Measured Comparative Performance of TCP Stacks

Sam Jansen[1] and Anthony McGregor[1,2]

[1] Department of Computer Science, The University of Waikato
[2] National Laboratory for Applied Network Research (NLANR)*

Abstract. This extended abstract present findings on measured TCP performance of a range of network stacks. We have found that there are significant differences between the TCP implementations found in Linux, FreeBSD, OpenBSD and Windows XP.

1 Introduction

Implementations of a protocol vary in many respects, including how well they perform. There are several reasons for this variation. When implementing an Internet protocol, the programmer will refer to the protocol's specifications, for example its RFC. These specifications are normally written in English and may be ambiguous. Some aspects of the protocols behaviour may be left to the implementor. Even with a very tightly specified protocol, implementations do not always correctly meet the specification. This may be due to logic errors or misinterpretations of the specification. In some cases decisions are made that violate the specification to gain better performance.

In 1997 Paxson analysed TCP by writing a tool to automatically analyse a large amount of trace data he had available [1]. This was successful in finding implementation problems in a range of TCP variants of the time. However, the tool has a serious limitation: the code needs to be updated and hand crafted for every TCP implementation that is studied.

Previous work looking at TCP performance has looked at specific types of congestion control [2], sometimes under specific conditions such as a mobile ad hoc network [3], a lossy radio link [4] or ATM [5]. Studies comparing variants of TCP have also been performed, for example comparing New Reno and Vegas and Westwood+ [6]. Paxson's research involved TCP stacks from 1997: Solaris 2.4, NetBSD 1.0, Linux 1.0, Windows 95, Windows NT and others. TCP has evolved significantly in the past 7 years; this paper focuses on TCP implementations used in 2004 and 2005.

We hypothesise that todays TCP implementations will perform correctly under congestion regardless of their BSD lineage, in contrast to Paxson's findings. Further, we believe that TCP implementations have diversified sufficiently

* NLANR Measurement and Network Analysis Group (NLANR/MNA)is supported by the National Science Foundation (NSF) under cooperative agreement no. ANI-0129677.

that there are significant differences in measured performance between implementations, whether of BSD lineage or not. We use a test-bed network called the *WAND Emulation Network* which is described in the next section. Some measured TCP performance results are presented in Sect 3.

2 Emulation Network

The WAND Network Research Group has built a network of 24 machines dedicated to network testing. Machines are configured so there is a *control network* connecting the machines to the control machine and an *emulation network* which is configured by changing patch panels. Each machine has one Ethernet card connected to the control network, and one Ethernet card connected to the emulation network, which has four ports in the case of router machines. This allows arbitrary network topologies to be created between machines at a maximum speed of 100Mbit/s.

All machines are connected through one central switch to a control machine as well as having serial connections to the same machine. To simulate link delay and bandwidth limits, FreeBSD Dummynet [7] routers are used.

The control machine is able to install operating system images onto the machines on the emulation network in less than five minutes, making it possible to test a variety of different operating systems in a short time span. Images of Linux, FreeBSD, OpenBSD, Solaris and Windows XP are available.

It is possible to write scripts that run commands on the machines on the emulation network and send their output back to the control machine. This makes it possible to design, execute and record tests on the control machine.

3 TCP Performance

The following tests have been performed with the following operating systems: Linux, FreeBSD, OpenBSD and Windows XP with Service Pack 2.

3.1 Bidirectional Random Loss

This section presents a study of TCP performance under random loss in both directions; that is, both data and acknowledgement packets are dropped randomly using a uniform model. Random loss is interesting to study because Lakshman and Madhow [8] report that random loss is a simple model for transient congestion and of interest in the context of networks with multimedia traffic.

Figure 1 shows the topology used in this test. The bottleneck link is configured to have a propagation delay of 100ms and bandwidth limited at 2Mb/s. Router R1 drops packets randomly using Dummynet's *packet loss rate* option. The goodput over a single TCP stream from host H1 to H2 is measured. *Goodput* is the amount of data successfully read from the TCP socket by the application at the receiving end of a TCP connection. Hosts H3 and H4 are unused. Each test lasted 60 seconds.

