УДК 004.002
Melnyk V.M.
Lutsk National Technical University

FULL PERFORMANCE EFFECTS FOR PARALLEL TCP SOCKETS ON A WIDE-AREA NETWORK

Melnyk V.M. Full Performance Effects for Parallel TCP Sockets on a Wide-Area Network. This work describes the theoretical effects of using parallel TCP flows to improve network performance for intensive applications with distributed data. The theory knowledge were presented for a widearea network to assess how parallel flows improve throughput, and took clearance in choossng the necessary number of flows to improve throughput with avoiding congestion. An empirical expression and the uses of parallel flows are discussed.
KEYWORDS: Network, Full Performance Effects, Parallel TCP connections, sockets, Wide-Area Network.
Fig. 6., Ref. 32.

Мельник В.М. Ефекти повного відтворення для паралельних TCP сокетів, реалізованих на широкомасштабній лінії зв'язку. Дана праця описує теоретичні ефекти використання TCP-потоків для покращення відтворення зв'язку між інтенсивними додатками з розподіленими даними. Теоретичні дані продемонстровані, щоб встановити як паралельні потоки покращують мережеві з'єднання і підтвердити прозорість в виборі необхідного потокового номера для того, щоб покращити мережеве з'єднання для широкомасштабного зв'язку минаючи переповнення. Обговорюється емпіричне рівняння і використання паралельних потоків.
Ключові слова: Мережа, Ефекти повного відтворення, паралельний TCP зв'язок, сокети, широкомасштабна мережа.
Рис. 6., Літ. 32.

Мельник В.М. Эффекты полного воспроизведения для паралельных TCP сокетов, реализованных на широкомасштабной линии связи. Даная работа описывает теоретические эффекты использования TCP-потоков для улучшения воспроизведениия связи между интенсивными приложениями с распределенными данными. Теоретические данные продемонстрированны, чтобы установить как параллельные потоки улучшают сетевые соединения и подтвердить ясность у выборе необходимого потокового номера для того, чтобы улучить сетевое соединение для широкомасштабной связи минуя переполнения. Обговаривается емпирическое уравнение и использование паралельных потоков.
Ключевые слова: Сеть, эффекты полного воспроизведения, параллельная TCP связь, сокеты, широкомасштабная сеть.
Рис. 6., Лит. 32.

Introduction

To improve end-to-end network performance for applications that require substantial amounts of network bandwidth there are considerable efforts within the Grid and high performance computing communities. An experience submited in [1, 2] has shown than the end-to-end structural and load characteristics are available for a network. One source of poor TCP throughput is rate of a packet loss. It can be much greater than it would be reasonably expected [3]. The packet loss is interpreted by TCP as a network congestion proof between a sender and receiver. However, packet loss may be such as intermittent hardware faults due to other factors than network congestion [4].

To improve end-to-end performance the current efforts take advantage of the empirically discovered striping data transfer mechanisms across a set of parallel TCP connections to substantially increase throughput. So, application developers and network engineers must have an understanding of how parallel TCP connections improve summative throughput as well as the effects on a network. This article puts several questions concerning of the parallel TCP connections use. The question number one is how the use of parallel TCP connections increases aggregate throughput. The second one is how to determine the TCP connections number needed to increase and maximize throughput while avoiding network congestion. And the last one, it need to understand how parallel TCP connections affect a network, and under what conditions they should not be used. This paper suggests some guidelines for the parallel sockets use to maximize end-to-end applications performance whith simultaneously minimizing their network effects.

Current Work

Applications usually use one of two approaches to improve end-to-end network throughput that effectively defeats the congestion avoidance behavior of TCP. The first approach utilizes UDP, which puts responsibility for both error recovery and congestion control completely in the application hands. The second approach opens network connections of parallel TCP and "stripes" the data (similar to RAID)

© Мельник В.М.

across a parallel set of sockets. The above two approaches are hostile and don't permit the light sharing of the network bandwidth available to applications [5].

