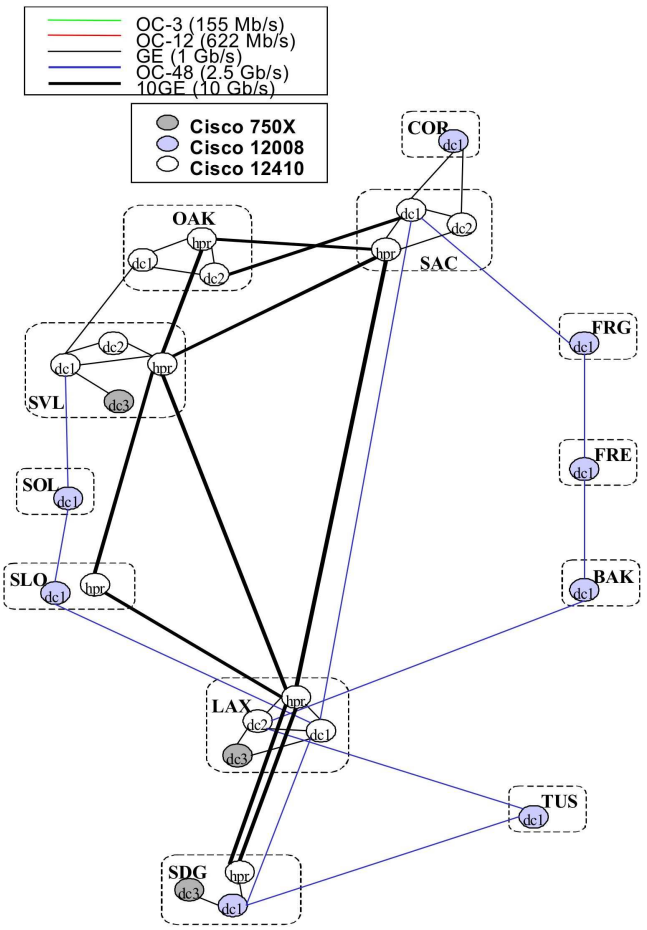



Fig. 4. The CENIC Backbone.

| Link | Propagation Delay(ms) |
|---|---|
| *SDG(hpr)-LAX(hpr)* | 1.0 |
| *SDG(dc1)-TUS(dc1)* | 0.7 |
| *SDG(dc1)-LAX(dc1)* | 1.0 |
| *LAX(dc2)-TUS(dc1)* | 0.3 |
| *LAX(dc2)-BAK(dc1)* | 1.0 |
| *BAK(dc1)-FRE(dc1)* | 1.0 |
| *FRE(dc1)-FRG(dc1)* | 0.5 |
| *FRG(dc1)-SAC(dc1)* | 1.0 |
| *SAC(dc1)-LAX(dc1)* | 3.0 |
| *SAC(hpr)-LAX(hpr)* | 3.0 |
| *SAC(hpr)-SVL(hpr)* | 1.0 |
| *SAC(hpr)-OAK(hpr)* | 0.6 |
| *SAC(dc1)-OAK(dc2)* | 0.6 |
| *LAX(hpr)-SVL(hpr)* | 2.8 |
| *LAX(hpr)-SLO(hpr)* | 1.6 |
| *OAK(hpr)-SVL(hpr)* | 0.3 |
| *OAK(dc1)-SVL(dc1)* | 0.3 |
| *SVL(dc1)-SOL(dc1)* | 0.7 |
| *SOL(dc1)-SLO(dc1)* | 0.8 |
| *SLO(dc1)-LAX(dc1)* | 1.6 |
| *SLO(hpr)-SVL(hpr)* | 1.5 |

TABLE I
LINK PROPAGATION DELAY

signal (taken as 133000 miles/sec). The propagation delay for all the links that belong to the same geographic area is taken as 0.1ms. For the rest of the links the propagation delays are shown in Table I.

We let each possible source-destination pair in the topology to correspond to a group of flows. Hence, in total there are 600 groups of flows. The input traffic that we impose forces links *SVL(hpr)-SVL(dc1)* and *LAX(dc2)-LAX(hpr)* to be congested. We are interested in studying the link *SVL(hpr)-SVL(dc1)*, which is traversed in both directions by a total of 70 groups of flows (35 per direction of the link). Since some of the flows that traverse the congested link of interest, link *SVL(hpr)-SVL(dc1)*, also traverse the other congested link, link *LAX(dc2)-LAX(hpr)*, the scaled replica should consist of both links. Otherwise, the scaled topology will not capture the effect that the congested link *LAX(dc2)-LAX(hpr)* has on the flows that go through it, and performance prediction will be inaccurate. Flows within each group have exactly the same characteristics as in the simple topology. (For simulation purposes we assume that the routers can hold 400 packets, and they use either DropTail or RED with parameters $min_{th} = 150$, $max_{th} = 350$ and averaging parameter $w = 0.00005$.)

Using DSCALEd, we can easily construct the scaled system shown in Figure 5. [6] Each arrow represents all groups of flows that traverse the corresponding link in the original topology in the direction of the arrow. Note that the figure shows only the left-to-right direction of the traffic. The two links are being traversed in the opposite direction in exactly the same way. In the scaled topology there are a total of 120 groups of flows. An appropriate fixed delay value is added to the round-trip time of all the packets of each group, as explained before. For example, let *grp1* be the group of flows which in the original topology follows the path: *SDG(dc3)-SDG(dc1)-SDG(hpr)-LAX(hpr)-SVL(hpr)-SVL(dc1)-SVL(dc3)*. In the scaled topology this group traverses link *SVL(hpr)-SVL(dc1)* only, and the fixed delay added to all *grp1* packets is $\sum_i P_i + L/C_i = 4.127$ms, where the sum is over all links $i$ along the path of the group in the original system that are not present in the scaled system. For another example, let *grp2* be another group of flows which in the original topology follows the path: *SVL(dc3)-SVL(dc1)-SVL(hpr)-LAX(hpr)-LAX(dc2)-TUS(dc1)*. In the scaled topology, *grp2* traverses links *SVL(dc1)-SVL(hpr)* and *LAX(hpr)-LAX(dc2)*, and the fixed delay added to all *grp2* packets is 3.213ms. Continuing this way we can compute the fixed delay for all the groups of flows.

Using DSCALEs, we can construct the scaled system shown in Figure 6. The arrows show the direction of the traffic as before. Looking at Figure 6, let $C1'$ be the capacity of link *SVL(dc1)-SVL(hpr)*, $C2'$ be the capacity of link *LAX(hpr)-LAX(dc2)*, and $P'$ be the propagation delay of both links.

Following the same methodology as in the simple topology scenario, we can write the following equations for *grp1* and

[6] This process can be automated. We have tools, available upon request, that automatically perform most of the required tasks, and we plan to create a software tool that puts everything together.

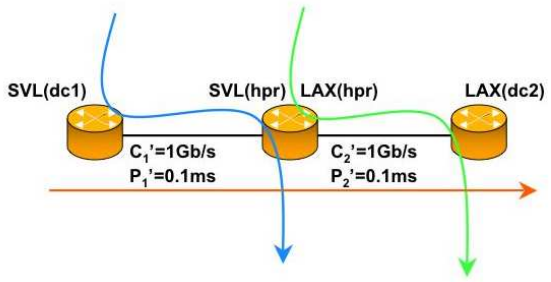



Fig. 5. Scaled system with link and flow information (DSCALEd).

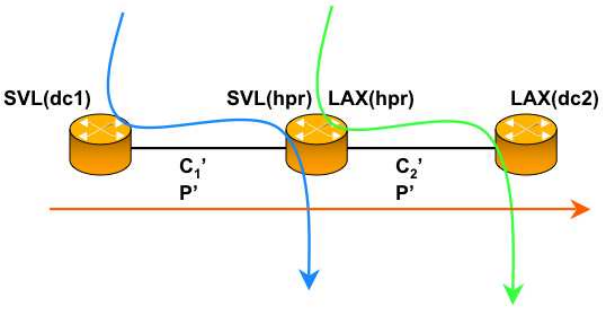



Fig. 6. Scaled system with link and flow information (DSCALEs).

*grp2* respectively:

$$\sum_i P_i = 4.2 = a_1 P', \tag{7}$$

$$\sum_j P_j = 3.4 = a_2(P' + P'), \tag{8}$$

where the sums are over all links $i$, respectively $j$, along the path of *grp1*, respectively *grp2*. Continuing this way we can write down similar equations for the rest of the 118 groups of flows.

