Internet Engineering Task Force Gang Wang
Yong Xia
NEC Labs China
David Harrison
BitTorrent
Internet Draft
Intended status: Informational April 25, 2007
Expires: October 2007

An NS2 TCP Evaluation Tool
draft-irtf-tmrg-ns2-tcp-tool-00.txt

Status of this Memo

Copyright Notice

Abstract

This document introduces a TCP performance evaluation tool for the network simulator NS2. This tool is motivated by the observation that there is significant overlap among (but lack of an agreed set of) the topologies, traffic, and metrics used by many researchers in the evaluation of TCP alternatives: effort could be saved by

starting research from an existing framework. As such, our tool includes several typical topologies and traffic models; it measures some of the most important metrics commonly used in TCP evaluation; and it can automatically generate simulation statistics and graphs ready for inclusion in latex and html documents. The tool also contains an extendable open-source framework. With community effort, we hope the tool evolves into a widely accepted, well-defined set of TCP performance evaluation benchmarks.

Table of Contents

1. Introduction

Issues regarding the behavior of TCP in high-speed, long-distance networks have recently attracted wide attention. The Additive-Increase-Multiplicative-Decrease (AIMD) congestion control algorithm employed by TCP [1] is designed to share network resources among competing users fairly and efficiently. However, research results show that TCP does not perform very well in high-speed, long-distance networks [2]. To solve this problem, many TCP variants have been proposed. These include HighSpeed TCP (HSTCP) [3], Scalable TCP (STCP) [4], FAST [10], Binary Increase Control (BIC) TCP [5], CUBIC TCP [6], H-TCP [7], XCP [8], and VCP [9], etc. The proliferation of these many options logically brings the following question: what is the most effective high speed transport protocol for general use?

In order to answer this question, a number of performance evaluations for the high speed TCP variants have been conducted. These include simulations [11, 12] using the network simulator NS2 [13], test-bed analysis [14] and real world experiments [15]. Although test-beds and real world experiments can produce much more realistic results than simulations, implementing a large and complex experimental scenario is challenging. Therefore, many researchers employ NS2 simulations to test their ideas in the early stage of protocol design. Typically, each researcher writes his/her own NS2 scripts, and much time is spent duplicating similar network topologies, traffic models and performance metrics. On the other hand, the community still lacks a set of ready-made tests to reveal common flaws during early development, and a set of automated benchmarks to serve as a meaningful comparison between schemes proposed by different research groups. One might ask: why don't we avoid wheel reinvention and form a community project to this end?

This document introduces a project [22, 21] which implements an extendable tool that automates the NS2 TCP simulation process as much as possible. Using this tool, one just needs to simply define simulation scenarios (including network topology, traffic model and performance metrics); the simulation results (statistics and graphs) are automatically generated. The tool is very easy to use and extend. It can save a lot of time spent in writing NS2 simulation scripts. More important, we expect that, over the time, the scenarios collectively defined by the users of this tool will converge onto a set of community-acceptable TCP performance evaluation benchmarks, which will in turn be used by each individual researcher.

This tool includes a suite of NS2 simulation scripts that:

1. Automate simulation and post-processing procedures;

2. Define a set of commonly used network topologies, traffic models and performance evaluation metrics.

This tool can be used not only for high-speed TCP protocols, but for other proposed changes to congestion control mechanisms as well, such as ECN added to SYN/ACK packets, changes to make small transfers more robust, changes in RTO estimation, and proposals to distinguish between loss due to congestion or corruption, etc.

This simulation tool does not attempt to be final. Instead, it intends to serve as a starting point. We invite community members to contribute to the project by helping extend this tool toward a comprehensive set of NS2 TCP evaluation benchmarks.

```
 Network Topology        Traffic Model       Performance Metrics
+-----------------+    +------------------+    +------------------+
| Dumb-bell       |    |Long-lived FTP    |    |Network Level     |
| Parking-lot     |    |Short-Lived Web   |    |                  |
| Simple-network  |    |Streaming Video   |    |Application Level |
+-----------------+    |Interactive Voice |    +------------------+
        |              +------------------+              |
        |                       |                        |
        |                       |                        |
        |                       |                        |
        +------------------------------------------------+
                                |
                                |
                                V
                       +------------------+
                       |   Results in     |
                       |   Statistics     |
                       |   and Graphs     |
                       +------------------+
```

Figure 1 The architecture of our tool.