Recent works [1, 2, 6] has concidered that the parallel socket approach greatly increases the aggregate network throughput available to an application, but some of them report [6] that the speedup is not reliable. The working process may be organized to address the issues of poor network performance and the unpredictable end-to-end network bandwidth availability. To address unpredictability, the Network Weather Service project [7] is working to predict the network bandwidth available between two sites on the Internet based on statistical forecasting. Efforts to address poor network performance include Bandwidth Brokering [8], Quality of Service (QoS) Reservation [9], and network and application tuning pains [4, 6].

The work devoted to the parallel TCP connections use is fundamentally experiential in nature and useful for an application perspective. The works [10, 11] describe about the transfer rate increase of medical images over the Internet. In work [12] is described done to increase the TCP throughput over satellite links. In [2] is developed a library (PSockets) to stripe transmissions of the data over multiple TCP network connections to deliver significantly increased performance on a poorly tuned host, what can be compared to the performance through a single TCP stream. Dimensions using the PSockets library for striping network I/O established that the use of 12 TCP connections increased TCP performance from 10 Mbit/sec to about 75 Mbit/sec. In works [13] and [14] authors have both developed modifications to TCP that take gain of the positive effects for parallel sockets of TCP. In the work [1] author provides an argument that explains why network performance is improved over multiple TCP streams compared with a single TCP stream. In [15] has created an extensive measurement infrastructure by the Stanford Linear Accelerator network research group to measure the multiple TCP connections effect between input Internet sites for the Atlas project.

This time alot of applications are using or planning to use parallel TCP connections to increase aggregate TCP throughput. The ever-present example of this thing is the Netscape browser. It uses by its clients a determined value of four for the number of parallel TCP connections [16]. The Grid FTP project allows the user to select the parallel TCP connections number to use for FTP data transfer [17]. Storage Resource Broker (SRB) has provisions to use multiple TCP sockets to improve SRB data transfer throughput [18]. The Internet-2 Distributed Storage Initiative in [19] is investigating the parallel TCP connections use to improve the distributed data caches performance. All of this current work has investigated the effects of parallel TCP connections from an experiential perspective. Researchers of the Stanford Linear Accelerator group have found that the parallel TCP connections number ranges from 4 (Netscape) to 12 (PSockets) to a number between 4 and 20 depending on the window size.

On the overall, the effects of using multiple network sockets fairness and efficiency of the network have been raised [5, 19, 13]. The traffic shaping mechanism and limit rate [20, 21] have been proposed and implemented to effort to prevent aggressive users from using more than their fair share of the network. Despite the demonstrated effectiveness the little work has been done to develop a theoretical model to validate the use of these optimal values. The models would help us understand the underlying mechanisms that allow parallel TCP connections to deliver extremely increased performance; the effects of using parallel sockets on the network fairness and efficiency; and under what conditions and circumstances should be used parallel sockets. On the next section a theoretical model of parallel TCP connections will be developped. It also explain how they take advantage of systemic noncongestion packet loss to improve aggregate throughput.

TCP Bandwidth Estimation Model

Some research works have resulted theoretical expressions to calculate TCP bandwidth of the single stream as a function of packet loss, round track time, size of a maximum segment, along with a handful of other various parameters. In [22] there are performed a detailed analysis of three techniques and assessed their ability to precisely estimate TCP bandwidth across a wide range of packet losses. The most correct model is described in [23] where an approximation of the subsequent form (1) follows:

$$TCP_{BW}(p) = \min\left(\frac{W_{max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1,3\sqrt{\frac{3bp}{8}}\right)p(1+32p^2)}\right)MSS \quad (1)$$

It is received from the original form to match the scale of Equation (2) by adding MSS. In this equation, $TCP_{BW}(p)$ shows bytes transmission per second. MSS is the maximum segment size, W_{max} is the maximum congestion window size, RTT is the round trip time, *b* is the number of transmitted data

© Мельник В.М.

packets that is acknowledged by one acknowledgement from the receiver (b = 2), T_0 is the timeout value and *p* is the packet loss, which is the number of retransmitted packets divided by the total number of transmitted packets.

