

Time Series Based Simulation Architecture

Tamás Máhr, Tímea Dreilinger, Attila Vidács
Dept. of Telecommunications and Telematics
Budapest University of Technology and Economics
H-1117 Magyar tudósok krt. 2., HUNGARY
{mahr, dreilinger, vidacs}@tnt-atm.tnt.bme.hu

Abstract

In this paper a new time series based simulation architecture is proposed. Nowadays as quality of service (QoS) issues (especially end-to-end issues) become more and more important, broadband inter-domain simulation scenarios are becoming hot topics. However, simulating backbone network traffic in a traditional packet based simulator is ineffective. Instead of dealing with individual packets, fluid flow simulators handle traffic flows that can dramatically lower the number of events in the simulator. Our proposed simulator architecture operates on an even higher level, on aggregate level. This simulator is especially designed to investigate inter-domain scenarios. The proposed architecture does not substitute but complement the classical ones.

1. Introduction

Simulation tools are instrumental in performance evaluation of telecommunication networks – they are much more effective from the cost and time perspective than the prototype implementation of a system. In recent past the size and the complexity of data communication networks have grown significantly. The widespread packet-based simulation tools spend too much time with state space maintenance caused by the huge amount of packets in the simulated network.

Many proposals have been made on behalf of speeding up the simulation. These methods can be assorted in three orthogonal types: increase the computational power, new simulation algorithms and higher-level abstraction models (see Figure 1).

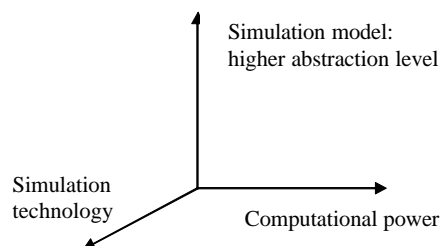


Figure 1 – Different methods to speed-up simulations

In the direction of computational power, the simulation can be faster using more powerful machines. Beside these, multiprocessors and distributed systems may provide better performance. However, adaptive traffic introduces such dependencies, which make difficult (or even impossible) to partition the network [1].

In the direction of simulation technology, new algorithms can speed-up the simulation. These special simulation techniques focus on optimising the state space maintenance or to explore important but rarely occurring events. (For example, the calendar queue and splay tree algorithm serve the purpose of improving the efficiency of event list manipulation [2]. Another method is the so-called RESTART mechanism [3], which explores rare event simulation.)

In the direction of simulation models, the higher-level abstraction simplifies the simulation and improves its efficiency. (Simulation technique model, such as the packet train, models a cluster of closely spaced packets as a single, so called “packet train” [4].) Nevertheless, the accuracy of performance measures is relaxed and the mapping to the physical world is more complex.

Another technique simplifying the simulation is the fluid model, which was first proposed by Anick *et al.* in [5] in order to model data network traffic. The fluid simulation paradigm is analogous to natural liquids; it operates on continuous amount of data rather than individual packet instances. The only difference is that telecommunication fluids cannot be mixed when they are multiplexed in a network queue, they remain isolated. The evenly spaced packets arriving close to each other may be modelled as a single fluid chunk with a constant fluid rate, with small time-scale variations.

A fluid simulator keeps track of the fluid rate changes at traffic sources and network queues, while an equivalent packet-level simulator would keep track of all packet instances in the network. The higher level of abstraction suggests that less processing might be needed to simulate the traffic. It is true only for simple fluid networks, where traffic flows do not compete for resources. (An example can be a link that connects two

nodes and never experiences queuing, this component can be modelled using only a constant propagation delay.) But whenever the flows need to share the available bandwidth due to congestion, the management of the limited resource can significantly increase the total processing required by the fluid simulator. In [6] the authors found that the fluid simulation can sometimes be less efficient than packet-level simulation. The explanation is fundamental from the point of view of the state-space maintenance of fluid simulation. When the bandwidth sharing of fluid flows changes due to congestion, the flow rates need to be propagated throughout the downstream queues. This way a single rate change may affect a large amount of flows, which is called as ripple-effect. A relationship can be formalised [7] for calculation the speed-up factor of fluid simulation with respect to the packet sending rate and the number of flows in case of on-off sources. As a conclusion, authors suggest to aggregate all flows, which are out of scope of the investigation to one background flow to decrease the number of update events. However, this aggregation is usually cannot be carried out due to disjoint routes in the network.