According to the procedure, we also write the following set of equations:

$$\sum_{i=1}^{120} a_i = \delta, \tag{9}$$

$$s_1 = \frac{\sum_{i=1}^{120} \alpha_i \lambda_i I_i}{\sum_{i=1}^{120} \lambda_i I_i}, \tag{10}$$

$$I_i = \begin{cases} 1 & \text{if } grp_i \text{ traverses link } \textit{SVL(dc1)-SVL(hpr)} \\ 0 & \text{otherwise} \end{cases}$$

$$s_2 = \frac{\sum_{i=1}^{120} \alpha_i \lambda_i J_i}{\sum_{i=1}^{120} \lambda_i J_i}, \tag{11}$$

$$J_i = \begin{cases} 1 & \text{if } grp_i \text{ traverses link } \textit{LAX(hpr)-LAX(dc2)} \\ 0 & \text{otherwise} \end{cases}$$

$$C_1' = s_1 \cdot 1Gbps, \tag{12}$$

$$C_2' = s_2 \cdot 1Gbps. \tag{13}$$

Equations (10)...(13) ensure that the traffic intensity on the two congested links (from left-to-right) is the same in the original and scaled network. Notice that because of traffic symmetry, the factor $s_i (i = 1, 2)$ is the same on both directions of the link, and hence it suffices to compute it on one direction only. All the equations together comprise a system of 125 equations and 125 unknowns ($\alpha_1, ..., \alpha_{120}$, $C_1'$, $C_2'$, $s_1, s_2$, and $P'$). We solve the system using $\delta = 60$ and get: $C_1' = 450$Mbps, $C2' = 550$Mbps, $P' = 4.78$ms, and the values of $\alpha_1 ... \alpha_{120}$ (in this scenario $\alpha_1 = 0.88, \alpha_2 = 0.36$, and so on).

In general, if we have $N$ groups of flows that traverse $K$ bottleneck links in total, we can always write a set of $N + 2K + 1$ equations with $N + 2K + 1$ unknowns.

### D. Two-way traffic

Traffic on bidirectional network links is often asymmetric [45]. This traffic asymmetry may result from the nature of the applications. For example, web and ftp applications are inherently asymmetric; One direction carries small request messages and the other direction carries the actual data. In this case, one can assume that the TCP acknowledgements (ACKs) for the data will experience negligible queueing delays. This is the case in the simple topology used to introduce the two methods.

However, there are links where actual data transfer exists in both directions (two-way traffic). This is the case in the scenario with the CENIC backbone topology. In such cases, ACK packets interact with data packets and may experience significant queueing delays. In addition, in-contrast to one-way traffic where the ACKs provide a reliable clock to regulate traffic and keep the queue length variations modest, in two-way traffic the ACKs can become compressed together and hence loose their clocking properties. In the literature (e.g [46], [12]) this phenomenon is called *ACK compression*.

DSCALEd can be applied in the case of two-way traffic without any modification. Applying DSCALEs is more involved. In general, the corresponding set of equations of the DSCALEs method yield two different scaling factors, one for each direction. Hence, one would scale the capacity along one direction with a different factor than that along the other direction. This would scale the dynamics of data packets and acknowledgments of the same flow differently, and, as a result, the method would be inaccurate. Note that this is not an issue when the two-way traffic is "symmetric", that is, when flows on both directions of a link arrive with the same rates and follow the same paths inside the network. In this case, the capacities of both directions are scaled by the same amount.

## V. Simulation Results

In this section we use ns-2 [47] to investigate whether our methods can accurately predict the performance of IP networks from scaled-down replicas. We work with the simulation setups shown in Figures 1 and 4.

A word on network traffic properties is in order. It has been shown that the size distribution of flows on the Internet is heavy-tailed [7]. Hence, Internet traffic consists of a large fraction of short flows, and a small fraction of long flows that carry most of the traffic. Also, it has been recently argued that since network sessions arrive as a Poisson process [48], [49], [11] [7], network flows are *as if* they were Poisson [4]. (In

[7] That network sessions are Poisson is not surprising since a Poisson process is known to result from the superposition of a large number of independent user processes.

particular, the equilibrium distribution of the number of flows in progress is *as if* flows arrive as a Poisson process.)

We have taken these observations into account and used in most of our simulations heavy-tail distributed, Poisson flows. (Of course, while flow arrival times are Poisson, packet arrivals are dictated by the TCP dynamics.) We have also run simulations where flow arrival times are not Poisson, but are dictated from real traces. Notice that the Poisson assumption at the flow level is *not* a requirement for DSCALEd, it is only for DSCALEs as shown in the theoretical analysis in Section VI. Interestingly, the simulation results in Section V-E imply that DSCALEs is reasonably accurate in predicting performance in practice, even when flows are not Poisson.

We use the ns-2 built-in routines to generate sessions consisting of a single object each. This is what we call a flow in the simulations. Each flow consists of a Pareto-distributed number of packets, with an average size of 12 packets and a shape parameter equal to 1.2.

### A. Simple topology experiments

We begin by using DSCALEd to show that a number of performance measures of the original network depicted in Figure 1 can be predicted by the scaled replica depicted in Figure 2. (Recall that the fixed delay factor $d$ equals 100.416ms.) In particular, we compare the distribution of the number of active flows, the histogram of the flow transfer times (flow delays) and the distribution of the queue length in front of link *R2-R3* of the original topology and link *R2'-R3'* of the scaled topology. The flow arrival rates are the same within each group and equal 51 flows/sec. The results do not depend on whether the rates are equal or not, as we have verified by simulations. The average packet queueing delay on link *R2-R3* is 41ms and on link *R1-R2* 2ms. The drop probability on link *R2-R3* is approximately $4\%$ and no drops occur on link *R1-R2*.

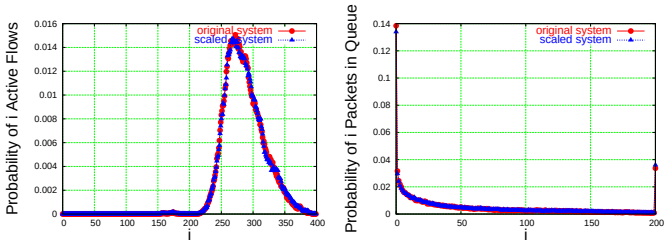



Fig. 7. Distribution of (i) the number of active flows ($HSM = 0.967$), and (ii) the queue length on the bottleneck link ($HSM = 0.962$). (DSCALEd, simple topology)

Figure 7(i) plots the overall distribution of active flows on links *R2-R3* and *R2'-R3'*. It is visually evident from the plot that the two distributions match. [8]

In addition to visually comparing distributions and histograms it is also important to quantify differences using a statistical measure. To this end, to compare two histograms, say $H_1$ and $H_2$, we use the following histogram similarity measure ($HSM$):

$$HSM = 1 - C_v, \tag{14}$$

where $C_v = \sqrt{\frac{\chi^2}{2}}$ is the Cramer's V coefficient, and $\chi^2 = \sum_i \frac{(H_1(i)-H_2(i))^2}{H_1(i)+H_2(i)}$ is the well-known chi-square statistic [50]. The sum is over all histogram chunks and $H_j(i)$ $(j = 1, 2)$ is the frequency of the events observed in the $i^{th}$ chunk. Notice that $HSM$ takes values in $[0, 1]$, with 1 indicating that the two histograms are identical (in this case $\chi^2 = 0$). The HSM value is given in the caption of each plot of this section.

Figure 7(ii) plots the distribution of the queue lengths for the two links. Again, the two distributions match. (Note that the right-most point on the plot gives the probability that there are 200 or more packets in the queue.)

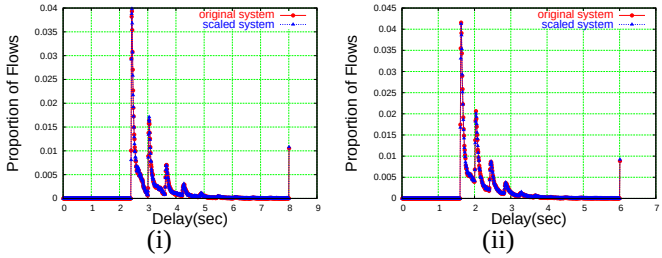



Fig. 8. Histogram of flow delays of (i) *grp1* ($HSM = 0.952$), and (ii) *grp3* ($HSM = 0.960$) flows. (DSCALEd, simple topology)

Figure 8(i) plots the histogram of the flow transfer times (flow delays) for the flows of *grp1*, and Figure 8(ii) does the same for *grp3*. In both cases, we use delay chunks of 10ms each. It is evident from the plots that the distribution of the delays match. 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 and face no congestion, the portion of the curve between the first and the second peaks corresponds to flows which send only one packet and face congestion (but no drops), the next peak corresponds to flows which send two packets and face no congestion, and so on. The right-most point of Figure 8(i) represents the proportion of flows that belong to *grp1* and have a delay of more than 8sec. The right-most point of Figure 8(ii) represents the proportion of flows that belong to *grp3* and have a delay of more than 6sec.

We now proceed to use DSCALEs. Recall that the scaled replica is shown in Figure 3, where $\alpha_1 = 0.6$, $\alpha_3 = 0.4$, $C' = 5$Mbps and $P' = 500$ms.