It was found that the equation in [24] is essentially as accurate for packet loss rates less than 1/100 as equation (1), and it has a much simpler form:

$$BW < \frac{MSS \ C}{RTT \sqrt{p}} \qquad (2)$$

where *p*, *MSS* and *RTT* are the same variables used in (1) equation. C is a constant and BW is the number of bytes transmitted per second. To understand the leading mechanisms of TCP throughput, it is useful to consider the dynamic behavior of *MSS*, *RTT* and *p* and the result each has on overall TCP bandwidth., MSS is the most static amon these three factors. If both TCP session sides have MTU discovery enabled [26] within the host operating system, both sides will attempt to negotiate the largest possible maximum transmission unit (and as a result MSS) possible for the session. The MSS setting depends on the structural characteristics of the network, host adapters and operating system. Too often, the common maximum MTU supported by networks and network adapters is 1500 bytes. In some cases the data link layers of routers and switches making up the end-toend network will support larger frame sizes. If the TCP connection MTU can be increased from 1500 bytes to 9000 bytes, then equation (2) right hand side increases by a factor 6, that increasing a maximum TCP bandwidth by a factor of 6 as well.

The RTT value is more dynamic during a session than MSS, but less dynamic than *p*. The lower bound on the RTT value is the signal transmission speed from host to host across the network, which is essentially limited by the light speed. As the end-to-end network path length increases, the introduction of routers and framing protocols on the physical links between the two hosts adds latency to the RTT factor, and other factors involved with queuing and, as well, congestion can increase RTT. From an end host perspective, however, to substantially improve RTT there is little that can be done. The final factor *p* (packet loss rate) is the most dynamic parameter in conditions to MSS, RTT and p. The avoidance algorithm of the TCP congestion in [27] interprets packet loss as an indication of the network congestion and the sender should decrease its transmission rate. The packet loss rate *p* spans many orders of magnitude and represents a significant contribution to variability in end-to-end TCP performance in the operational Internet. It's important to note that the packet loss rate has been observed to fall into two regimes: packet loss due to network congestion and traffic insensitive packet loss.

On the next we will present the expression derivation for aggregate TCP bandwidth, describe some of the packet loss characteristics on the Internet and explain how these characteristics affects the single and multi stream TCP sessions performance.

If an application uses *n*-multiple TCP streams between two hosts, the aggregate bandwidth of all *n* TCP connections can be derived from equation (2), where MSS_i, RTT_i and p_i represent the related parameters for each TCP connection i:

$$BW_{agg} \leq C \left[\frac{MSS_1}{RTT_1 \sqrt{p_1}} + \cdots + \frac{MSS_n}{RTT_n \sqrt{p_n}} \right] \qquad (3)$$

As MSS is determined on a system wide level by a network architecture and MTU discovery combination, it is reasonable to assume that each MSS_i value is identical and constant across all betweenhosts simultaneous TCP connections.

It can be assumed that RTT is equivalent across all TCP connections, since every packet for each of them will likely take the same converge to equilibrium and network path. Note that since the TCP congestion avoidance algorithm is an equilibrium process that seeks to balance all TCP streams to fairly share network bottleneck bandwidth [28]. Each stream must either respond to changes in the packet loss rate, RTT, or a combination of both of them to converge to equilibrium. Since all streams on a set of parallel TCP connections are between two hosts, all the streams should converge to equivalent RTT values, as long as the network between the hosts remains uncongested. For purposes of this discussion, *C* can be set aside. So, equation (3) can be modified to:

$$BW_{agg} \leq C \frac{1}{RTT} \left[\frac{MSS}{\sqrt{p_1}} + \cdots + \frac{MSS}{\sqrt{p_n}} \right] \qquad (4)$$

On equation (4) examination some parallel TCP connections features become apparent. An application opening *n* multiple TCP connections is essencialy creating a large virtual MSS on the aggregate connection that is *n* times the MSS of a single connection. Factoring MSS out of equation (4) produces:

© Мельник В.М.

$$BW_{agg} < \frac{MSS}{RTT}\left[\frac{1}{\sqrt{p_1}} + \cdots + \frac{1}{\sqrt{p_n}}\right] \quad (5)$$

It becomes apparent that given the relatively static nature of the MSS and RTT values compared with the dynamic nature of *p*. The packet loss rate *p* is a primary factor in determining aggregate TCP throughput for session of a parallel TCP connection.