Going further on the ‘abstraction’ axis on Figure 1 we arrive to a simulation architecture that handles neither packets nor fluid flows but traffic aggregates, in addition it handles it in a large-scale discrete time space. It speeds up the simulation by evaluating the state space in large-scale (second, minute) discrete time points and by reducing the state space by handling only aggregate traffic.

The simulator considers only the border routers of each domain; more precisely in the simulator the border routers represent all the effect of the domain on an aggregate, through traffic. As a result the core routers of the domain are not represented in the simulator. In contrast to the packet or fluid flow simulators it does not schedule packets or flow rate changes, but time series. The input of the border routers is given by time series as well as the output.

The main advantage of this simulator architecture is that the simulation time does not depend on the amount of traffic in the simulated network. During simulation the input time series are processed by each border router, so the simulation time depends on the length of the time series, the number of border routers and the complexity of the border router models.

The paper is organised as follows. In section 2 the time series based simulation architecture is proposed. In section 2.3 a few models that are (or may be) used in the architecture are mentioned. In section 3 test results are shown and in section 4 the conclusion is drawn.

2. Simulation architecture

The simulation consists of networking elements. Each element behaves like UNIX filters: they get some input and they produce some output that can be fed to other elements as input. In this architecture inputs and outputs are time series describing the aggregate load on the particular element. The time series can be generated based on a traffic source models or it may be a result of some measurement done in a real network. The elements should provide the output based on the input, so that the result should represent the effect of the element on the aggregate traffic. Such elements can be concatenated – like UNIX filters –, the output of an element can be fed as input to another.

2.1 Network elements

From the architecture’s point of view it does not matter what the elements represent. They can be border routers, complete domains or just simple links. The complexity (and the runtime) of the simulation is larger if the elements represented a smaller part of the network but the accuracy is better. For example on Figure 2 the arrows are link elements modelling a link between the boxes. The boxes can be border nodes or domains in which case the links are intra-, or inter-domain links, respectively.

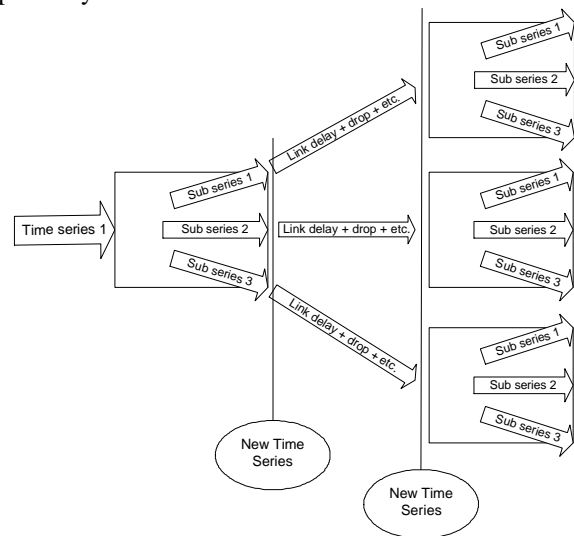


Figure 2 – General architecture

Each simulator element has a pluggable model that has to act on the same way on the input time series as the modelled entity. The models are the soul of each simulator element. They take the input time series of the corresponding element and produce the output time series according to what they model. From the

architecture's point of view again it does not matter what the models try to model or how accurate they work. The simulation can be run even if the models give a totally unacceptable result. The aim of the simulation defines always the type of the models that are to be used. Choosing the appropriate models is an important part of the configuration of the simulation scenario.

2.2 Processing the time series

The execution of the simulator differs from the usual way. It is divided into several runs. In one run it evaluates a whole time series (as input load) on an element. Next it takes the next element, feeds in the previous output as input and does the computation again. The selection of the elements to evaluate is important. Certainly only those elements can be evaluated that has every input available. This constraint results in the following conditions:

- There must not be any circle in the traffic or in the effect of the traffic (no feedback).
- An input must not depend on any output of the same element (even on an indirect way); otherwise it would cause a deadlock in the simulator.

This certainly does not hold if we want to model all individual router in a domain. The output traffic of a router will have a feedback effect on the input on the same link through the neighbouring router. But if we model only the border routers of a domain (interconnected to each other) then the inputs and outputs of the border routers are the inputs and the outputs of the domain. Then we can make that simplifying assumption that the input of the domain (on each border router) does not depend on the output of the domain. That is why on this lowest level of modelling each border router element will have not one but two models. One for the input traffic and another for the output traffic and they are independent of each other.