A comment on how we sample flows is in order. Because of the heavy-tailed nature of the traffic, there is a small number of very large flows that has a large impact on congestion. To guarantee that we sample the correct number of these flows within each group, we separate flows into large (elephants) and small (mice) and sample exactly a proportion $\alpha_i$, $i = 1, 3$, of them.

The plots of the distribution of active flows and the distribution of the queue lengths are identical between the two methods so we don't show them again. However, under the second method, the histogram of flow transfer times requires

[8] In this scenario the distributions of the number of active *grp1* and *grp2* flows also match between the two systems. We do not present the plots since they are similar to Figure 7(i).

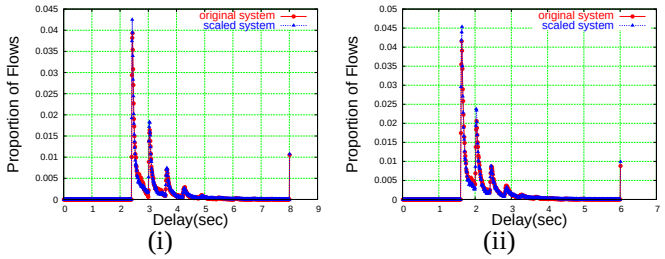



Fig. 9. Histogram of normalized flow delays of (i) *grp1* ($HSM = 0.903$), and (ii) *grp3* ($HSM = 0.902$) flows. (DSCALEs, simple topology)

some normalization. Figure 9(i) plots the histogram of the normalized flow transfer times (flow delays) for the flows of *grp1*, and Figure 9(ii) does the same for *grp3*. In both cases, we use delay chunks of 10ms each, and the normalization is done as follows: (Normalized Flow Delay) $= a_i$ (Delay due to Propagation Time) $+ s$ (Total Queueing Delay), where $a_i$, $i = 1, 3$, are the sampling ratios and $s$ equals $\frac{C'}{C_2}$, the ratio of the capacities of the congested link in the original and scaled network. Equations (4) and (5) provide the justification for the multiplication with the $a_i$'s. A rigorous justification for the use of $s$ is given in Section VI. The intuition behind this decision is that if queue sizes are the same in the two systems, queueing delay is inversely proportional to the link capacity. (Note that from a practical point of view, it is quite easy to separate in simulations the queueing delay from the rest of the delay. Hence, it is easy to perform the required normalization.) Returning to the figures, it is evident from the plots that the distribution of the normalized delays match.

In summary, both methods are quite accurate: they achieve $HSM$ values that are at least 0.90. Clearly, it is not possible to achieve $HSM$ values equal to one, since both scaled systems ignore the small, yet non-zero, queueing delay of link *R1-R2*. Notice also that DSCALEd is a bit more accurate than DSCALEs. The former yields $HSM$ values larger than 0.95 while the later yields values that are at least 0.90. The reason for this is that DSCALEs uses sampling. In theory, random sampling captures the statistics of an infinite length input accurately. But, in practice, we work with a finite length input, that is, with a finite number of flows. Hence, the statistics of the subset of sampled flows might differ a bit from the statistics of the set of all flows. Since flow sizes are heavy-tailed, some flows might be exceptionally large. If some of these flows are not sampled, their absence can have a sizeable effect on the overall statistics of the sampled flows. Yet, the total number of flows in our experiment is large, and the performance penalty that we pay due to sampling is quite small. For a more detailed discussion on the relationship between traffic sampling and accuracy, see Section VII.

### B. CENIC backbone experiments

We now present simulation results for the topology shown in Figure 4, when using either of the proposed methods (the corresponding scaled systems are shown in Figures 5 and 6). The flow arrival rates are the same within each group and equal 250 flows/sec. The average queueing delays on the congested links *SVL(dc1)-SVL(hpr)*, *SVL(hpr)-SVL(dc1)*, *LAX(dc2)-LAX(hpr)* and *LAX(hpr)-LAX(dc2)* are respectively 0.15ms, 0.31ms, 0.18ms and 0.13ms. Drops occur only on *SVL(hpr)-SVL(dc1)*, with a probability 2%.

Because of limitations in space we do not present results for all the groups of flows. We only present flow delay histograms for *grp1* and *grp2* flows, which were defined in Section IV. (recall that *grp1* flows follow the path: *SDG(dc3)-SDG(dc1)-SDG(hpr)-LAX(hpr)-SVL(hpr)-SVL(dc1)-SVL(dc3)*, and *grp2* flows follow the path: *SVL(dc3)-SVL(dc1)-SVL(hpr)-LAX(hpr)-LAX(dc2)-TUS(dc1)*). We also present the packet delay histograms and the distribution of active flows on links *SVL(hpr)-SVL(dc1) and LAX(dc2)-LAX(hpr)*. Further, we show that the two methods can predict ACK compression by providing the ACK delay histograms in both directions of link *SVL(hpr)-SVL(dc1)*. Similar results hold for the rest of the groups of flows and for the queue length distributions.

We begin by using DSCALEd. The scaled system is shown in Figure 5. The fixed delay factor for each of the 120 groups present in the scaled system is computed as described in Section IV. Figures (10) ... (13) represent the simulation results when using DSCALEd.

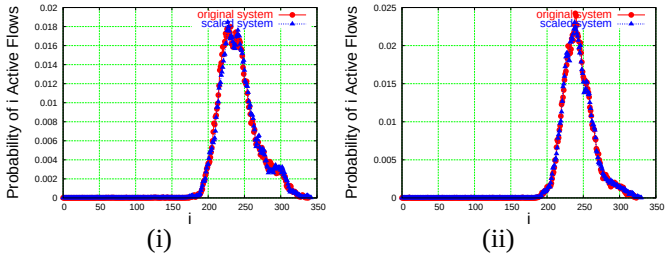



Fig. 10. (i) Distribution of number of active flows on *SVL(hpr)-SVL(dc1)* ($HSM = 0.944$), and (ii) Distribution of number of active flows on *LAX(dc2)-LAX(hpr)* ($HSM = 0.953$). (DSCALEd, CENIC topology)

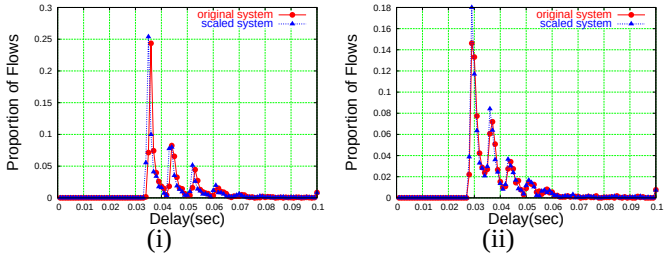



Fig. 11. Histogram of flow delays of (i) *grp1* ($HSM = 0.852$), and (ii) *grp2* ($HSM = 0.853$) flows. (DSCALEd, CENIC topology)

From the plots we can conclude that the scaled system produced by DSCALEd can predict the various performance measures of interest with a high accuracy.

We now move to the second method for which the scaled system is shown in Figure 6. Recall from Section IV that $C_1' = 450$Mbps, $C2' = 550$Mbps, $P' = 4.78$ms, and we know the values of $a_i$'s, $(i = 1...120)$.

The results are shown in Figures (14) ... (17). The normalization of the packet queueing delays is done as follows: (Normalized Queueing Delay) $= s_i$ (Queueing Delay), where $s_i$, $i = 1, 2$, the ratio of the capacities of the congested links in

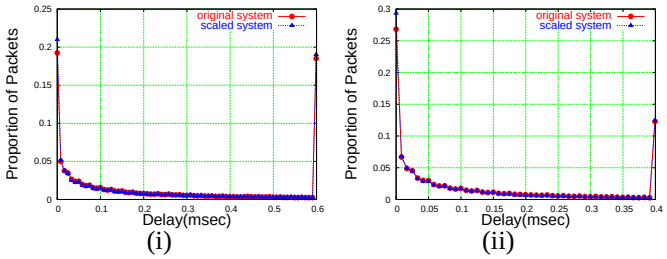



Fig. 12. (i) histogram of packet delays on link *SVL(hpr)-SVL(dc1)* ($HSM = 0.967$), and (ii) histogram of packet delays on link *LAX(dc2)-LAX(hpr)* ($HSM = 0.963$). (DSCALEd, CENIC topology)

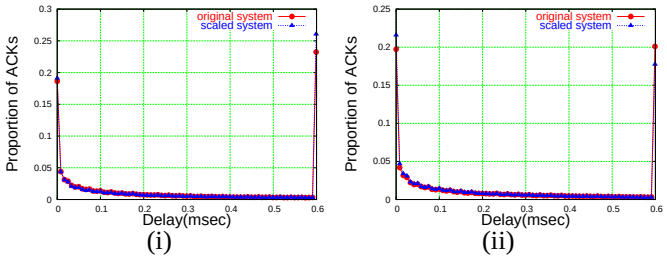



Fig. 13. Histogram of ACK delays on (i) forward direction ($HSM = 0.961$) and (ii) backward direction ($HSM = 0.965$), of link *SVL(hpr)-SVL(dc1)*. (DSCALEd, CENIC topology)

the original and scaled network. The plots are similar as before and their explanation is the same. Also, the normalization of the flow delays is done in the same manner as before.

