NetScale: Scalable Time-stepped Hybrid Simulation of Large IP Networks

Laurent Fournié, Dohy Hong and Florent Perisse
N2NSoft
11 bd Sébastopol, 75001, Paris FRANCE
{fournie, hong, perisse}@n2nsoft.com

ABSTRACT

This paper presents a scalable time-stepped hybrid simulation algorithm which is well adapted to the simulation of large IP networks (up to one million of competing flows and network elements), while tracking reactive traffic behaviour and packet-level phenomena e.g. timeout trigger or packet burstiness. This simulation paradigm allows one to gain several orders of magnitude of computation time compared to traditional discrete event simulation approaches. The accuracy of this simulation method is evaluated by comparison to other simulation methodologies. Examples of large scale network simulations are also presented.

Categories and Subject Descriptors

I.6 [**Simulation and Modeling**]: Model Validation and Analysis, Simulation Output Analysis

General Terms

Algorithms, Design, Performance.

Keywords

Discrete event simulation, congestion control, time-stepped hybrid simulation, TCP/UDP traffic, scheduling.

1. INTRODUCTION

Internet traffic and in particular TCP/IP controlled traffic is *reactive*, in that the amount of traffic sent by some source continuously adapts to the network conditions, namely to the number and the behaviour of the other sources competing for the network resources.

One can distinguish several classes of simulation methodologies for handling such reactive traffic: packet level–discrete event simulation, hybrid simulation and fluid simulation.

Packet level–discrete event simulation (such as that used in NS) is the most accurate but it suffers from serious scalability limitations. The fluid simulation approach (of e.g. [11, 7]) scales well but fails taking into account key network engineering features such as buffering or scheduling policies, as it does not allow one to accurately simulate or analyze *events*, be it packet level, session level or buffer level events.

Hybrid simulation uses both fluid simulation paradigms and discrete event simulation concepts. As we shall see with our simulation tool, NetScale, hybrid simulation offers a good compromise between accuracy and scalability and allows one to cope with the adaptive nature of Internet traffic.

Several hybrid approaches have been investigated in the past (e.g. [11, 6, 2, 1]). The hybrid simulation approach described in this paper was initially motivated by the models proposed in [1]. However, the incorporation of further ideas borrowed from [2, 6], makes it much more powerful than the initial vision. One of the key features of the approach, which is already present in [1] but absent from [2, 6], is the notion of flow aggregation, which relies on the definition of flowclasses and route pre-computation. Another important feature common to most hybrid methods is that events are gathered by time steps but not sorted within each time step. In contrast with what is done in [6], the proposed simulation methodology does not keep in memory all packet header information (which is computationally expensive) but only some partial information, such as the number of packets for each flowclass and for each time step. Indeed, flowclass information is enough to efficiently route packets in the network and simplifies the simulation of the router buffering/scheduling policies and that of TCP dynamic.

The main goal of the present paper is to discuss and illustrate the possibilities opened by this type of hybrid simulation.

2. ARCHITECTURE MODEL

2.1 Network model

The network is described by a set of nodes and links. We define two classes of nodes: a set $\mathcal{C}$ of core nodes and a set $\mathcal{T}$ of terminal nodes. Core nodes are meshed by a set $\mathcal{L}$ of links on which routing policies can be defined. These core nodes have instantiations as routers, switches, DSLAMs, Base Stations, RNCs etc. To each node or link, one can attach a scheduling or queue management policy. Terminal nodes represent access terminals and are directly attached to a core node. They have instantiations as servers, xDSL connections, mobile phones, Ethernet access terminals, WiFi access points etc.

2.2 Traffic model

The traffic is described at flow level by a set $\mathcal{F}$ of flows. A flow is characterized by two terminal nodes, a source node and a destination node, and its service class (SC). The service class is used, in particular, to map the Class of Service (CoS) in IP headers or the ATM traffic classes.

2.2.1 Application layer

We use a generic ON-OFF source model (cf. e.g. [1, 3]): an application is characterized by

- the statistics of the destination location,
- the direction of the transmission: download, upload or both, (e.g. download for HTTP session, upload for P2P passive session, both for bidirectional VoIP),
- the file size distribution for TCP transfers (the ON duration is not known a priori), or the ON duration and send rate parameters for applications over UDP (Voice or Video in CBR/VBR mode),
- the distribution of idle periods.

