M.I.T. Laboratory for Computer Science

Request for Comments No. 252
July 26, 1983

Building a Congestion Control Simulator

by Lixia Zhang

This is a proposal for building a simulator to verify a proposed congestion control algorithm for DoD Internet Protocol, and to test its performance. The algorithm is described in [4].

1. Why Building This Simulator

Simulation is a powerful tool for testing new techniques in many cases, where experimental study will be uneconomical, difficult, or hazardous. It has been especially heavily used in the short history of computer communication networks, in design of new networks, in overall evaluation of network routing algorithms, and of network performance [1, 2, 3].

The congestion control algorithm proposed in [4] tries to attack the congestion problem from a new direction. Intuitively, it is sound. But we do not have experience with this kind of dynamic network control, nor do we have any concrete prediction on its performance. The algorithm need be thoroughly tested and proved before being implemented in networks. Doing an experimental test, i.e. to implement it on a test gateway as well as on a couple of local hosts and then to see how well it performs, is infeasible. Because although in this way the algorithm can be tested in a real network environment, the implementation of the testing code will be expensive. It also needs participation of several machines. More importantly, it can only test the algorithm in a fixed, confined physical environment, while we would like to verify the algorithm under several dramatically different internetwork environments.

Therefore simulation seems to be a good choice (and probably the only tool available) for our

testing. Although it surely takes time to write a good simulator code, it is not as expensive as the real implementation. The simulation does have a drawback that it is unable to precisely describe the real system, but we can always modify the simulation model to make it close enough for our testing purpose.

The recovery and correction of any problems prior to implementation of a new algorithm is of course a great advantage and would result in significant savings in time and effort. But we expect to gain more from the simulation than just exposing bugs in the algorithm design. Next section specifies the goals of the simulation.

2. Goals

Through simulating the proposed algorithm, we expect to achieve the following goals:

1. To test whether the algorithm is able to effectively control packet traffic and to avoid congestion in an internet environment.
2. To compare the network performance in the cases with and without the congestion control.
3. Under the situation that some hosts may not respond to the congestion control, test whether the gateway be able to deal with properly. (This is important because we cannot expect that everyone implements the congestion control on the same day, or would ever implement it, when it is decided to add this algorithm to IP.)
4. To evaluate the control overhead and its effect on the network performance under both light and heavy load conditions.
5. To help improve the congestion control algorithm. The proposed one by no means can possibly be the best[1]. We expect the simulation will help us gain further understanding of the inside world of the network and shed some light on the direction of our next step.
6. To help answer questions

   ~ What is the effect of each control parameter?

   ~ How should one choose and tune the control parameters of the algorithm to achieve required performance? (i.e. try to find some heuristic rules).

Building of the simulator is time consuming. In order to make the effort worthwhile, the

[1] For example, there is no timing information used in the current algorithm. Should a timer be used in measuring the traffic change tendency? We may be able to answer this question after the simulation.

simulator should also be extendable so that it can be used later on for studying other network problems.

3. What Have Been Learned from Previous Work

It is always wise and important to learn from previous experience when starting a new study. Conventional network simulations have played an important role in the network history and will continue to serve a useful purpose. The basic steps of those simulations are first to define a network topology, a statistic model of packet traffic pattern (mostly they model the input from a host as Poisson arrival with exponential packet length), a model of the algorithm under study, then to put all these together to run, and finally compute the accumulated statistical data as the simulation results.

There are a number of drawbacks in those simulations:

1. It is difficult to use this kind of models to simulate inter-related problems that must be solved in the design of specific network functions. For example, although a specific aspect of a routing algorithm could be analyzed by such a model, it would be a difficult task to simultaneously introduce, say, fluctuations in the traffic flow, outrages in network components, or line errors, etc..
2. Since the topology model is fixed, it is necessary to change the code to simulate different network environments.
3. The network traffic is fixed to a single (also must be simple) statistical model, which may not properly reflect the real network load.
4. Statistical evaluation is an averaging operator in nature. All the dynamic characteristics of the network operation will be averaged out in the statistical results. This is appropriate for capacity planning or new system design where questions concerning the average values of response time, throughput, resource utilization, etc., but it is not appropriate for the congestion control where the dynamic information is needed.
5. The resulting large quantities of data are time-consuming to understand and interpret.