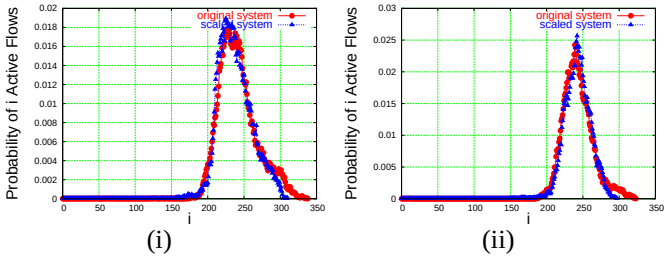



Fig. 14. (i) Distribution of number of active flows on *SVL(hpr)-SVL(dc1)* ($HSM = 0.850$), and (ii) Distribution of number of active flows on *LAX(dc2)-LAX(hpr)* ($HSM = 0.834$). (DSCALEs, CENIC topology)

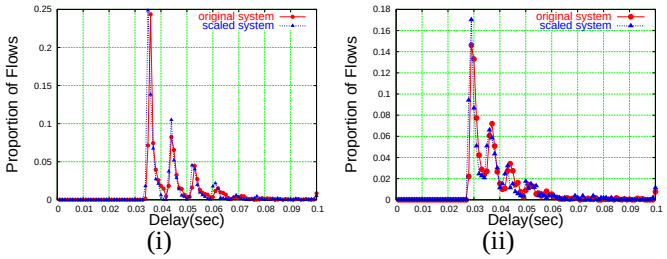



Fig. 15. Histogram of normalized flow delays of (i) *grp1* ($HSM = 0.741$), and (ii) *grp2* ($HSM = 0.779$) flows. (DSCALEs, CENIC topology)

It is obvious from the plots that the scaled system produced by DSCALEs can accurately predict the various performance measures of interest.

Based on the figures of this subsection, we make a couple of interesting observations. First, the methods are slightly

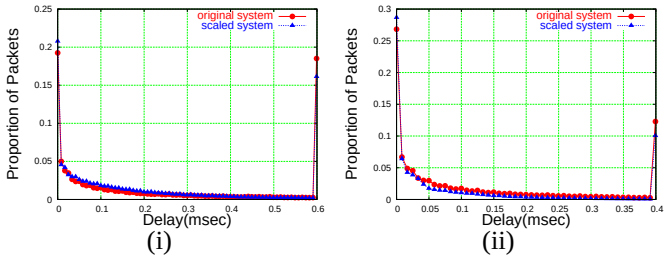



Fig. 16. (i) histogram of normalized packet delays on link *SVL(hpr)-SVL(dc1)* ($HSM = 0.912$), and (ii) histogram of normalized packet delays on link *LAX(dc2)-LAX(hpr)* ($HSM = 0.809$). (DSCALEs, CENIC topology)

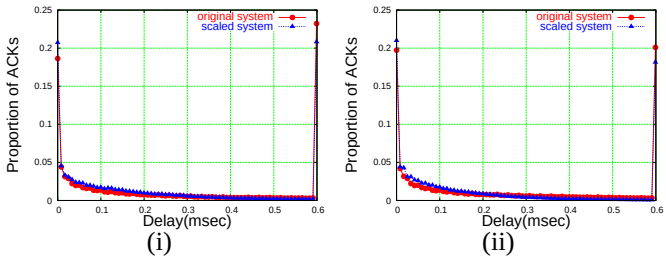



Fig. 17. Histogram of normalized ACK delays on (i) forward direction ($HSM = 0.898$) and (ii) backward direction ($HSM = 0.833$), of link *SVL(hpr)-SVL(dc1)*. (DSCALEs, CENIC topology)

less accurate in the CENIC topology scenario than in the simple topology scenario. This is expected: CENIC is a very large network, and downscaling ignores a large number of uncongested links. Second, notice that both methods do a bit worse when predicting end-to-end delays than when predicting other per-link measures. This is because end-to-end delays depend on all the links along the path, and the small queueing delays due to the uncongested links are not captured in the scaled systems. Last, notice that DSCALEd is again more accurate than DSCALEs. As in the simple topology scenario, this is due to the inaccuracies introduced by undersampling. In particular, since the CENIC experiment consists of 600 groups of flows and we want to keep the total number of flows reasonable (1.5 million flows), there are only 2500 flows per group, which is not a very large number to take samples from. The slightly less than 0.80 $HSM$ values in Figure 15 exemplify this issue.

### C. Two-way asymmetric data traffic

To demonstrate how our methods work in the presence of two-way *asymmetric* data traffic, we use the setup shown in Figure 1 and we inject another group of flows with a rate equal to 100 flows/sec. This group, which we call *grp4*, enters at *R3* and exits at *R1*. The corresponding scaled system is the one shown in Figure 2 with the difference that the new group of flows is present and goes from *R3'* to *R2'*.

Figure 18 shows that DSCALEd can predict the ACK delays, which implies that the ACK compression is the same between the two systems. The method predicts the rest of the performance metrics of interest with the same high accuracy. We do not present the corresponding plots since they are similar as before.

We now move to DSCALEs. Taking $\delta = 2.5$, we get

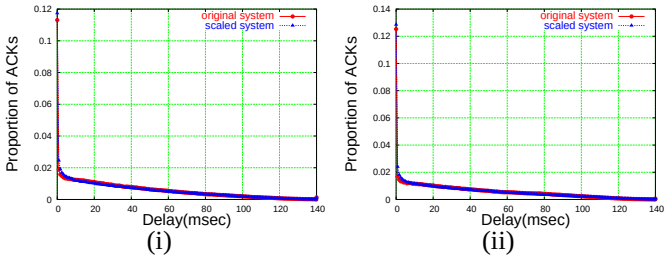



Fig. 18. Histogram of ACK delays on (i) forward direction ($HSM = 0.961$) and (ii) backward direction ($HSM = 0.966$), of link *R2-R3* and *R2'-R3'*. (DSCALEd, simple topology, two-way asymmetric data traffic)

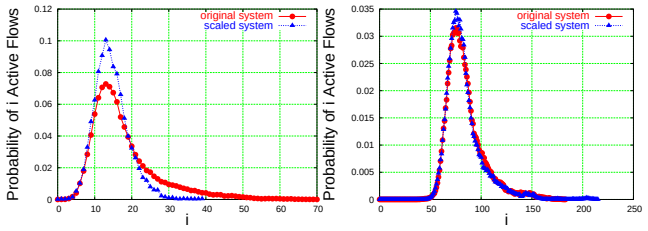



Fig. 20. (i) *grp3* distribution of flows ($HSM = 0.737$), and (ii) overall distribution of flows ($HSM = 0.929$). (DSCALEs, simple topology, large queueing delays)

$\alpha_1 = 0.9375, \alpha_3 = 0.625$ and $\alpha_4 = 0.9375$, where $\alpha_1$ and $\alpha_3$ are the sampling factors for *grp1* and *grp3* respectively, and $\alpha_4$ is the sampling factor for *grp4*. According to the method, the propagation delay is $P' = 320$ms for both directions of the link *R2'-R3'*, whereas the capacity for the forward direction (*R2'-R3'*) is $C'_f = 7.8125$Mbps and the capacity for the backward direction (*R3'-R2'*) is $C'_b = 9.375$Mbps.

Figure 19 demonstrates how DSCALEs predicts the overall distribution of active flows on the forward and backward direction of the link *R2-R3*. From the figure we observe that the accuracy of the method is still quite good. However, it has been reduced, compared to the accuracy of the method for the simple topology experiments presented earlier. As mentioned before, this is because the dynamics of data packets and acknowledgments of the same flow are being scaled differently, because we scale the capacity on each direction of the link *R2-R3* with a different factor.

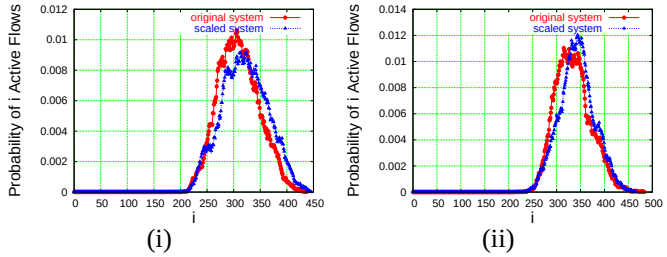



Fig. 19. Distribution of Active Flows on (i) forward direction ($HSM = 0.83$) and (ii) backward direction ($HSM = 0.85$), of link *R2-R3* and *R2'-R3'*. (DSCALEs, simple topology, two-way asymmetric data traffic)

### D. A closer look at large queueing delays

Recall that during the derivation of the DSCALEs equations in Section IV, the assumption was that end-to-end queueing delays are small in comparison to total end-to-end delays. To illustrate how DSCALEs works when queueing delays are relatively large compared to the end-to-end delays, we consider the original topology shown in Figure 1 with the only difference that $P_2 = 25$ms and the flow arrival rate is 47 flows/sec (same for all the groups). In this case, the average queueing delay on link *R2-R3* (as calculated by the simulator) is 26ms, which is comparable to the end-to-end latency of a *grp3* packet. The scaled system is shown in Figure 3, where $\alpha_1 = 0.83$, $\alpha_3 = 0.17$, $P' = 150.4$ms and $C' = 5$Mbps.