From equation (4) it also is apparent that the increased virtual MSS of parallel TCP connections is directly affected by the packet loss rate *p* and RTT of each connection. RTT has hard lower bounds. They are structural and difficult to address. On the other hand packet loss *p* is the parameter that is most sensitive to network load. It can be affected by several factors.

It is observed that packet loss falls down into two characteristic cases: random losses which do not due to congestion and congestion related losses. In [25] is found that packet losses tend to occur at random intervals in multiple packets bursts, rather than single packet drops, and In [29] is found bursty packet loss behavior as well. Additionally, the event of the chance of a packet loss increases when packets are in intermediate hops queue and the network becomes loaded. In [3] is found that packet loss demonstrates random characteristics when the stream uses a fraction of the bandwidth of available network.

As there is increase of the multiple TCP connections number, the behavior of each packet loss factor *p* is unaffected as long as few packets are queued in routers or switches at each hop in the path of the network. In the lack of congestion, it is appropriate to assume that the packet loss proportion will be fairly distributed across all connections. So, when the aggregate packet stream begins to create the congestion, any router or switch may begin to drop packets. The packet loss which is attributable to each TCP stream will be depended of the queuing discipline, and of any phase effects caused by TCP senders sharing a network like bottleneck [30].

So, when congestion occurs there are four exceptions to the assumption that packet loss is fairly distributed. It was been empirically determined in works [31] that three the pathological conditions are availeble. One state, lockout, occurs when in a router one stream dominates the queue. The second state, usualy drop-tailed queues are arising when queuing algorithms unfairly aim a number of flows through the queue with rates of excessive packet loss for newly arriving packets. The third state produces transmission time distributions of the heavy-tailed data due to the rates of the congestion and high packet loss [32]. Finally, in work [30] in found that the convergence of multiple TCP streams at a congested blockage can create phase effects in which one stream unfairly dominates the queue and therefore the outbound link.

The unfair packet loss distribution is an undesirable condition in congested routers [21]. To fairly distribute packet loss with providing mechanisms in routers there were designed and deployed new queuing schemes, such as Random Early Detection (RED) [21]. For this analysis it is assumed that packet loss impacts equally parallel TCP streams. The next example illustrates the multiple TCP streams impact in an uncongested network: if it is assumed that MSS = 4418 bytes, RTT = 70ms, and pi=1/10000 for all connections, and it is used

$$K = \frac{MSS(4418\ bytes)}{RTT(70m\ sec)}\left(\frac{\frac{8bits}{byte}}{\frac{100000bits}{Mbyte}}\right)\left(\frac{100m\ sec}{sec}\right) \approx \frac{1}{2} \quad (6)$$

the upper bound on aggregate TCP bandwidth may be calculated by using (5) equation. Table 1 showes the calculation results for a following sockets number.

Connections	$\sum_n \frac{1}{p_i}$	**Max. Aggrigate Bundwith**
1	100	50 Mb/sec
2	100+100	100 Mb/sec
3	100+100+100	150 Mb/sec
4	4(100)	200 Mb/sec
5	5(100)	250 Mb/sec

© Мельник В.М.