Libraries with the main usual applications (P2P, FTP, HTTP, video, VoIP...) have been developed based on statistical research results (e.g. for the typical file size distribution [10, 4]) and on specification standards (e.g. 3GPP).

2.2.2 *Transport layer*

As NetScale simulates all packet transmissions and receptions, no modeling part is necessary to integrate transport protocols. In particular, TCP reactiveness will be the consequence of the protocol specifications at the source and destination. Libraries with main transport protocols and variants (UDP, TCP Reno, NewReno, SACK, Tahoe...) have been implemented.

3. SIMULATION METHODOLOGY

3.1 Time stepped simulation

Time-stepped simulation consists in discretizing time into a fixed-length time step and in treating the bundle of packets falling into the time step without differentiating the detailed events. The time step is typically a few milliseconds. The reader should refer to [6] for a more detailed description of the time stepped approach and the classification of the different hybrid methods.

3.2 Flowclasses

As each terminal node is connected to a unique core node, we can consider the set of flows which have the same characteristics: source/destination core nodes, path in the core network and SC. We will refer to it as a flowclass. After each session establishment or routing change, the new path is computed according to core node routing tables and the flow is added to the corresponding flowclass. This scheme is compatible with advanced routing: it has been successfully used in [8] to study dynamic load-balancing mechanisms.

Defining flowclasses allows one to gather packets (and acks) and to route them all at once, saving computation time (compared to e.g. [6] which stores a list of all packet headers and computes the next hop for each packet). Packets are stored per flowclass in so-called buckets, simple elements holding the number of packets and losses. Buckets are routed from one core node to another along the flowclass path until they reach the destination core node. They are delayed along the way because of buffering and propagation.

When a bucket reaches a congested core node, it is added to the queue (cf. Section 3.4). If the buffer constraint is reached, some of its packets are lost (and its loss counter is incremented). Then, the queue outputs one bucket per flowclass. Each bucket is sent to the corresponding link, where it is stored during the propagation delay (rounded to the time step granularity, the rounding bias is added to next delay, to avoid cumulative rounding error).

3.3 Packet aggregation/de-aggregation

Packets generated flow by flow according to the application and transport layers (cf. Section 2.2) are enqueued at the source terminals and scheduled following the specification of each terminal type.

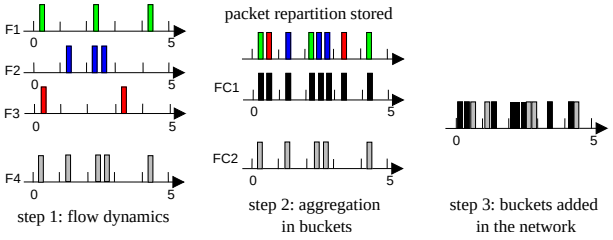

Figure 1: Packet aggregation

Figure 1 describes the way NetScale generates buckets in the core nodes at each time step:

- step 1: packets are dequeued flow by flow from the source terminals;
- step 2: packets from the same flowclass are gathered in a bucket. Moreover, the number of packets per flow for this time step is stored to correctly spread packets at destination nodes (see step 6);
- step 3: buckets are added in the first core node queue, then propagated along the flowclass route.

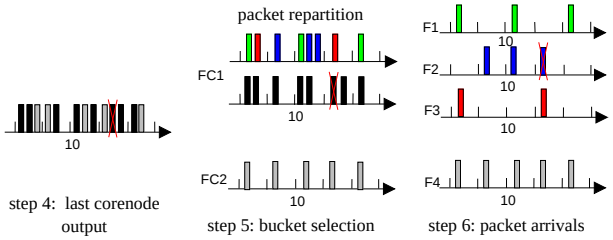

Figure 2: Packet repartition among flows

Once the last core node output is computed, we have to assign packets to their original flow. Figure 2 describes the way NetScale spreads packets:

- step 4: the last core node outputs buckets with the corresponding number of transmitted and lost packets;
- step 5: buckets which have reached their destination are removed from the core network;
- step 6: using the flowclass packet repartition stored in step 2, packets are assigned to their respective flows and scheduled in destination terminal nodes.