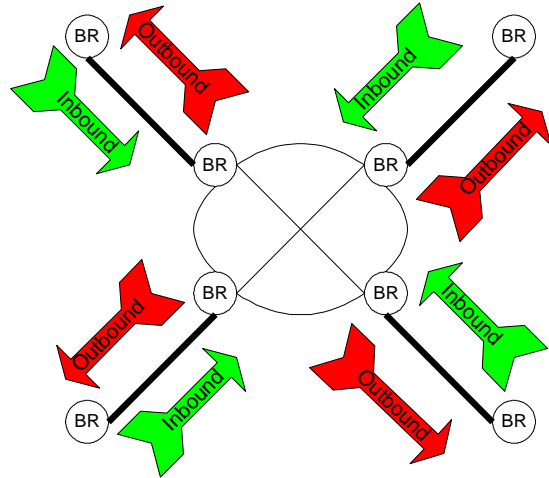


Figure 3 – Inbound and outbound traffic of a domain

If we are interested only in a path in the network, then it is obvious that we do not have to do the computation for all models of all elements but for only those that are on the path, and those that have some effect on the path.

The flowchart on Figure 4 describes how the simulator will execute a simulation scenario. First it creates the border nodes, group them into domains and defines the inter-domain links. Then it marks which direction of which inter-domain link we are interested in. Then it installs the models and sets up the input traffic on the network edges. As the last configuration step it determines the processing order of the border nodes. In the node execution loop it initializes the border node (the vexed model of the node), let it process its input, then propagates the output to the next nodes. It repeats the loop till unprocessed node exists and in the end it gathers the state information of the borders that are of interest.

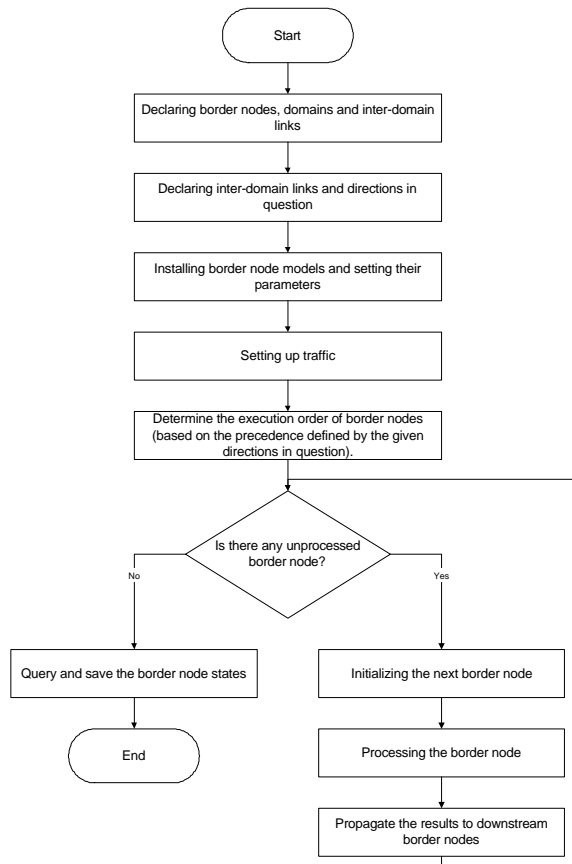


Figure 4 – Flowchart of the simulation

2.3 Models of nodes

Models are pluggable modules that process the input of simulator elements they are assigned to. There is a well defined interface between the element and the model through that the element can initialize the model, can have its input processed by the model and can query the results from the model.

Each model has the ability to distribute the output (in form of time series) among several output links according to a given distribution.

The most basic model is the ‘NullModel’ that has no other attributes. It simply copies the input to the output and distributes it if needed. A more sophisticated model (used in the test case) is the ‘LeakyBucket’ model. It models a router with a single server single buffer (FIFO) model. Any other (more sophisticated) models are imaginable while they conform to the defined interface.

In addition to computing the output load (throughput) in the time points given by the input time series the models may (should) produce some QoS state data (drop, delay, jitter) also in the time points given by the input

time series. This state information can be used in the end to evaluate the simulation scenario.

3. Comparing ns2 to time series simulator

3.1 Reference ns2 simulation

As a reference in evaluating the proposed architecture we used an ns2 simulation scenario. As shown in Figure 5 the scenario consisted of 9 access networks connected to one backbone with a single traffic sink. Since all traffic was directed to this single sink, output traffic could be measured only at border router 0. At every other border routers the traffic entered only into the backbone domain, there was no outgoing traffic at those points. Bottleneck links were the inter domain links between the domains (10 Mb/s) and the one to the sink (50Mb/s). The traffic was a compound traffic. One third of it was CBR and two thirds of two types of Pareto-type on-off sources with different parameters. During the simulation the load on the inter-domain links and the drops and the delay at border 0 were registered.