2. Tool Components

The architecture of our tool is shown in Fig. 1, which is primarily composed of the following components: network topologies, traffic models, performance evaluation metrics, and, after a simulation is done, a set of result statistics and graphs generated.

2.1. Network Topologies

The tool includes three commonly used topologies in TCP performance evaluations. They are single-bottleneck dumb-bell, multiple-bottleneck parking-lot, and a simple network topology. More realistic and complex topologies can be added to the tool easily.

2.1.1. A Single-Bottleneck Dumb-Bell Topology

This is shown in Fig. 2, in which source nodes and sink nodes connect to router 1 or router 2. The bandwidth between the two routers is much lower than the other links, which causes the link between the routers to be a bottleneck. (Traffic can be either uni-directional or bi-directional.)

```
   Src_1                                  Sink_1
        \                                /
         \          bottleneck          /
 Src_2 --- Router1 -------------- Router2 --- Sink_2
         /                              \
        /                                \
   Src_N                                  Sink_N
```

Figure 2 A dumb-bell topology.

2.1.2. A Multiple-Bottleneck Parking-Lot Topology

The parking-lot topology shown in Fig. 3 is similar to the dumb-bell topology except that it introduces cross traffic traversing the intermediate routers.

```
        Src_1    CrossSrc_1  CrossSrc_2  ...             Sink_1
             \       |           |                       /
              \      |           |                      /
      Src_2 --- Router1 --- Router2 ---   ...   RouterN --- Sink_2
              /                  |                 |    \
             /                   |                 |     \
        Src_N              CrossSink_1   ...  CrossSink_N  Sink_N
```

Figure 3 A parking-lot topology.

2.1.3. A Simple Network Topology

A simple network topology is illustrated in Fig. 4. In this configuration, the core routers represent the backbone of the network with the access routers responsible for sender or receiver nodes to connect to the network. It is similar to the transit and stub domains in GT-ITM [16]. Static routing is employed as the default routing protocol.

```
                              CR1
              PC1            /   \              PC5
                 \          /     \            /
                  --- AR2 --- CR2    CR4 --- AR4 ---
                 /          \     /            \
              PC2            \   /              PC6
                              CR3
                               |
                              / \
                           PC3   PC4

                       PC: Personal Computer
                       AR: Access Router
                       CR: Core Router
```

Figure 4 A simple network topology.

All the links in the above topologies have settable parameters such as link capacity, propagation delay, queue discipline, and so on.

2.2. Traffic Models

The tool attempts to apply the typical traffic settings. The applications involved include four common traffic types.

2.2.1. Long-lived FTP Traffic

FTP traffic uses infinite, non-stop file transmission, which begins at a random time and runs on the top of TCP. Implementation details and choice of TCP variants are decided by users, which is not in the scope of this tool.

2.2.2. Short-lived Web Traffic

The web traffic module employs the PackMime HTTP traffic generator [17], which is available in the recent NS2 releases.

2.2.3. Streaming Video Traffic

Streaming traffic is modeled using CBR traffic over UDP. Both sending rate and packet size are settable.

2.2.4. Interactive Voice Traffic

There are currently two synthetic voice traffic generation methods available in this tool. One is based on CBR-like streaming traffic. The other is generated according to a two-state ON/OFF model, in which ON and OFF states are exponentially distributed. The mean ON period is 1.0 sec, and the mean OFF duration is 1.35 sec. These values are set in accordance with ITU-T recommendations [18], but are changeable if needed.

The voice packet size is 200 bytes, including the 160 bytes data packet (codec G711, 64 kbps rate and 20 ms duration), 20 byte IP header, 8 byte UDP header, and 12 byte RTP header. These parameters can be changed by using other voice/audio codecs.

2.3. Performance Metrics

A comprehensive list of the metrics for TCP performance evaluation is described in [19]. In the first step, this tool tries to implement some commonly used metrics described in [19]. Here we follow [19] and classify the metrics into network metrics and application metrics. They are listed as follows.

2.3.1. Throughput, Delay, Jitter and Loss Rate

2.3.1.1. Throughput

For network metrics, we collect bottleneck link utilization as the aggregate link throughput.

Throughput is sometimes different from goodput, because goodput consists solely of useful transmitted traffic, where throughput may also include retransmitted traffic [19]. But users care more about the useful bits the network can provide. So the tool collects application level end-to-end goodput no matter what the transport protocol is employed.

For long-lived FTP traffic, it measures the transmitted traffic during some intervals in bits per second.

For short-lived web traffic, the PackMime HTTP model collects request/response goodput and response time to measure web traffic performance.