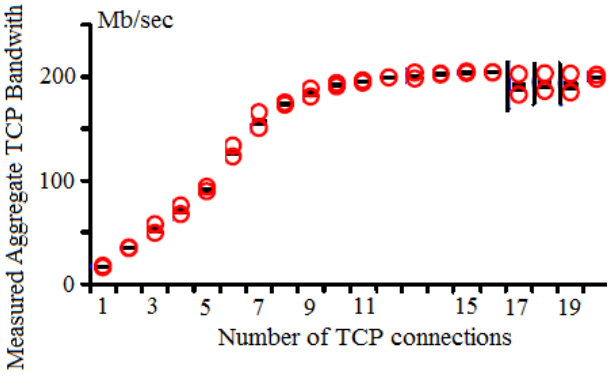


Fig. 3 – Throughput of Parallel TCP Sockets with MSS of 1448 Bytes

Table 1 – Packet Loss on Aggregate TCP Bandwidth

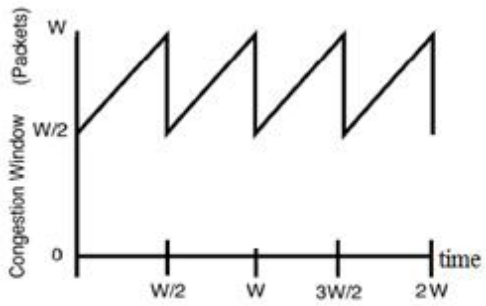
At present, as the aggregate network utilization increases to the point where queues and buffers in switches and routers begin to be overflowed and packets are dropped, then the network becomes congested. If the packet loss is fairly shared over all of the connections through a switch or router due to congestion, the negative packet loss effects on the aggregate TCP bandwidth for a set of *n* simultaneous connections is overstated by *n* factor. So, if the packet loss rate from the previous example doubly increases, the multiplicative packet loss rate factor in Table 1 is reducing from 100 to 70.7. For five concurrent streams, this reduces aggregate bandwidth from 250 to 176.8 Mb/sec. That counts a reduction of 30%. Even with this reduction, however, the aggregate bandwidth of 176.8 Mb/sec using five parallel TCP connections is still considerably better than the throughput with only one connection at the rate of the desirable packet loss.

It is some difficult to predict at what point the packet loss will become congestion dependent as the number of parallel TCP connections increase. There is, however, a definite bend in the graph curve of packet loss what suggest that adding additional network sockets beyond a certain entrance will not get better aggregate TCP performance. An examination of Figs 1, 2 and 3 indicates that for a MTU of 1500 bytes, 10 sockets is the effective maximum number of sockets; for a MTU of 3000 bytes, 5 sockets is the effective maximum; and for a MTU of 4418 bytes, 3 or 4 sockets is the effective maximum. The effective maximum presented in Figure 3 (MTU 1500) roughly corresponds to the results of Sivakumar [2], who found that the point of maximum throughput was 16 sockets or less. Sivakumar did not mention the MTU used in [2], but if the default system settings or MTU discovery were used on the system, the MTU used was probably less than or equal to 1500 bytes.

Why Parallel Sockets Work

Using of parallel TCP sockets would improve aggregate throughput, since one would expect that a network would make a best effort for maximization throughput through a single stream. There are sources of traffic insensitive packet loss that are not due to congestion. In this random packet loss regime, the parallel TCP connection use allows an application to ease the negative effects of the packet loss misinterpretation by the control algorithm of the TCP congestion. It need to give an explanation of why using parallel TCP connections increases aggregate throughput.

The derivation of equation 2 in work [24] uses a geometric argument with constant probability packet loss rate 1/p=(W/2)2+1/2(W/2)2, where W is the congestion window size in packets. When a loss event appears every 1/p packets,