Figure 20(i) plots the *grp3* distribution of active flows for both the original and the scaled system. The method is not very accurate. This is evident visually and from the relatively low, for the simple topology scenario, $HSM$ value. Yet, the method can accurately predict "aggregate-based" performance metrics, like the overall distribution of active flows and the queue length distributions. Section VI clearly defines which metrics are "aggregate-based", and proves the above statements. Figure 20(ii) shows the overall distribution of active flows on the two systems, and it is evident that the method is quite accurate.

### E. Real Internet traces

We now illustrate how our methods works with real Internet traffic. We use the setup shown in Figure 1 with $C_1 = 1$Gbps, $C_2 = 400$Mbps, $P_1 = P_2 = 1$ms, and with routers having the same characteristics as before. In this experiment the flow arrival times are no longer dictated by a Poisson process through the simulator, but instead are extracted from real Internet traces from the Abilene-I data set [51]. The flow size is Pareto-distributed with the same parameters as before.

As mentioned earlier, the Poisson assumption at the flow level is *not* a requirement for DSCALEd to work. Figure 21 shows how the method can accurately predict the *grp1* and *grp3* flow delay histograms. The same holds for the rest of the performance measures of interest. [9]

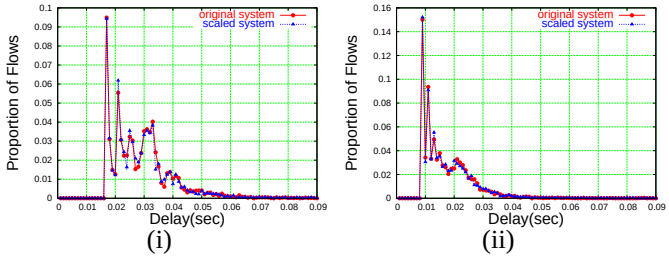



Fig. 21. Histogram of flow delays of (i) *grp1* ($HSM = 0.919$), and (ii) *grp3* ($HSM = 0.926$) flows. (DSCALEd, simple topology, real trace)

Figure 22 shows how well DSCALEs can predict the *grp1* and *grp3* flow delay histograms. Notice that in these plots there are small visual differences between the histograms obtained from the original and scaled system. Also, the $HSM$ values are around 0.80, the lowest values obtained in the simple topology scenario. This is caused by the non-Poisson flow arrivals. The theoretical analysis in Section VI discusses in detail why DSCALEs requires Poisson flow arrivals. It is

[9]Note that we use a different trace for each group of flows. This is in accordance to the theoretical requirement for independent flow arrivals within different group of flows, discussed in Section VI.

interesting to note though that these simulation results imply that, in practice, the accuracy of the method is quite resilient to deviations from this requirement, since the differences between the histograms are small.

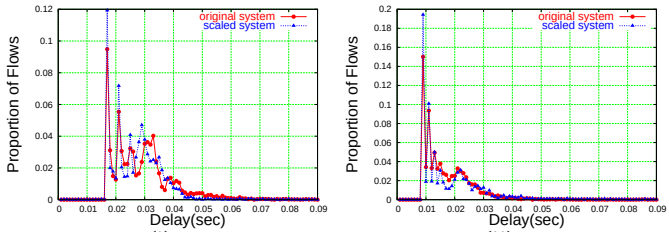



Fig. 22. Histogram of flow delays of (i) *grp1* ($HSM = 0.787$), and (ii) *grp3* ($HSM = 0.824$) flows. (DSCALEs, simple topology, real trace)

## VI. Theoretical support

In this section we formalize the important results of this paper. First, we show that it is always possible to construct a scaled network from any original network by removing uncongested links, without violating the following natural constraint: the links of the scaled system must be traversed by the same groups of flows and in the same direction as in the original system. Second, we prove that DSCALEd preserves network performance under the assumption that uncongested links do not alter the arrival process of the packets that go through them. This result holds for any stationary arrival process. Third, we prove that DSCALEs preserves aggregate performance metrics under the assumption that flow arrivals are Poisson, and per-group performance metrics under the additional assumption that end-to-end delays are mainly dictated by propagation delays.

### A. Downscaling Topology

Recall that the first three steps in the method summaries presented in Section IV are identical. In other words, both methods perform the same tasks when removing uncongested links and reconnecting together the remaning links to form the downscaled topology. The way by which the methods choose to attach one link to another and build the scaled topology depends on the direction that groups of flows are traversing the links in the original system. In order to preserve performance one has to make sure that the links of the scaled system are being traversed by the same groups of flows and in the same direction as in the original system. But is this always possible? For example, consider two links in the original system, say links $(e_1, e_2)$ and $(e_3, e_4)$, and two groups of flows, where the first group traverses these links in the direction $(e_1, e_2)$, $(e_3, e_4)$ and the second group in the direction $(e_1, e_2)$, $(e_4, e_3)$. It is easy to see that building the scaled system by attaching these two links together is not possible. Thus, before we analyze DSCALEd and DSCALEs, we prove the following lemma:

*Lemma 1:* The construction of the scaled topology is always possible, as long as minimum cost routing is used.

*Proof:* Without loss of generality, consider again the links $(e_1, e_2)$, $(e_3, e_4)$ and two groups of flows that traverse them. We claim that the relative direction by which the two groups are traversing one link is the same with the relative direction that they traverse the other. If not, it means that while both groups left the first link from the same end-point, say $e_2$, one group entered the second link from $e_3$ and the other from $e_4$. But since both groups follow the path with the minimum cost to their destination, they follow the minimum sub-path from $e_2$ to link $(e_3, e_4)$, and they must enter the link from the same point.[10] ■

### B. Preserving Performance with DSCALEd

Under the assumption that uncongested links do not contribute to the end-to-end queueing delays and do not affect network and flow dynamics, it is easy to see why the proposed method would preserve performance if one would add fixed delays at exactly the point where uncongested network segments existed in the original network. By adding fixed delays to packets, we make sure that the transmission and propagation delays remain unaltered between the original and the scaled system. And, by retaining all congested links we make sure that end-to-end queueing delays are unaltered.

However, DSCALEd adds fixed delays at the source of each packet. This minimizes bookkeeping since one doesn't have to retain information about where each uncongested link used to be in the original network. The following lemma establishes that adding the fixed delays at the sources does not alter the network dynamics.

*Lemma 2:* If flow arrivals are independent among different groups and stationary, adding the fixed delays at the sources does not alter the packet arrival processes at the links.

*Proof:* Consider two scaled systems. In the first, fixed delays are added at exactly the points where uncongested network segments existed in the original network. In the second, fixed delays are added at the sources of the packets. We prove that the packet arrival processes at the links of these systems are the same.

Without loss of generality, let $I_1(t)$ and $I_2(t)$ denote the stationary flow arrival processes of two groups of flows, *grp1* and *grp2* respectively, at a congested link. Now, let $d_1$ denote the delay factor added to all the packets of *grp1* flows and $d_2$ the delay factor added to all the packets of *grp2* flows, due to downstream uncongested segments that were removed from the network. This results in a time-shift of the *grp1* arrival process by $d_1$, and of the *grp2* arrival process by $d_2$. Since flows between different groups arrive independently and the arrival processes are stationary, $(I_1(t), I_2(t)) \stackrel{d}{=} (I_1(t + d_1), I_2(t + d_2))$, that is, the overall arrival process to the link remains the same if the fixed delays are added prior to the link. This takes care of the first packet of each flow.

What about subsequent packets of a flow? First, note that networks are discrete-event systems, clocked by transmissions and acknowledgments of packets. Now, consider the first packet of a flow that arrives at the congested link in the first scaled network (where fixed delays are added at exactly the point where uncongested network segments existed in the

[10] In case there are equal minimum cost paths and routers use load-balancing, one may need to add a zero-delay link to the scaled topology in order to allow traffic to enter $(e_3, e_4)$ from both directions.

original network). The packet will experience an amount of queueing, and then, an amount of transmission and propagation delay, before it reaches its destination. Now, consider the lifetime of the same packet in the second network (where fixed delays are added at the sources). This packet experiences the same amount of queueing delay, but then, is delivered directly to its destination. This means that the source of the packet will receive an acknowledgment earlier, compared to the first system. But the amount of propagation and transmission delays that the packet did not experience after leaving the link in the second system, is accounted in the fixed delay imposed at the source of the packet. This means that while the acknowledgment was received earlier, the next packet arrival (triggered by the acknowledgment) is delayed until the acknowledgment is received in the first system as well. As a result, the second packet arrives at the link of the second system at the same time as the corresponding packet arrives at the link of the first system. Continuing inductively, we can see that the packet arrival process at the links of the two systems has not been altered. ■

We can now state the following theorem whose proof follows immediately from the discussion above.