Voice and video traffic are different from above. Their performance is affected by packet delay, delay jitter and packet loss rate as well as goodput. So their goodput is measured in transmitted packet rate excluding lost packets and delayed packets in excess of a predefined delay threshold.

2.3.1.2. Delay

We use bottleneck queue size as an indication of queuing delay in bottlenecks. Besides mean and max/min queue size statistics, we also use percentile queue size to indicate the queue length during most of the time.

FTP traffic is not affected much by packet transmission delay.

For web traffic, we report on the response time, defined as the duration between the client's sending out requests and receiving the response from the server.

For streaming and interactive traffic, packet delay is a one-way measurement, as defined by the duration between sending and receiving at the end nodes.

2.3.1.3. Jitter

Delay jitter is quite important for delay sensitive traffic, such as voice and video. Large jitter requires much more buffer size at the receiver side and may cause high loss rates in strict delay requirements. We employ standard packet delay deviation to show jitter for interactive and streaming traffic.

2.3.1.4. Loss Rate

To obtain network statistics, we measure the bottleneck queue loss rate.

We do not collect loss rates for FTP and web traffic because they are less affected by this metric.

For interactive and streaming traffic, high packet loss rates result in the failure of the receiver to decode the packet. In this tool, they are measured during specified intervals. The received packet is considered lost if its delay is beyond a predefined threshold.

2.3.2. Response Times and Oscillations

One of the key concerns in the design of congestion control mechanisms has been the response time to sudden network changes. On the one hand, the mechanism should respond rapidly to changes in the network environment. On the other hand, it has to make sure changes are not too severe to ensure the stability of the network [19]. This tool is designed so the response time and fluctuations can be easily observed using a series of figures it generates, if the simulation scenarios we use include variable bandwidth, round trip delay, various traffic start times and other parameters.

2.3.3. Fairness and Convergence

In this tool, the fairness measurement uses Jain's fairness index [20] to measure the fair bandwidth share of end-to-end FTP flows that traverse the same route.

Convergence times are the time elapsed between multiple flows from an unfair share of link bandwidth to a fair state. They are quite important for environments with high-bandwidth, long-delay flows. This tool includes scenarios to test the convergence performance.

2.3.4. Robustness in Challenging Environments

A static link packet error model has been introduced in the tool to investigate TCP performance in challenging environments. Link failure, routing changes and other diagnostic markers can easily be tested by changing the tool's parameters.

2.4. Simulation Results

The tool includes a package [21] to automatically generate the above-discussed performance metrics. At the end of a simulation, it also automatically generates a series of user-defined statistics (e.g. bottleneck average utilization, bottleneck 90-percentile queue length, average per-flow goodput, etc.) and graphs (like bottleneck utilization and queue length variation over time, per-flow throughput over time, etc). It can create latex and html files in order to present the simulation results in a paper or webpage form. All the simulation-generated data is stored in a temporary directory for later use.

3. Usage Description

Here we present a brief description on how to use this software. For the details the reader is referred to the manual available at

[22]. The main body of this tool includes three files: create_topology.tcl, create_traffic.tcl, and create_graph.tcl. As their file names indicate, create_topology.tcl implements the three common network topologies discussed in Section 2.1, create_traffic.tcl defines the traffic model parameters in the simulation (see Section 2.2), and crate_graph.tcl generates simulation statistics (see Section 2.3) and plots graphs at the end of simulations. For the details on the graphs that can be generated, please refer to the manual available at [21].

Three example scripts are given in the package. Let's take the dumb-bell topology as an example. The simulation script is test_dumb_bell.tcl, and its default parameters are set in def_dumb_bell.tcl. Those parameters can be changed as needed.

Totally, the parameter setting includes three parts: topology setting traffic setting, and simulation statistics and graph setting. The topology setting defines the specific topology parameters. For dumb-bell, it sets the bottleneck bandwidth, round trip time, propagation delay, and packet error rate in the bottleneck link. The traffic setting defines the traffic parameters used in the simulation, such as the number of FTP traffic, what high-speed TCP protocol employed by FTP, using AQM or not, and how long the simulation runs. Finally, you choose the performance statistics to be generated (like bottleneck utilization, packet loss rate, etc.), and what kinds of figures will be displayed (e.g., queue length variation over time).

After running "ns test_dumb_bell.tcl", simulation statistics in text format and graphs in encapsulated postscript format are stored in the directory /tmp/expX, where X stands for the sequence number of the simulation. At the same time, all the simulation results can be viewed through a portal file /tmp/expX/index.html.