In building up our simulation, we should try to avoid these problems. In particular, we should make effort on showing the transient state of the network, because a system can be controlled or tuned only if the usage of its resources and the characteristics of its work load are known *instant by instant.* We also want to make our model be able to simulate the real network traffic load closely, which requires, I think, far more than a simple Poisson model. In fact, due to the random, bursty,

and application-dependent nature of packet traffic, it is questionable whether we could ever have an attractable statistical model for it. To overcome these instantaneousness and modeling problems, I propose to use instant display of the simulation process and interactive simulation techniques, as described in the next section.

4. Things Needed for the Simulation

First of all we need a simulation model, then a program language to code the simulator, a machine to run the code, and a display screen with the support software.

4.1. Physical Resources

According to the available resources, the simulator will run on BORAX, use the BLINK protocol and ALTO screen for simulation display. Things are just available with no alternatives. The same is true for language selection: all display software is coded in CLU and so the simulator is better in CLU, too. Since we do not have simulation language available, CLU is probably the best choice anyway, because of its modular structure (and because I like to make this as a good exercise for my later CLU coding work).

One good point is that we do not worry much about the CPU cycles the simulation run will consume. So we will not pay much attention to the efficiency of the code or the length of runtime it needs.

4.2. Simulation Model

We need a model of the system under study, which includes network environment, traffic load, and the proposed control mechanism. This simulator model should meet the following requirements:

1. It should simulate the network environment correctly. It is not expected to have very precise description of the system, but it must take into account all necessary details related to our testing goals: communication delay, host response delay, gateway packet queueing delay, packet processing delay, control overhead, etc..
2. It should be able to dynamically change the network traffic load to closely simulate the real packet flows in the network.
3. It should be able to simulate occasional gateway failure/recovery, transmission errors, and other events that regularly occur in operational networks.

At the same time of meeting these requirements, The model must be simple, otherwise it would be intractable. It is always a good idea to start with a simple model which can later be extended by more sophisticated details when needed. We like to make the simulation as an iterative procedure, where initial models guide the first implementation; measurements of real characteristics then are fed back into refined models and so forth. This is important for improving the value of simulations. We use modular approach in coding the simulator so that later changes or extensions on parts will be easily done.

It is very difficult to characterize real world situations that occur in a network by mathematical models. We can use a random procedure to reasonably well generate error packets, but not to model traffic changes (how can we simulate TFTP and BLAST transfers by one model?). Often, it is the non-analytic nature of "natural" events that shows the most radical changes in the network operation and is the most difficult to deal with. One way to solve this problem is interactive simulation, using console to control and make changes during the simulation run. In this way we may make any event happen (gateway down and up, host sends and finishes, in burst or in any other pattern) at will and observe the reaction of the system.

The basic elements in an internet environment are gateways, networks, and hosts. We first define a CLU cluster for each of them, then we can create as many instances of them as we like, specify the characteristics of each[2], and connect them together in a desired topology. When the simulator starts running, it can be interrupted by the console commands at anytime. We may then examine the snapshot, the statistical data, or change the parameters of some host or gateway (e.g. change the sending rate of a host, or even decide to turn a gateway down, but no change of physical connections), and continue the simulation again with new parameter values to see their effects.