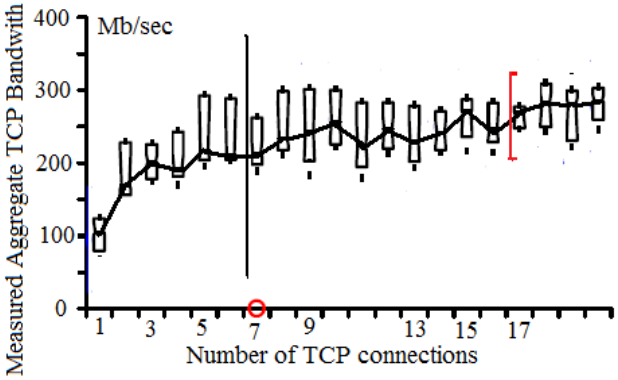


Fig. 1 – Throughput of paralel TCP Sockets with MSS of 4366 Bytes

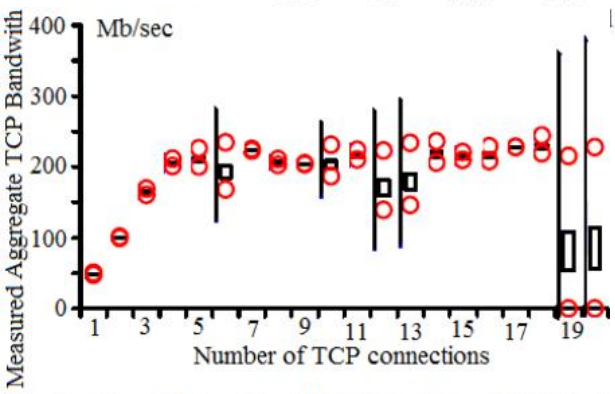


Fig. 2 – Throughput of Parallel TCP Sockets with MSS of 2948 Bytes

© Мельник В.М.

the slow-start algorithm will decrease the congestion window by half. This leads to the «saw tooth» pattern shown in fig. 4.

If the assumption that p is a constant probability is modified by the assumption that, for an individual TCP stream, it is independent of the loss rate of other TCP streams from the same sender on an uncongested network, and that for each stream *i*, *pi* is from a distribution identical to the other distributions for loss rate, the situation described in fig. 4 can be used to describe the effects of parallel TCP connections as shown in fig. 5.

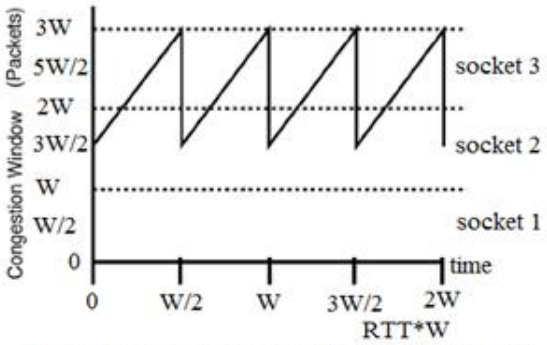


Fig. 6 – Geometric Construction of the Aggrigate Effects for Multiple TCP Connections

Given that the packet loss rates of parallel TCP connections are not all sensitive to traffic, and that packet losses occur in each channel at the same rate (as long as packet losses are not due to network congestion), an interesting effect occurs. If the three streams in Figure 5 are combined into the aggregate representation shown in Figure 6, it is clear that using multiple network sockets is in essence equivalent to increasing the recovery rate from a loss event from one MSS per successful transmission to three times MSS. Note that this increased recovery rate is theoretical and functionally equivalent to using a larger MSS on a single channel with the same packet loss rate.

As the number of simultaneous TCP connections increases, the overall rate of recovery increases until the network begins to congest. At this point, the loss rate of the packet becomes dependent on the number of sockets and the amount of congestion in the network. The packet loss rate change indicates that the network is congested, and that the TCP sender should reduce its congestion window. As the number of parallel TCP connections increases, and the higher packet loss rates decrease the impact of multiple sockets, the aggregate TCP bandwidth will stop increasing, or begin to decrease.

Given that the aggregate rate of congestion recovery across the parallel TCP streams is functionally equivalent to an increased recovery rate, there is an interesting observation that can be made. TCP connections over wide area networks suffer from the disadvantage of long round trip times relative to other TCP connections. This disadvantage allows TCP senders with small RTTs to recover faster from congestion and packet loss events than TCP sessions with longer RTTs. Since the use of parallel TCP sockets provides a higher recovery rate, hosts with longer RTTs are able to compete on a fairer basis with small RTT TCP connections for bandwidth in a bottleneck.

Selecting the Number of Sockets

When the packet loss rate *p* transitions from the random loss to the congestion loss regime, the benefits from using additional sockets is offset by the additional aggregate packet loss rate. From the previous section, it is apparent that the knee that is present in the TCP bandwidth curve directly corresponds to the knee in the packet loss curve. The challenge in selecting an appropriate number of sockets to maximize throughput is thus the problem of moving up to, but not beyond, the knee in the packet loss curve.