The parking-lot and network simulations are similar to the dumb-bell topology.

4. Security Considerations

There are no security considerations in this document.

5. IANA Considerations

There are no IANA considerations in this document.

6. Acknowledgments

The authors would like to thank Dr. Sally Floyd of ICIR for her encouragement and a lot of valuable advice. Part of David Harrison and Yong Xia's work was conducted when they were PhD students at Rensselaer Polytechnic Institute (RPI). They thank Prof. Shivkumar Kalyanaraman of RPI for his support and guidance.

7. Informative References

[1] V. Jacobson, "Congestion Avoidance and Control", SIGCOMM'88, August 1998.

[2] S. Floyd, S. Ratnasamy, and S. Shenker, "Modifying TCP's Congestion Control for High Speeds", http://www.icir.org/floyd/notes.html.

[3] S. Floyd, "HighSpeed TCP for Large Congestion Windows", RFC 3649, December 2003.

[4] T. Kelly, "Scalable TCP: Improving Performance in HighSpeed Wide Area Networks", ACM Computer Commun. Rev., vol.33, no.2, pp.83-91, April 2003.

[5] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks", IEEE INFOCOM'04, pp.2514-2524, March 2004.

[6] I. Rhee and L. Xu, "Cubic: A New TCP-Friendly High-Speed TCP Variant", PFLDnet 2005.

[7] D. Leigh and R. Shorten, "H-TCP Protocol for High-Speed Long Distance Networks", PFLDnet 2004.

[8] D. Kataabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks", SIGCOMM'02, August 2002.

[9] Y. Xia, L. Subramanian, Ion Stoica, and S. Kalyanaraman, "One More Bit is Enough", SIGCOMM'05, August 2005.

[10] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance", INFOCOM'04, March 2004.

[11] M. Nabeshima and K. Yata, "Performance Evaluation and Comparison of Transport Protocols for Fast Long-Distance Networks." IEICE Trans. Commun., vol.E89-B, no.4, April 2006.

[12] S. Mascolo and F. Vacirca, "The Effect of Reverse Traffic on the Performance of New TCP Congestion Control Algorithms", PFLDnet 2006.

[13] NS2. http://www.isi.edu/nsnam/ns.

[14] S. Ha, Y. Kim, L. Le, I. Rhee, and L. Xu, "A Step toward Realistic Performance Evaluation of High-Speed TCP Variants", PFLDnet 2006.

[15] H. Bullot, R.Les Cottrell, and R. Hughes-Jones, "Evaluation of advanced TCP stacks on fast long distance production networks", Journal of Grid Computing. vol.1, pp.345-359, 2003.

[16] GT-ITM. http://www.static.cc.gatech.edu/fac/Ellen.Zegura/graphs.html.

[17] J. Cao, W.S. Cleveland, Y. Gao, K. Jeffay, F.D. Smith, and M.C, Weigle, "Stochastic Models for Generating Synthetic HTTP Source Traffic", INFOCOM'04, March 2004.

[18] ITU-T Recommendation, "Artificial conversational speech", ITU-T, March 1993.

[19] S. Floyd, "Metrics for the Evaluation of Congestion Control Mechanisms", http://www.icir.org/tmrg/.

[20] R. Jain, "The Art of Computer Systems Performance Analysis", Wiley-Interscience, New York, NY, 1991.

[21] D. Harrison, RPI NS2 Graphing and Statistics Package, http://networks.ecse.rpi.edu/˜harrisod/graph.html

[22] G. Wang and Y. Xia, An NS2 TCP Evaluation Tool, http://labs.nec.com.cn/tcpeval.htm

Author's Addresses

Gang Wang
NEC Laboratories China
14/F, Bldg.A, Innovation Plaza, Tsinghua Science Park
No.1, Zhong Guan Chun East Road, Beijing, China
Phone: +86-10-6270-5962 ext 511
Email: wanggang@research.nec.com.cn

Yong Xia
NEC Laboratories China
14/F, Bldg.A, Innovation Plaza, Tsinghua Science Park
No.1, Zhong Guan Chun East Road, Beijing, China
Phone: +86-10-6270-5962 ext 320
Email: xiayong@research.nec.com.cn

David Harrison
BitTorrent
201 Mission Blvd Suite 900
San Francisco, CA 94105
Phone: +1-415-568-9000 ext 9012
Email: dave@bittorrent.com

Intellectual Property Statement

this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.