In order to see dynamic instances of the network operations and make the interactive simulation possible, I plan to expend the real time scale by 1000 times, i.e. using 1 second to simulate 1 milli-second in the real network (however, I'm afraid that this would be too slow). The simulator has a central clock, and an event list which contains the events from every host (sending or receiving packets), every network (delivering packets within some delay period), and every gateway. As the

---

[2]Take *host* as an example, we may specify a host's max packet sending rate, packet sending pattern, whether it supports the congestion control, and the statistical data we are interested in, etc.

clock runs, timeup events are removed form the list and processed, and new events are scheduled on the list and so on. We can expect to see the moving of packets on the screen, the growing and shrinking of packet queues at gateways.

The topology of the simulator is created at the beginning of each simulation run. How to model packet length distribution and host sending pattern still remains as two open questions. Since they are separate procedure modules in the code, this does not affect the mainframe run. We assume some simple model first, say constant packet length and Poisson arrival rate. It will be interesting to see whether the proposed control algorithm works under those mathematical traffic models. We will change them to or add some more realistic models we can think of later.

In building the simulator, we pay special attention to several important points:

- ~ The correctness of the model, this is the most important point.
- ~ Code modularity, extentability, and generality in terms of an inter-network environment.
- ~ Separation of the simulator framework from
    - * the congestion control strategy (so we can change the control algorithm easily and use the simulator for other purposes);
    - * the traffic modeling (because we like to do the test under different traffic models, may change the traffic model any time during the simulation run, and we are still looking for good modeling.)
- ~ A good simulation display. A picture is worth a thousand words, and many times intuition can help us understand the problem better than the numbers.

Routing issues are totally avoided in the current proposal.

There are two foreseen drawbacks in this simulator: it will probably run slowly (we have no idea what time scale expansion is proper, but we can easily change it at anytime), and the limited ALTO screen may not be able to show an internet environment of moderate size (Is it large enough to simulate the traffic on main branches of the proposed ATHENA project?).

5. First Outline of the Simulator

The first version of the code contains mainbones of the simulator, but it must be simple.

Basically, there are two processes in the simulator, one is console command interpreter, one is simulation process. The code starts with initialization of the internet topology and initialization of parameters of each network component (gateways, hosts, nets). Then the command interpreter will wait for new commands. When it receives a command, it interprets and passes to the simulation process. The simulation process starts by starting the clock running, then each host sends packets according to its transmission pattern. The connected network delivers the packets to gateways, and to the destinations (there may be more than one connections for each host). This process will be shown on the ALTO screen. At the same time, all the simulation conditions and events, such as the network topology, traffic rates, etc., as well as the statistical accumulations will be recorded. The whole simulation run will be controlled and terminated by the console.

The congestion control of the gateway will be computed and performed. The control tables will be shown on the display (but when there are several gateways in the network, the screen space will be a proble). We may make the control packets look different than data packets to tell their flowing. The algorithm is described in detail in [4], so we will not repeat it here.

We get the simulation result in two ways: interactive display, as well as the final printing of statistical data accumulated during the run. This will reduce the difficulties in the final analysis of the large quantities of data.

6. Conclusion

The proposed simulator has the following distinguished features:

- ~ Using screen to display instant situation. help our intuition in understanding the problem.
- ~ Slowing down the "real life" to see step by step "what is going on" in the network, (of course, assume that the simulator model is close enough, or we will gradually change it to be close enough, to the real world.)
- ~ Can stop at any instance to analyze the snapshot and then continue.
- ~ Interactive processing, can change parameters dynamically to simulate "real life" situations, such as damaged packets, load changes, gateway failure and recovery, ...

The simulation progress and final results will be reported in later group rfcs.

References

[1]

L. Kleinrock.
Analytic and Simulation Methods in Computer Network Design.
AFIPS Conference Proceedings, May, 1970.

[2]

S. Schoemaker, *Editor.*
Computer Networks and Simulation I.
North-Holland Publishing Company, 1978.

[3]

S. Schoemaker, *Editor.*
Computer Networks and Simulation II.
North-Holland Publishing Company, 1982.

[4]

Lixia Zhang.
Congestion Control at Internet Gateway.
Technical Report CSR-RFC 250, MIT-LCS, July, 1983.