*Theorem 1:* If flow arrivals are independent among different groups, and the arrival process is stationary, DSCALEd preserves the network dynamics in distribution. In particular, the distribution of metrics like the number of active flows, the flow delays, the queue lengths etc. remain unchanged.

It is interesting to point out that if the fixed delays are added at exactly the point where uncongested network segments exist in the orignal network, DSCALEd preserves the network dynamics as a function of time.

### C. Preserving Performance with DSCALEs

First, we briefly review one main result from [17]. In a network where flows arrive as a Poisson process, do the following operations to construct a slower replica: (i) sample each incoming flow independently with probability $\alpha$, (ii) reduce link capacities by the same factor $\alpha$, and (iii) increase propagation delays (and protocol timeouts) by a factor $1/\alpha$. Then, the state of the original network at time $t$, is the same in distribution with the state of the slower replica at time $t/\alpha$. In simple words, the following scaling law holds:

*Law 1:* Performance is preserved if the arrival rate $\lambda$, the capacities $C$, and the propagation delays $P$ change to $\alpha\lambda$, $\alpha C$ and $P/\alpha$ respectively.

We now state two results for DSCALEs. We distinguish between performance metrics that depend on the properties of the "supergroup" consisting of all groups of flows going through a link, called aggregate-based metrics, and metrics that also depend on the properties of a specific group of flows, called group-based metrics. To make the distinction clear, supposed we have a single link that is traversed by two groups of flows. If we vary the value of the arrival rates of the two groups such that their sum remains the same, aggregate-based metrics remain unchanged, whereas group-based metric change. The distribution of the number of active flows (overall distribution), and the queue length distributions are clearly aggregate-based metrics. The flow transfer times depend on propagation and queueing delays, and since the latter are aggregate-based, the same holds for flow transfer times. In contrast, the distribution of the number of active flows of a particular group (per-group distribution) is clearly a group-based metric.

*Theorem 2:* Using DSCALEs, the distributions of aggregate-based metrics, e.g. the queue length distributions and the overall distribution of active flows, remain unchanged.

*Proof:* Without loss of generality, look at the link *R2-R3* of the network in Figure 1, and assume that the flows of all groups belong to a new "supergroup". As a result of merging two Poisson processes with rates $\lambda_1$ and $\lambda_3$, the arrival process at the original link is still Poisson with rate $\lambda = \lambda_1 + \lambda_3$. (Notice that *grp1* flows arrive as a Poisson process at the link *R2-R3* since the first, uncongested link does not alter the traffic statistics by assumption.) Further, the average propagation delay is $P = \frac{\lambda_1(P_1+P_2)+\lambda_3 P_2}{\lambda_1+\lambda_3}$.

Let's compute these quantities at the link *R2'-R3'* of the scaled network in Figure 3. It is a simple property of the Poisson process that sampling a proportion $\alpha$ of the points of a rate $\lambda$ Poisson process will yield a Poisson process with rate $\alpha\lambda$. In addition, the independent sampling process does not destroy the i.i.d nature of the flow sizes. Thus, *grp1* flows arrive as a Poisson process with rate $\alpha_1\lambda_1$ and *grp3* of flows arrive as a Poisson process with rate $\alpha_3\lambda_3$. Hence, the overall arrival process is still Poisson with rate $\alpha_1\lambda_1 + \alpha_3\lambda_3$. The average propagation delay equals $\frac{\alpha_1\lambda_1 P' + \alpha_3\lambda_3 P'}{\alpha_1\lambda_1+\alpha_3\lambda_3} = \frac{\lambda_1(P_1+P_2)+\lambda_3 P_2}{\alpha_1\lambda_1+\alpha_3\lambda_3}$.

Hence, the average arrival rate $\lambda$, the capacity $C_2$, and the average propagation delay $P$, change to $s\lambda$, $sC_2$ and $P/s$ respectively. Thus, the link *R2'-R3'* is a slower, by a factor of $s$, replica of the link *R2-R3*, and, according to Law 1, performance metrics such as the distribution of active flows and the queue length distribution are the same on the two links. ■

*Theorem 3:* Using DSCALEs, the distributions of group-based performance metrics, e.g. the per-group distribution of active flows, are altered, unless the end-to-end queueing delays are negligible compared to the total end-to-end delays.

*Proof:* Without loss of generality, lets look again at the link *R2-R3* of the network in Figure 1, and concentrate only on *grp3* flows. The average arrival rate of this group at the original link equals $\lambda_3$, its propagation delay is $P_2$ and the link capacity is $C_2$. Now look at the link *R2'-R3'* of the scaled network in Figure 3. The average arrival rate of *grp3* flows equals $\alpha_3\lambda_3$, the propagation delay equals $P' = \frac{P_2}{\alpha_3}$ and the link capacity is $sC_2$. Since $s \neq \alpha_3$, Law 1 does not apply.

Now lets look at the case where the end-to-end queueing delay is insignificant compared to the end-to-end propagation delay. In this case, the round-trip time of each packet is dictated by the propagation delay only. Hence, the round-trip time of *grp3* packets is stretched by $\alpha_3$. Since the flow arrival rate and the packet round-trip time are stretched by the same factor $\alpha_3$, the link *R2'-R3'* is a slower, by a factor of $\alpha_3$, replica of the original link. Considering *grp1* flows only, their rate changes to $\alpha_1\lambda_1$ and their round-trip time is stretched by $\alpha_1$ ($P' = \frac{P_1+P_2}{\alpha_1}$). Hence, the link *R2'-R3'* is also a slower replica of the original link by a factor $\alpha_1$. Since the link *R2'-R3'* comprises a slower replica on a per-group basis, per group performance metrics are preserved. ■

## VII. Computational savings and accuracy

In this section, we present results which illustrate the effectiveness of the proposed methods in reducing the computational requirements of simulations. The metric of interest is the time needed to run an experiment. (This time should not be confused with the simulator's "virtual" time.)

Clearly, as one reduces the size of the scaled network and the amount of traffic that is fed into it, the simulation time also reduces. But, this may occur at the expense of accuracy. For this reason, we define a metric to quantify accuracy, and whenever we refer to time-savings we also report the corresponding accuracy. In particular, to quantify accuracy we use the average histogram similarity measure (HSM), where the average is taken over the HSMs obtained from comparing: (i) the overall distribution of active flows in the original and scaled network, (ii) the end to end delay histogram of the groups of flows of interest in the two networks, and (iii) the packet queueing delay histogram on the congested links of the two networks.

For our experiments we used a 2GHz processor with 2GB memory (RAM). Notice that in order to facilitate large-scale simulations, we had to modify the built-in routines of ns-2,[11] in order to allocate memory for the TCP sessions dynamically. In the absence of this modification, ns-2 would statically allocate memory for all sessions at the beginning of the simulation, which restricts the total number of sessions that one may have per experiment due to memory issues. Note that the modification does not alter the simulation-times, except for removing the time that it takes to load the flow-information in the memory at the beginning of the simulation.

| Method | Size | Traffic | Time(min) | Avg. HSM |
|---|---|---|---|---|
| Original | 1 | 1 | 61 | 1 |
| DSCALEd ($\alpha = 1$) | 1/2 | $2/3$ | 31 | 0.96 |
| DSCALEd ($\alpha = 0.7$) | 1/2 | $7/15$ | 24 | 0.94 |
| DSCALEd ($\alpha = 0.5$) | 1/2 | $1/3$ | 15 | 0.92 |
| DSCALEd ($\alpha = 0.3$) | 1/2 | $1/5$ | 9 | 0.87 |
| DSCALEd ($\alpha = 0.2$) | 1/2 | $2/15$ | 5 | 0.84 |
| DSCALEd ($\alpha = 0.1$) | 1/2 | $1/15$ | 3 | 0.81 |
| DSCALEs ($\delta = 1$) | 1/2 | $1/3$ | 15 | 0.91 |
| DSCALEs ($\delta = 0.6$) | 1/2 | $1/5$ | 8 | 0.87 |
| DSCALEs ($\delta = 0.4$) | 1/2 | $2/15$ | 6 | 0.82 |
| DSCALEs ($\delta = 0.3$) | 1/2 | $1/10$ | 4 | 0.81 |
| DSCALEs ($\delta = 0.2$) | 1/2 | $1/15$ | 3.5 | 0.78 |

TABLE II

Simulation time for the simple topology.