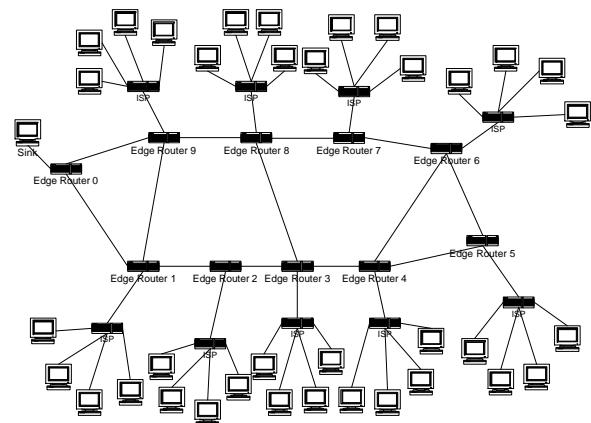


Figure 5 – Simulated topology

3.2 Time series based simulation

In the proposed time series based simulator architecture we modelled the backbone domain by its border routers. The bottleneck links were the same (inter-domain links) and for the intra-domain topology a full mesh virtual interconnection was simulated. We used the previously registered ns2 input load as input at routers 1-9 and compared the output load, the drop and the delay time series to the ones generated by ns2.

As the next figures show the output time series (Figure 6 and Figure 7) and the drop time series (Figure 8 and Figure 9) are almost the same in the two cases.

This means that if we are interested only in a high level, aggregate value of these QoS parameters (not in a per flow or per packet basis) then the proposed time series based simulator performed in accuracy as well as the classical packet based, but much faster.

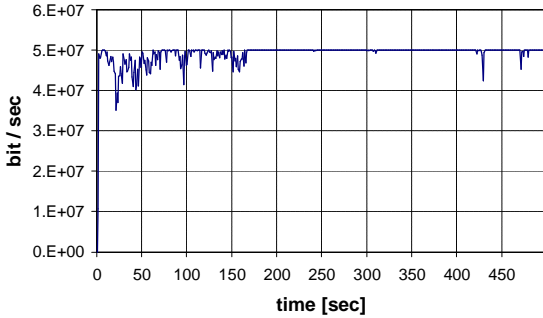


Figure 6 – Output time series of border node 0 in ns2

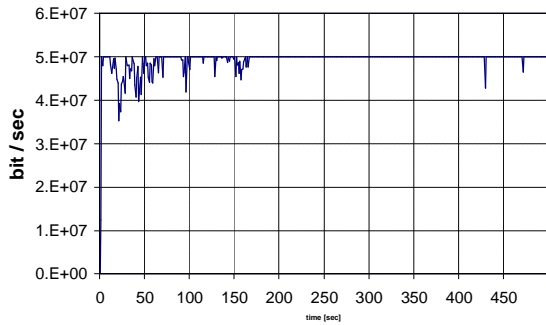


Figure 7 – Output time series of border node 0 in the proposed simulator architecture

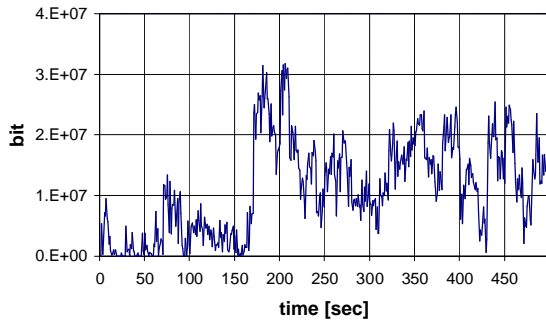


Figure 8 – Drops at border node 0 in ns2

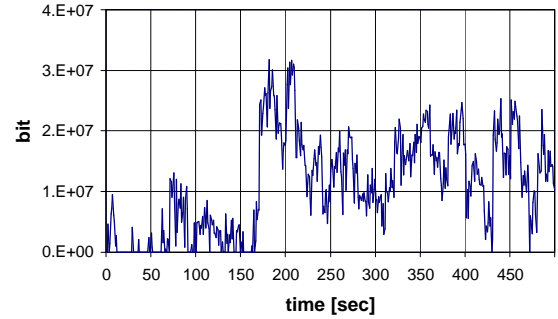


Figure 9 – Drops at border node 0 in the proposed simulation architecture

The delays (Figure 10 and Figure 11) registered in the two simulators are different, but they have the same characteristics. They vary heavily in the period in the first 170 seconds, and then they are around 14 msec till the 420 sec simulation time. Then there are two spikes at 425 sec and the 465 sec simulation time.