Any application using parallel TCP connections must select the appropriate number of sockets that will maximize throughput while avoiding the creation of congestion. The applications avoid congesting a network to prevent congestion collapse of the bottleneck link. In aggreement with the data, adding additional TCP connections beyond the knee in the packet loss curve has no additional benefit, and may some decrease aggregate performance.

Determining the point of congestion in the end-to-end network a priority is difficult, if not impossible, given the inherent dynamic nature of a network. However, it may be possible to gather relevant parameters using Web100 from actual data transfers, which then can be used in combination with prediction methods of statistical time-series to attempt the predict of the end-to-end packet loss rate *p*, RTT and MSS, and thus the limit on TCP bandwidth. In addition to use statistical predictions to forecast the value of *p*, it may also be possible to use the same techniques to collect and store information on the number of parallel TCP connections necessary to maximize aggregate performance and avoid congestion. To maximize

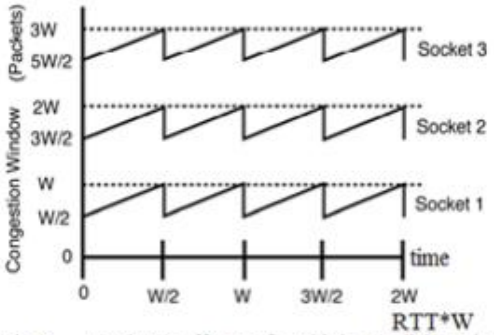


Fig. 5 – Aggrigate Effects of Multiple Parallel Sockets

© Мельник В.М.

throughput, the predicted values of *p* and the effective number of parallel TCP connections can then be used as a starting point for a simple greedy search algorithm that adjusts the number of parallel TCP connections.

Conclusion

This work puts the question of how parallel TCP connections can improve aggregate TCP bandwidth. It also addresses the question of how to choose the maximum number of sockets necessary to maximize TCP throughput while imultaneously and avoid congestion. A theoretical model was developed to analyze the questions. The findings indicate that in the absence of congestion, the use of parallel TCP connections is equivalent to using maybe a large MSS on a single connection, with the added benefit of reducing the negative effects of random packet loss. It is imperative that application developers do not arbitrarily select a value for the parallel TCP connection number. If the selected value is too large, the aggregate flow may cause network congestion as well and throughput will not be maximized.