Table II shows how effective our methods are when applied to the simple tandem topology shown in Figure 1. The size column indicates the fraction of the links of the original network that are part of the scaled one. Since we retain one of the two original links, the size of the scaled network is half the original size. The traffic column indicates the fraction of flows of the original network that traverse the scaled network. Recall that only two out of the initial three groups of flows are present in the scaled experiment. Further, recall that if scaling in time [1], [17] is used in conjunction with DSCALEd, each flow is sampled with probability $\alpha$. Hence, under DSCALEd, the fraction of flows in the scaled experiment equals $\alpha\frac{2}{3}$. Finally, recall that under DSCALEs $\delta = \sum \alpha_i$, where $\alpha_i$ is the sampling factor for each flow-group $i$, $1 \leq i \leq N$. (When a group is not present in the scaled network due to topological downscaling, we consider the corresponding $\alpha_i$ to be equal to 0.) Since originally we have $N$=3 groups, under DSCALEs a fraction $\frac{\delta}{3}$ of the original flows traverse the scaled network.

Simulating the simple tandem topology with a total of $390000$ flows takes $61$min, whereas using DSCALEd with $\alpha = 1$ takes 31min. To obtain more impressive savings, we can also scale the system in time by a factor of $\alpha < 1$. For example, when we use an $\alpha$ of 0.1 the simulation time is only 3 min, but the lower average-HSM value in comparison to the $\alpha = 1$ case indicates that the accuracy of the method is reduced. The gain we get by using DSCALEs is also notable. For example, for $\delta = 1$ the simulation takes only 15 minutes to terminate.

A brief discussion on how to choose a proper $\delta$ is in order. Different values of $\delta$ will result in different values of the sampling factors, $\alpha_i$, for each of the $N$ groups of flows, and hence, in a different total number of flows that are present in the scaled system. In particular, the ratio $\frac{\delta}{N} \leq 1$ equals the proportion of the original traffic that is present in the scaled system. While picking a small $\delta$ expedites simulations, too small a $\delta$ would result in very small sampling factors, and this might lead to undersampling and inaccuracies. For this reason, we recommend that $\frac{\delta}{N}$ not be made too small. In particular, if $M \leq N$ is the total number of groups that have some of their flows present in the scaled system ($\alpha_i > 0$), we recommend that $\frac{\delta}{M} \geq 0.1$. For similar reasons, when one scales a network in time by a factor $\alpha$, we recommend that $\alpha \geq 0.1$. The connection between $\alpha$ or $\delta$ and the accuracy is discussed in detail later in the section. (See Figure 24 and the accompanying text.)

| Method | Size | Traffic | Time(min) | Avg. HSM |
|---|---|---|---|---|
| Original | 1 | 1 | 1725 | 1 |
| DSCALEd ($\alpha = 1$) | 2/41 | 1/5 | 40 | 0.92 |
| DSCALEd ($\alpha = 0.5$) | 2/41 | 1/10 | 16 | 0.83 |
| DSCALEd ($\alpha = 0.1$) | 2/41 | 1/50 | 5 | 0.79 |
| DSCALEs ($\delta = 60$) | 2/41 | 1/10 | 15 | 0.82 |
| DSCALEs ($\delta = 30$) | 2/41 | 1/20 | 11 | 0.80 |

TABLE III

Simulation time for the CENIC topology.

Table III shows the effectiveness of our methods when applied to the more complex topology shown in Figure 4. The total number of flows used in this experiment is $1500000$ belonging to 600 different groups. The corresponding scaled systems are shown in Figures 5 and 6. Out of the 600 initial groups of flows, 120 are present in the scaled experiments. Note that when $\alpha = 1$, the traffic that traverses the scaled network is $1/5^{\text{th}}$ of the original traffic, since only 120 out of the initial 600 groups of flows are present in the scaled experiment.

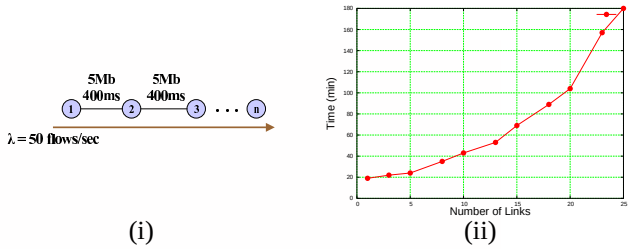
The gain we get by using either of the methods is quite remarkable, close to two orders of magnitude, while the accuracy of the method is quite good.

[11] The patch can be found at http://www-scf.usc.edu/∼fpapadop/.

It is worth pinpointing the cause of these significant computational gains, and investigate the trade-off between execution time and simulation accuracy. There are two reasons why we save time with our methods: (i) topological downscaling, that is, the act of removing uncongested links and nodes, and (ii) traffic sampling, that is, the act of considering as input only a fraction of the original flows.

Let us first concentrate on topological downscaling. Recall that uncongested links have small but non-zero delays. Hence, the more uncongested links one ignores, the less accurate performance prediction is, for example, with respect to the end to end delay. Because of this, the average HSM of DSCALEd for $\alpha$=1, 0.5 and 0.1 is higher in Table II, which corresponds to a scenario where only one out of two links is ignored, than in Table III, which corresponds to a scenario where 39 out of 41 links are ignored. Interestingly enough, our methods are very accurate even in the later scenario.

A comparison of these tables also reveals that the larger the number of links one ignores, the larger the time savings are, as expected. For example, simulations that use DSCALEd with $\alpha$=1 and 0.1 run 2 and 20 times faster than the original simulation in case of the scenario of Table II, and 43 and 345 times faster in case of the scenario of Table III. The large gains associated with topological downscaling leads us to the following question: In what way does the time needed to run an experiment depend on the number of links/nodes that comprise a network topology? To answer this question we use the experimental topology shown in Figure 23(i). For that topology we measure the time needed to run an experiment when we have $n$ nodes and $n-1$ identical links, while varying $n$ from 1 to 25. The total number of flows for this experiment is 100000 and they have the same characteristics as before. Figure 23(ii) shows that the time needed to run the experiment increases with the number of links, with the increase being between linear and exponential. Hence, computational savings are expected to increase dramatically as the size of the network under study increases. We believe this to be the case for any event-driven simulator, not just ns-2, since the number of events that a simulator needs to schedule increases rapidly as the length of the paths grows.




(i) (ii)

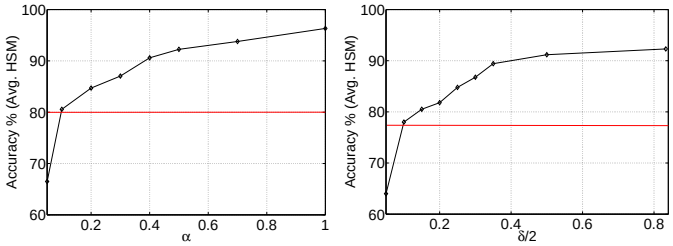
Fig. 23. (i) Experimental topology, and (ii) simulation time vs. number of links on the experimental topology.

Now, let us take a closer look at traffic sampling. Going through both Table II and III it is evident that as the fraction of ignored, non-sampled traffic increases, the time savings also increase. This comes as no surprise, since as the number of flows decreases, the number of simulator events decreases, and the simulator terminates faster. From the tables it is evident that the simulation time depends approximately linearly on the number of flows, which is also somewhat expected.

What is of more interest is how accuracy is affected by traffic sampling. While it is reasonable for the accuracy to decrease as the fraction of ignored traffic increases, the precise relationship between them is hard to predict. In essence, the issue relates to (i) how close to the actual distribution the observed histograms are, which is a matter of convergence, and (ii) how representative of the initial traffic the sampled traffic is, which is a matter of undersampling.

For the observed histograms to converge to the corresponding distributions, enough events (e.g. packet arrivals) should occur. Further, for the sampled traffic to accurately represent the original traffic, the right proportion of short and long flows should exist on the sample, a task that can be tricky when sizes are heavy-tailed. Interestingly enough, in our scenarios 10% of the original flows ($\alpha = 0.1$) is enough to achieve reasonably accurate performance prediction, where in the simple scenario we started with hundreds of thousands of flows belonging to a handful of groups, and in the CENIC scenario we started with millions of flows belonging to hundreds of groups.

Figure 24 shows how accuracy, measured by the average HSM, changes as a function of the intensity of traffic sampling in the simple topology scenario. As it is evident from the plot, for $\alpha < 0.1$ and $\frac{\delta}{2} < 0.1$ the average HSM starts dropping fast, whereas for larger values of $\alpha$ and $\delta$ it is quite high.




(i) (ii)

Fig. 24. Accuracy of the methods as a function of traffic sampling (i) DSCALEd (ii) DSCALEs. (Simple topology experiments)

Finally, we address one unanswered question related to DSCALEd: What is the complexity of adding a fixed delay to each packet? The reported simulation times imply that it is insignificant, but one might argue that as the size of the network increases this might be an issue. In particular, while it is very easy for a simulator to add a fixed delay to the round trip time of each packet, classifying each packet in order to add the proper delay might be computationally expensive if a very large number of source/destination pairs exist. However, hashing techniques can be used very efficiently for this purpose. In a network of $n$ hosts there are at most $O(n^2)$ pairs, and for any meaningful values of $n$ the associated complexity is trivial. For example, from an IP backbone point of view, the hosts mentioned above correspond to Points of Presence (POPs), and $n$ is quite small. Indeed, according to the geographic maps found in [52] $n$ is never more than 70.