As explained above, NetScale does not store information on the individual packets in the core nodes. As a consequence, we need a way to estimate the arriving packet sequence numbers (e.g. to decide when to trigger duplicate acks for TCP): we keep the information on lost packets until destination (cf. Section 3.2). Hence, assuming that packet order is kept and combining the information on the number of packets and of losses in each time step, we can compute the sequence numbers of arriving packets.

We also need to accurately estimate round trip times at TCP sources: we store at step 2 (Figure 1) the packet transmission time (rounded to the time step granularity, one value per flowclass), so that we can measure the experienced round trip time when acknowledgments arrive at the sender.

3.4 Queueing dynamics: example of a FIFO queue

Each core node is characterized by the choice of a scheduling policy, a queueing management, a propagation delay and a routing table. For the sake of simplicity, we only consider FIFO queues here. However, the SC information at flow level can be used to simulate any flowclass differentiation policy such as weighted fair queueing, DiffServ or features like shaping, policing, Active Queue Management mechanisms (e.g. WRED). The extension to input or output buffered switches and e.g. maximal weight scheduling can also be integrated.

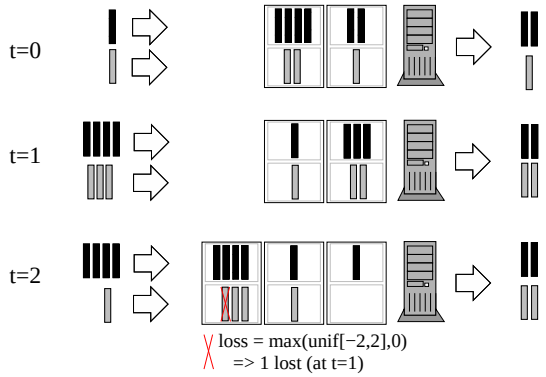

Figure 3: Simulation of a FIFO queue.

Figure 3 describes the NetScale modeling of a FIFO queue with a capacity D of 4 packets per timeslot and a buffer size B of 12 packets. Each time step, a new bucket for each flowclass is added to the queue size: $q' = q + A$, where q is the previous queue size and A the amount of packets in new buckets. After packet arrivals, we first check the buffer level. The number of lost packets is bounded by: $(q' - B)^+$ (all packets arrive at the beginning of the time step) and $(q' - B - D)^+$ (packets are spaced regularly). The number of losses depends on the packet inter-arrival law. If this law is known, we could pre-compute the corresponding empirical law of the number of losses. Here, we sampled a uniformly distributed random value between $(q' - B)$ and $(q' - B - D)$. The lost packets are then spread among flowclasses, with a probability proportional to the number of packets in the newly arrived buckets. Other models (e.g. cf. [6]) of loss repartition can be used as well. The presence of this modeling or measurement based element is the reason for which the simulation method described here is called hybrid.

Then, we serve the oldest buckets: packets are removed from the buckets and gathered in output buckets (again, buckets are separated by flowclass). For instance, at $t = 2$, buckets arrived at $t = -1$ and $t = 0$ are completely served and the fourth packet to be served is sampled randomly inside buckets arrived at time $t = 1$.

4. VALIDATION AND PERFORMANCE

To evaluate the accuracy of this approach, we first ran the whole set of scenarios of [5] for each TCP version. Figure 4 shows the outputs from NetScale for the case of TCP SACK with four packet losses. The time step was set to 10ms, so that the bottleneck router serves only one packet per time step. The packet dynamic matches very closely the one in [5], with an error on RTT bounded by the time step. The reactive nature of the TCP protocol is accurately captured by NetScale in all scenarios. With a longer time step, as the packet burstiness is not simulated inside one timeslot, the loss and input processes may be slightly different.

Time Step (ms)	ns2	1	2	4	10
AvgQueue (Mbit)	7.17	7.04	7.00	6.98	6.97
Loss Proba (%)	0.216	0.217	0.201	0.222	0.223
CPU Time Ratio	1	0.52	0.27	0.13	0.07
Memory (MB)	27	2.9	2.7	2.6	2.6

Table 1: Impact of the time step.