References

1. Lee, J., Gunter, D., Tierney, B., Allock, W., Bester, J., Bresnahan, J. and Tecke, S. Applied Techniques for High Bandwidth Data Transfers across Wide Area Networks Dec 2000, LBNL-46269.
2. Sivakumar, H., Bailey, S., Grossman, R. L., PSockets: The Case for Application level Network Striping for Data Intensive Applications using High Speed Wide Area Networks, SC2000: High-Performance Network and Computing Conference, Dallas, TX, 11/00.
3. Bolot. J-C., «Characterizing End-to-End packet delay and loss in the Internet», Journal of High Speed Networks, 2(3), 1993, – pp 305 – 323.
4. Pittsburgh Supercomputer Center Networking Group. «Enabling High Performance Data Transfers on Hosts», http://www.psc.edu/networking/perf_tune.html.
5. Floyd, S., and Fall, K., Promoting the Use of End-to-End Congestion Control in the Internet, IEEE/ACM Transactions on Networking, August 1999.
6. Lee, J. Gunter, D., Tierney, B., Allock, W., Bester, J. Bresnahan, J., and Tecke, S., Applied Techniques for High Bandwidth Data Transfers across Wide Area Networks. Sept. 2001, LBNL-46269, CHEP 01 Beijing China.
7. Wolski, R., «Dynamically Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service.» In 6th High-Performance Distributed Computing, Aug. 1997.
8. Sander, V. Adamson, W., Foster, I., Alain, R. End-to-End Provision of Policy Information for Network QoS. In 10th High-Performance Distributed Computing, August 2001.
9. Georgiadis L., Guerin R., Peris, V., Sivarajan K. Efficient network QoS provisioning based on per node traffic shaping. IEEE ACM Trans. on Networking, Aug., 1996.
10. Long, R., L. E. Berman, L. Neve, G. Roy, and G. R. Thoma, "An application-level technique for faster transmission of large images on the Internet", Proceedings of the SPIE: Multimedia Computing and Networking 1995 Vol. 2417 February 6-8, 1995, San Jose, CA.
11. Long L. R., Berman L E., Thoma GR. "Client/Server Design for Fast Retrieval of Large Images on the Internet. Proceedings of the 8-th IEEE Symposium of Computer-Based Medical Systems (CBMS '95), Lubbock TX, June 9 – 10, 1995, pp. 284 – 291.
12. Allman, M. Ostermann, S. and Kruse, H. Data Transfer Efficiency Over Satellite Circuits Using a Multi-Socket Extension to the File Transfer Protocol (FTP). In Proceedings of the Acts Results Conference. NASA Lewis Research Center, September, 1995.
13. Eggert, L., Heidemann, J. and Touch, J. Effects of Ensemble-TCP. ACM Computer Communication Review, 30 (1), January, 2000, – pp. 15 – 29.
14. Balakrishnan, H., Rahul, H. and Seshan, S., "An Integrated Congestion Management Architecture for Internet Hosts", Proc. ACM SIGCOMM, September 1999.
15. Internet End-to-End Performance Monitoring. http://www-iepm.slac.stanford.edu/.
16. Cohen, E., Kaplan, H. and Oldham, J., "Managing TCP Connections under Persistent HTTP", Proceedings of the Eighth International World Wide Web Conference, Toronto, Canada, May 1999.
17. Grid Forum GridFTP Introduction: http://www.sdsc.edu/GridForum/RemoteData/Papers/gridftp_intro _gf5.pdf.
18. Baru, C., Moore, R., Rajasekar, A. and Wan, M., "The SDSC Storage Resource Broker." In Procs. of CASCON'98, Toronto, Canada, 1998
19. Floyd, S., "Congestion Control Principles", RFC 2914.
20. Semeria, C. «Internet Processor II ASIC: Rate-limiting and Traffic-policing Features.» Juniper Networks White Paper. http://www.juniper.net/techcenter/techpapers/200005.html.
21. Floyd, S. and Jacobson, V. Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking, 1(4), August 1993, – 397 – 413.
22. http://citeseer.nj.nec.com/floyd93random.html.
23. Bolliger, J., Gross, T. and Hengartner, U., Bandwidth modeling for network-aware applications. In INFOCOM'99, March 1999.
24. Padhye, J., Firoiu, V., Towsley, D. and Kurose, J., Modeling TCP throughput: a simple model and its empirical validation. ACMSIGCOMM, September 1998.
25. Mathis, M., Semke, J., Mahdavi, J. and Ott, T., «The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm». Computer Communication Review, July 1997, vol. 27, No 3.
26. Paxson, V., «End-to-end Internet packet dynamics.» in Proc. ACM SIGCOMM, September 1997. – pp. 139 – 152.

© Мельник В.М.

27. Mogul, J. and Deering, S., "Path MTU Discovery," Network Information Center RFC 1191. – Apr. 1990, pp. 1 – 19.
28. Jacobson, V., Congestion Avoidance and Control. In Proceedings of the ACM SIGCOMM'88 Conference. – 1988. – pp. 314 – 329.
29. Chiu, D-M., Jain, R., "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," Computer Networks and ISDN Systems, – 1989, vol. 17. – pp. 1 – 14.
30. Borella, M. S., Swider, D., Uludag, S. and Brewster, G., "Internet Packet Loss: Measurement and Implications for End-to-End QoS Proceedings. International Conference on Parallel Processing, Aug. 1998.
31. Floyd, S. and V. Jacobson. 1992. On traffic phase effects in packet-switched gateways. Internetworking: Research and Experience 3. – pp. 115 – 156.
32. Feng, W. and Tinnakornsrisuphap, P., "The Failure of TCP in High-Performance Computational Grids", SC2000: High-Performance Network and Computing Conference, Dallas, TX, 11/00.
33. Guo, L., Crovella, M. and Matta, I., "TCP congestion control and heavy tails," Tech. Rep. BUCSTR – 2000-017, Computer Science Dept – Boston University, 2000.

© Мельник В.М.