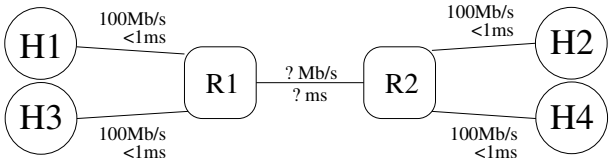

Fig. 1. Test network setup

Table 1. TCP performance during 5% bidirectional loss

TCP Implementation	Min	Mean	Max	SD
Linux 2.6.10	164.38	213.98	287.67	22.75
Linux 2.4.27	153.82	207.42	248.70	22.86
FreeBSD 5.3	136.77	176.20	225.01	17.11
FreeBSD 5.2.1	128.74	162.81	219.01	19.56
Windows XP SP2	89.90	137.31	191.00	21.67
OpenBSD 3.5	63.84	117.98	166.82	22.11

Table 1 shows recorded performance in kilobits per second under 5% random loss. The four numbers in the table are recorded goodput: minimum, mean, max and standard deviation. For each network stack, the test was run 100 times. All tests were run with kernel parameters with their defaults. Increasing the buffer sizes of any of the stacks studied made little difference, even though Windows XP defaults to only 8kB (compared to up to 64kB on other operating systems). While there is variation from run to run, it is small compared to the mean. Measurements with SACK turned on and off showed that SACK increases performance by just over 5% in this scenario.

3.2 Reverse Path Congestion

The test network topology is the same as presented in the previous section in Fig. 1. No artificial loss is added by routers R1 or R2 but host H4 sends data over a single TCP stream to host H3. The TCP stream from host H1 to host H2 is measured. The buffer sizes on routers R1 and R2 are set to 8 packets, the bottleneck link is set at 2Mb/s with 50ms delay. This allows R2 to be congested by the TCP stream from H4 to H3 which has the effect of congesting the acks of the measured TCP stream. H3 and H4 use Linux 2.4.27 while the operating system on H1 and H2 vary. The stacks are configured as in Sect. 3.1 apart from the size of the TCP socket buffers. The TCP socket buffer size for both receive and send buffers are set to 64kB for all network stacks in the test.

Table 2 shows the measured goodput at host H2 in kilobits per second. The variation between stacks is not as large as in the previous section but there is still a significant difference of 32% between the lowest and highest.

Table 2. TCP performance during reverse path congestion

TCP Implementation	Min	Mean	Max	SD
Linux 2.4.27	1220	1296	1375	33.3
FreeBSD 5.3	1128	1242	1366	52.8
FreeBSD 5.2.1	1099	1205	1289	48.6
Windows XP	906	1024	1152	58.5
OpenBSD 3.5	1273	1352	1438	40.4

4 Summary

This abstract shows that there is a large difference between the measured performance of the TCP stacks studied: Linux, FreeBSD, OpenBSD and Windows XP. During bidirectional random loss, the Linux TCP stack is able to obtain the most goodput by quite a long way. In this scenario OpenBSD is only able to achieve just over half the goodput that was measured with Linux 2.4 and 2.6 kernels while Windows XP achieves just 64% of the goodput measured with Linux. Windows XP is additionally limited by its default TCP window sizes, which are very small by today's standards.

Further analysis of these results is not presented because of the lack of space available.

References

1. Paxson, V.: Automated packet trace analysis of TCP implementations. In: SIGCOMM. (1997) 167–179
2. Fall, K., Floyd, S.: Comparison of Tahoe, Reno and SACK TCP (1995)
3. Holland, G., Vaidya, N.H.: Analysis of TCP performance over mobile ad hoc networks. In: Mobile Computing and Networking. (1999) 219–230
4. Kumar, A.: Comparative performance analysis of versions of TCP in a local network with a lossy link. IEEE/ACM Transactions on Networking **6** (1998) 485–498
5. Comer, D., Lin, J.: TCP buffering and performance over an ATM network (1995)
6. Grieco, L.A., Mascolo, S.: Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control. SIGCOMM Comput. Commun. Rev. **34** (2004) 25–38
7. Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols. ACM Computer Communication Review **27** (1997) 31–41
8. Lakshman, T.V., Madhow, U.: The performance of TCP/IP for networks with high bandwidth-delay products and random loss. IEEE/ACM Transactions on Networking **5** (1997)