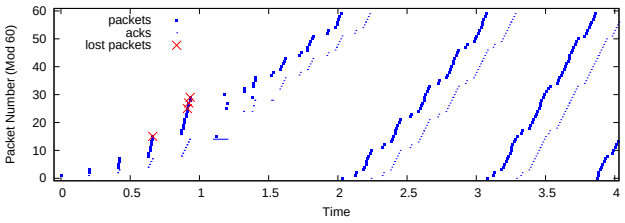

Figure 4: Sack TCP with four dropped packets

The impact of the time step is shown in Table 1 for a Dumbbell network with 100 on/off TCP SACK flows.[1] The file size (respectively off time) distribution is exponential with mean 20 Mbit (resp. 5s). The shared capacity and buffer size are 100Mbit/s and 1250 pkts with packet size of 1 kbytes and maximum window size of 60 pkts. The minimum round trip times (RTTmin) are sampled uniformly on [100, 200] ms. For the timeout probability, we noticed a significant difference (0.6 e^{-5} vs 1.2 e^{-5} for NetScale and ns2 resp.): as NetScale and ns2 results match closely for this probability when running TCP Reno (2.8 e^{-4} for NetScale vs 2.5 e^{-4} for ns2), we suspect that this difference comes from variants of the SACK protocol. Figure 5 plots the input rate density for ns2 and NetScale with different time steps. As predicted, the model accuracy increases for smaller time steps.

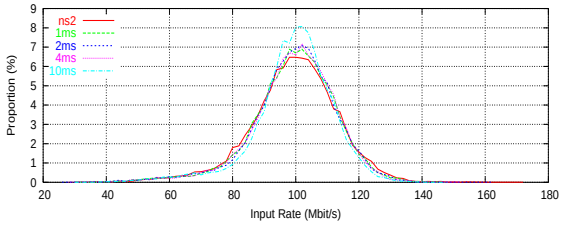

Figure 5: Input Rate density

To evaluate the benefits of NetScale, we compared its performance with that of ns2, for an increasing number of flows. The scenario is a Dumbbell network with N TCP persistent flows. The shared capacity is set to $2Mbit/s \times N$ and the buffer size to $25 \times N$ pkts. The RTTmin is equal to 0.1s. Figure 6 shows ns2 (thin lines) and NetScale (large lines) run times (red lines with crosses) and used memory sizes (blue lines with squares), as a function of the capacity, for a simulation time of 500s. The time step is 10ms.

For small networks, we can see that CPU time and memory size are 10 times lower with NetScale. However, the gain by flowclass aggregation, route pre-computation and time

[1]For each scenario, the average value of three simulations is presented.

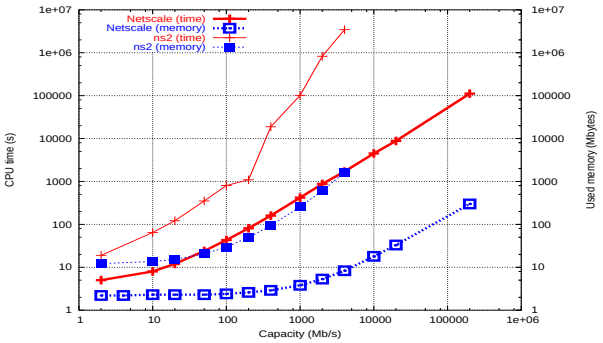

Figure 6: Run time and memory occupation.

stepped simulation, is more significant for large networks: CPU time increases linearly whereas it grows drastically for ns2. The gain factor for 1000 flows is 1000.

5. EXAMPLES OF LARGE NETWORK SIMULATIONS

NetScale has been also tested in various large heterogeneous network architectures. Simulation results obtained by NetScale on the European research network GEANT (Figure 7) topology are reported in [9]: the topology includes 19 core nodes and 60 gigabit links, on which several hundreds of thousands DSL access terminals have been connected and up to 15 millions of TCP sessions analyzed (hundreds Tbits of data for 1000s simulated).