## VIII. Conclusion and Future Directions

We proposed two methods, DSCALEd and DSCALEs, to scale down an arbitrary network topology that is shared

by TCP flows and controlled by various AQM schemes. DSCALEd is applicable to any scenario. DSCALEs has some limitations: it cannot accurately predict some group-based metrics when queueing delays are large in comparison to the total end-to-end delays.

Both methods preserve the performance of the original system. Hence, they can be used to study large networks via small replicas. We show this via extensive simulations and theoretical arguments. We also show that simulating a replica can be up to two orders of magnitude faster than simulating the original network. The computational gains might be even more pronounced for larger topologies than the ones that we study.

In the near future we plan to create a software tool that takes as input the original topology and traffic and produces a user-configurable downscaled network configuration.

There are some interesting yet challenging extensions to this work. This paper shows that downscaling the Internet while maintaining performance characteristics is possible. It studies, via simulations, the interplay between the amount of downscaling and the accuracy loss that might occur. What is of great interest is to study this tradeoff using rigorous analytical tools. For example, it would be useful to analytically quantify the relationship between $\alpha$, the factor by which one samples flows, and the accuracy of performance prediction in practical situations where the total number of flows is a large yet finite number. Similarly, it would be helpful to analytically quantify the relationship between the number of uncongested links that are ignored by topological downscaling and the achieved accuracy.

As a final note, topological downscaling is based on the assumption that uncongested links do not alter packet dynamics. A number of works have been cited that support this claim experimentally in a variety of scenarios. Our simulation results indirectly validate this assumption for the case of Internet TCP traffic, and our theoretical results support the assumption under a simplified view of uncongested links, according to which their effect is a fixed delay to each packet. What is of great interest is to theoretically verify this assumption when queueing dynamics of uncongested links are taken into account. While this has been done for the case of open-loop networks [24], no such analysis exists for close-loop systems that resemble the Internet. We believe this to be a very interesting future-work direction.

## REFERENCES


[1] K. Psounis, R. Pan, B. Prabhakar, and D. Wischik, "The scaling hypothesis: Simplifying the prediction of network performance using scaled-down simulations," in *Proceedings of ACM HOTNETS*, 2002.

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

[3] C. Barakat, P. Thiran, G. Iannaccone, C. Diot, and P. Owezarski, "A flow-based model for internet backbone traffic," in *Internet Measurement Workshop*, 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] D. Wischik, "The output of a switch, or, effective bandwidths for networks," *Queueing Systems*, vol. 32, 1999.

[6] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, 1998.

[7] 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*, vol. 5, no. 1, pp. 71–86, 1997.

[8] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the sprint IP backbone," in *IEEE Network*, 2003.

[9] K. Papagianaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C.Diot, "Analysis of measured single-hop delay from an operational backbone network," in *Proceedings of IEEE INFOCOM*, 2002.

[10] J. Liu and M. Crovella, "Using loss pairs to discover network properties," in *ACM SIGCOMM Internet Measurement Workshop*, 2001.

[11] V. Paxson and S. Floyd, "Wide area traffic: the failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, 1995.

[12] J. C. Mogul, "Observing TCP dynamics in real networks," in *Proceedings of ACM SIGCOMM*, August 1992.

[13] J. S. Ahn and P. B. Danzig, "Speedup and accuracy versus timing granularity," *IEEE/ACM Transactions On Networking*, vol. 4, no. 5, Oct. 1996.

[14] D. Nicol and P. Heidelberger, "Parallel execution for serial simulators," *ACM Transactions On Modeling and Computer Simulations*, vol. 6, no. 3, July 1996.

[15] A. Yan and W. B. Gong, "Fluid simulation for high speed networks," *IEEE Transactions On Information Theory*, June 1999.

[16] B. Liu, 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 IEEE INFOCOM*, 2001.

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

[18] R. Pan, B. Prabhakar, K. Psounis, and D. Wischik, "Shrink: Enabling scaleable performance prediction and efficient simulation of networks," *IEEE/ACM Transaction on Networking*, to appear.

[19] K. Papagiannaki, *Provisioning IP Backbone Networks Based on Measurements*, Ph.D. thesis, University College London, March 2003.

[20] C. Fraleigh, *Provisioning Internet Backbone Networks to Support Latency Sensitive Applications*, Ph.D. thesis, Stanford University, June 2002.

[21] C. Fraleigh, F. Tobagi, and C. Diot, "Provisioning IP backbone networks to support latency sensitive traffic," in *Proceedings of IEEE INFOCOM*, 2003.

[22] D. Y. Eun and N. B. Shroff, "Simplification of network analysis in large-bandwidth systems," in *Proceedings of IEEE INFOCOM*, 2003.

[23] D. Y. Eun and N. B. Shroff, "Analyzing a two-stage queueing system with many Point Process arrivals at upstream queue," *Queueing Systems*, vol. 48, September 2004.

[24] D. Y. Eun and N. B. Shroff, "Network decomposition: Theory and practice," *IEEE/ACM Transactions On Networking*, June 2005 (to appear).

[25] D. Y. Eun and N. B. Shroff, "Network Decomposition in the many sources regime," *Advances in Applied Probability*, vol. 36, September 2004.

[26] "Intermapper web server," `https://intermapper.engineering.cenic.org`.

[27] R. Pan, B. Prabhakar, K. Psounis, and M. Sharma, "A study of the applicability of a scaling hypothesis," in *Proceedings of ASCC*, 2002.

[28] 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.

[29] 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.

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

[31] P. Tinnakornsrisuphap and A. Makowski, "Limit behavior of ecn/red gateways under a large number of tcp flows," in *Proceedings of INFOCOM*, 2003.

[32] V. Krishnamurthy, J. Sun, M. Fabloutsos, and S. Tauro, "Sampling internet topologies: How small can we go?," in *Proceedings of International Conference on Internet Computing*, 2003.

[33] T. Bu and D. Towsley, "On distinguishing between internet power law topology generators," in *Proceedings of IEEE INFOCOM*, 2002.

[34] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *Proceedings of ACM SIGCOMM*, 1999.

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

[36] W. Wei, B. Wang, D. Towsley, and J. Kurose, "Model-based identification of dominant congested links," in *ACM SIGCOMM Internet Measurement Conference*, 2003.

[37] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of wide-area internet bottlenecks," in *Proceedings of ACM SIGCOMM conference on Internet measurement*, 2003.

[38] G. de Veciana, G. Kesidis, and J. Walrand, "Resource management in wide-area ATM networks using effective bandwidths," in *IEE J. on Selected Areas in Comm., vol. 13, no. 6, pp. 1081–1090*, Aug. 1995.

[39] A. Fei, G. Pei, R. Liu, and L. Zhang, "Measurements on delay and hop-count of the internet," in *IEEE GLOBECOM- Internet Mini-Conference*, 1998.

[40] F. Begtasevic and P. V. Mieghem, "Measurements of the hopcount in the internet," in *Proceedings of Passive and Active Measurement (PAM)*, 2001.

[41] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale IP matrices from link loads," in *Proceedings of the ACM SIGMETRICS Conference*, 2003.

[42] N. Hu, L. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "Locating internet bottlenecks: Algorithms, measurements and implications," in *Proceedings of ACM SIGCOMM*, 2004.

[43] W. Fang and L. Peterson, "Inter-as traffic patterns and their implications," in *Proceedings of the 4th Global Internet Symposium*, December 1999.

[44] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, D. Papagiannaki, and F. Tobagi, "Design and deployment of a passive monitoring infrastructure," in *Proceedings of the Workshop on Passive and Active Measurements, PAM*, April 2001.

[45] V. Paxson, "End-to-end routing behavior in the internet," *IEEE/ACM Transactions On Networking*, vol. 5, no. 5, October 1997.

[46] L. Zhang, S. Shenker, , and D.D. Clark, "Observations on the dynamis of a congestion control algorithm: The effects of two-way traffic," in *Proceedings of ACM SIGCOMM*, September 1991.

[47] "Network simulator," `http://www.isi.edu/nsnam/ns`.

[48] A. Feldmann, A. C. Gilbert, and W. Willinger, "Data networks as cascades: Investigating the multifractal nature of internet wan traffic," in *Proceedings of ACM SIGCOMM*, 1998.

[49] C. J. Nuzman, I. Saniee, W. Sweldens, and A. Weiss, "A compound model for tcp connection arrivals," in *Proceedings of ITC Seminar on IP Traffic Modeling*, Monterey, 2000.

[50] W. H Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1992.

[51] "Passive measurement and analysis," `http://pma.nlanr.net/Special`.

[52] "Rocketfuel: An ISP topology mapping engine," `http://www.cs.washington.edu/research/networking/rocketfuel/interactive`.