The models used in the border nodes in the time series based simulator cause the difference. The way they are implemented do not support the exact delay computation, they provide only approximate results. By using more sophisticated models these deviations can be corrected.

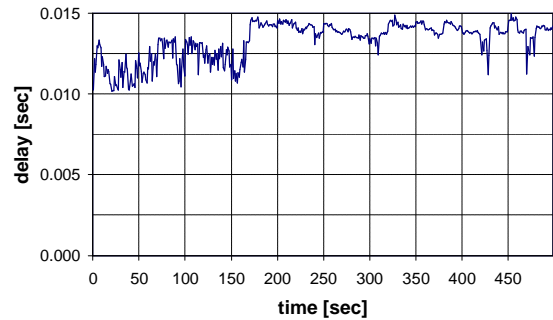


Figure 10 – Delay at border node 0 in ns2

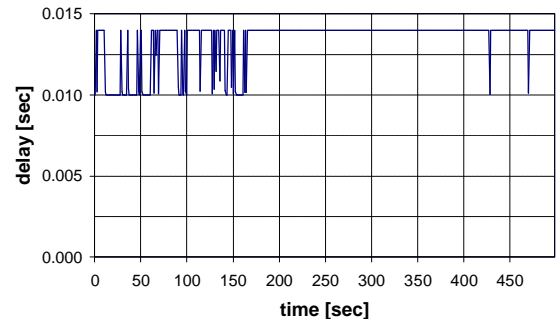


Figure 11 – Delay at border node 0 in the proposed architecture

4. Conclusion

Simulating inter-domain scenarios by classical, packet based network simulators (like ns2) is often not feasible due to the high number of events that should be handled by the scheduler of the simulator. Fluid flow simulators are on a higher abstraction level that means lower number of events.

The proposed time series based simulation architecture is on an even higher abstraction level, plus it evaluates the state space at large-scale discrete time points. This results in high level results (states of individual packets or flows cannot be determined) and short simulation runtime. As the tests showed the high level QoS values of the time series simulator match the ones generated by ns2. The runtime of the two architectures in inter-domain environment is not comparable, since the runtime of the time series based simulator does not depend on the amount of traffic, as it does in case of the packet-based architecture.

The proposed architecture does not substitute but complement the classical ones. On a higher abstraction level it performs better, so it should be used in an inter-domain scenario to discover the problematic parts of the network and then as zooming into smaller parts, the packet or fluid based approach may analyze the problem in more details.

5. References

- [1] P. Benko, "Accelerated Simulation of TCP/IP traffic using the modified fluid model", *M.Sc. Thesis*, Budapest University of Technology and Economics, Budapest, 1999.
- [2] R. Rögren, J. Riboe, R. Ayani, "A comparative study of some priority queues suitable for implementation of the pending event set", Dept. of Teleinformatics, Computer Systems Division, Royal Institute of Technology, Sweden, 1993.
- [3] M. Villén-Altaminrano and J. Villén-Altaminrano, "RESTART: a straight-forward method for simulation of rare events", in *Proc. of the 1994 Winter Simulation Conference*, Lake Buena Vista, Florida, USA, Dec, 1994, pp. 282-289.
- [4] J.S. Anh and P.B. Danzig, "Packet network simulation: speedup and accuracy versus timing granularity", *IEEE/ACM Transactions on Networking*, vol. 4, no. 5, pp. 743-757, Oct, 1982.
- [5] D. Anick, D. Mitra, M. M. Sondhi, "Stochastic theory of data handling system with multiple sources", *The Bell System Technical Journal*, col. 61, no. 8, pp. 1871-1897, Oct., 1982.
- [6] B. Liu, Y. Guo, J. Kurose, D. Towsley, W. Gong, "Fluid simulation of large scale networks: issues and tradeoffs", in *Proc. of PDPTA '99*, Las Vegas, NV, June, 1991, pp. 2136-2142.
- [7] B. Liu, D. R. Figueiredo, J. Kurose, D. Towsley, W. Gong, "A Study of network simulation efficiency: fluid simulation vs. packet-level simulation", in *Proc of INFOCOM 2001*, Anchorage Alaska, April 2001.