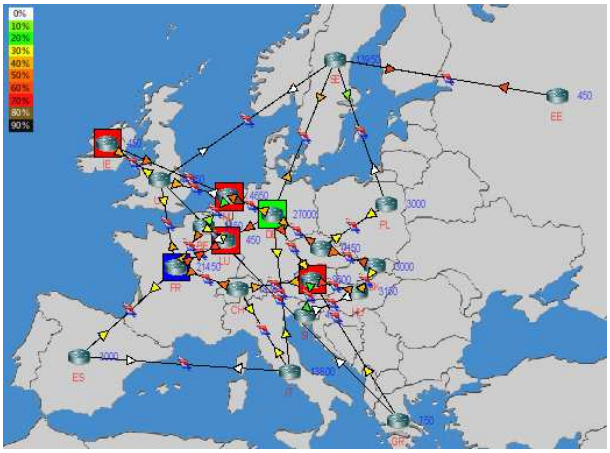

Figure 7: The GEANT (European Research Network) topology.

For this scenario, NetScale could both derive detailed end-to-end session level performance (goodput, losses, jitters...) and global link/network properties (utilization, availability...) in classical and dynamic routing settings. For more details, the reader should refer to [8, 9].

We also used NetScale for analyzing large tree-like DSL access networks with IP/ATM type DSLAMs. In such a context, TCP plays a key role, as access is most often the bottleneck, and NetScale allowed us to study the detailed service differentiation mechanisms present in such IP/ATM architectures.

It is also possible to mix the hybrid approach with pure packet level simulation: such a mixed approach has been validated e.g. for large UMTS/HSDPA networks (including thousands of base stations). These last issues will be the object of a future publication.

6. CONCLUSION

In this paper, we briefly discussed the advantages of combining time-stepped hybrid simulation and a flow aggregation based on a pre-computation of routes. We showed that this approach combines scalability and an excellent accuracy when compared to packet level discrete event simulation. This approach opens new possibilities for simulating, analyzing and understanding large and heterogeneous IP networks in a systematic and unified way.

Acknowledgments

The authors are very grateful to François Baccelli for his very valuable comments and suggestions. The authors wish to thank also Philippe Raoult and Max Unger for their generous contributions.

7. REFERENCES

[1] Baccelli, F., Hong, D. (2003) Flow Level Simulation of Large IP Networks. *Proc. of INFOCOM*, April.

[2] Bohacek, S. Hespanha, J. P., Lee, J. and Obraczka K. (2003) A Hybrid Systems Modeling Framework for Fast and Accurate Simulation of Data Communication Networks. *ACM SIGMETRICS*, June.

[3] Cao, J., Cleveland, W., Gao, Y., Jeffay, K., Smith, F.D., Weigle, M. (2004) Stochastic Models for Generating Synthetic HTTP Source Traffic. *Proc. of INFOCOM*, March.

[4] Crovella, M.E., Taqqu, M.S. and Bestavros, A. (1998) Heavy-Tailed Probability Distributions in the World Wide Web. *A Prac. Guide To Heavy Tails: Stat. Tech. and Appl.*, Birkhauser Verlag.

[5] Fall, K., Floyd, S. (1996) Simulation-based Comparisons of Tahoe, Reno, and SACK TCP *Computer Communications review*, July

[6] Guo, Y., Gong, W-B. and Towsley, D. (2000) Time-stepped Hybrid Simulation (TSHS) for Large Scale Networks. *Proc. INFOCOM*, pp. 441-450.

[7] Liu, Y., Presti, F.L., Misra, V., Towsley, D., Gu, Y. (2003) Fluid Models and Solutions for Large-Scale IP Networks *Proc. of ACM SIGMETRIC*.

[8] Randriamasy, S., Fournié, L., Hong, D. (2006) Distributed Multi-path and Multi-objective routing for network operation and dimensioning. *2nd EuroNGI Conf*, April.

[9] Randriamasy, S., Fournié, L., Hong, D. (2006) Distributed adaptive multi-criteria load balancing: analysis and end to end simulation. *INFOCOM Poster and Demo Session*, April.

[10] Willinger, W., Paxson, V. and Taqqu, M.S. (1998) Self-Similarity and Heavy Tails: Structural Modeling of Network Traffic. *A Prac. Guide to Heavy Tails*Birkhauser Verlag.

[11] Yan, A. and Gong, W-B. (1999) Time-Driven Fluid Simulation for High-Speed Networks. *IEEE Trans. on Inf. Theory*, vol 45, no. 5, July.