Single / multi TCP and a fairness issue

Dunia Meftah Shwehdy Fatma Ali Badish Nadia Ali Badish

Abstract:

In distributed systems domain a good connection is a structural quest for messages transfer between clusters in today's high performance applications. Initiating a single TCP used in networks, than it drops as a bundle of TCP's called multiple or parallel TCP. This drop wasn't quite successful it also created some ugly fouling , for this reason some studies concluded that in general resultant a multi TCP is not much differ of a single TCP ,but others refined that the multi socket TCP achieves better throughput and performance than the single socket approach [1]. The applications that require good network performance often use multi socket TCP streams and TCP modifications to improve the effectiveness of TCP .but if the network bottleneck is fully utilized, this approach boosts throughput by unfairly stealing bandwidth from competing TCP stream. This dilemma driven to some opinions see that improve the effectiveness of TCP is much easier than to while maintaining fairness [2]. in this paper we submit our own destination to compare between single-socket TCP and multi-socket TCP to conclusiveness the argumentativeness according about any one is the best, in performance and throughput ,we also discussed a fairness issue in both ,and finally we defined an algorithm for enhance multi TCP's performance.

Key words

(TCP)Transmission Control Protocol, stream, congestion avoidance algorithm (CAA), socket, parallel TCP, single TCP, Link sharing, bottleneck, bandwidth, fairness, (AIMD) Additive-Increase Multiplicative-Decrease process.

ملخص البحث:

فيما يتعلق بمنظومة توزيع البيانات؛ يعد تأسيس اتصال جيد مَطْلَب تكويني لنقل الرسائل بين المجموعات في عصر البرمجيات "التطبيقات" عالية الأداء، فإنشاء بروتوكول أحادي للتحكم بنقل البيانات ليستخدم في برمجة الشبكات يترتب عليه حزمة البروتوكول المعروفة بالبروتوكول الموازي أو المتعدد. هذا التفرّع أو الانسدال لم يحقق أيّة فعالية، كما نجم عنه قصور وعيوب أخرى، ولهذا خلُصَت بعض الدراسات إلى أن البروتوكول المتعدد لا يختلف كثيراً عن البروتوكول الأحادي، بينما أظهرت أبحاث أخرى أن البروتوكول متعدد المصدر قادر على تحقيق أداء أفضل وفعالية أكبر من البروتوكول أحادي المصدر. وغالبًا ما تستخدم البرمجيات " التطبيقات" التي تتطلب شبكة اتصال ذات أداء عالي تدفق بروتوكولات متعددة المصدر وبروتوكولات تصحيح لتحسين معدل النقل للبيانات، ولكن إذا ما حدث اختناق داخل الشبكة؛ فهذا النهج يقوم بتسريع الأداء من خلال تضييق عرض النطاق على تدفق البيانات للبروتوكول "بروتوكول التحكم بالنقل". وقد دفعت هذه المشكلة بالبعض أن تحسين فعالية البرتوكول أيسر من الحفاظ على ثبات معدل نقل البيانات "الإنصاف". سنقدم في هذه الورقة دراسة مقارنة بين البروتوكول أحادي المصدر والبروتوكول متعدد المصدر والخروج بنتائج حول الجدل القائم عن الأنجع من ناحية الأداء والفعالية لكلا نوعي البروتوكول، حيث تناولت هذه الدراسة أيضًا معيار "الإنصاف" لكليهما، وانتهت بوضع خوارزمية لتحسين أداء بروتوكول التحكم لنقل البيانات.

الكلمات الأساسية: بروتوكول التحكم بالنقل، التدفق، خوارزمية منع الازدحام، مصدر، البروتوكول الموازي، البروتوكول الأحادي، مشاركة الاتصال، مختنق، عرض النطاق، الإنصاف، عملية الزيادة المضاعفة والانخفاض المضاعف

I. Introduction

Now day, moving a big size of data quickly over a shared wide area network is a necessity for many applications. As example The Atlas project that aim to move many petabytes of data per year between Europe and the United States, Multiuser collaborative environments combine visualization, video conferencing, and remote application steering into a distributed application with low latency and high bandwidth demands. And the Optiputer project target building a distributed computer using a wide area network path for a backplane. Many tools are used by applications that need to move large amounts of data over wide area networks. These huge projects need to tool to be able for meeting there needing, Many of these applications were used the Transmission Control Protocol (TCP) for accurate and reliable in-order transmission of data [5]. TCP relies on the congestion avoidance algorithm (CAA) to measure the capacity of a network path, fairly share bandwidth between competing TCP streams, and to increasing exploit the effective use of the network. But it was a relative slow, especially with starving to maximize the transmission speed during networks, so a new trend was taken, a session-layer solution designed for parallelizing stream data transfers. Parallelization is achieved by striping the data flow among multiple TCP channels. This solution did not require invasive changes to the networking stack and it be implemented entirely in user space. Moreover, it is flexible enough to suit several scenarios – e.g. it used to split a data transfer among multiple relays within a peer-to-peer overlay network [4].In this paper, we discussed many important and nail-biting issues and also we conclusive some issues and proposed a solution.

The paper is organized as follow:

In section II we will present a brief discuss why networks trend to using a multi TCP.
In Section III we present our results that demonstrate the different between pair forms of TCP connections in a performance and throughput.
In Section IV presents proposals for improving some of faults of multi TCP that limit its performance.

II. Why multi TCP?

Multi TCP streams are used for data transfer between clusters in high performance applications today, for example, in Data Reservoir system, parallel TCP streams are used for disk-to-disk data transfer between clusters. Data Reservoir clusters at distant locations are connected with high-bandwidth, long distance network, which is called Long Fat pipe Network (LFN) [3], and almost important applications through networks, multi TCP mean that one congestion window for all TCPs to the same destination in application layer, these TCPs are served in a round-robin fashion. It is extremely effective because the protocol became easily leveraged by all the applications that were use TCP with some changes in them. The Parallel TCP Transfers is a layer-V architecture designed to streak a TCP data flow over multiple (physical or logical) channels. This technique honored with being relatively simple to implement because it relies on TCP for the physical data transfer. But what is a TCP? TCP is a standard protocol for reliable data transfer. It uses acknowledgment (ACK) packet and retransmission mechanism to achieve reliable data transfer over unreliable network. The receiver sends ACK back to the sender to notify a successful reception of data. Data that has been sent but not acknowledged is called in-flight data. this pretty feature is one of others , where all independent on a segment with strong structure, this structure available Connection oriented where explicit set-up and tear-down of TCP session ,also Stream-of-bytes service where Sends and receives a stream of bytes, not messages ,and Reliable, in-order delivery wherever Checksums to detect corrupted data ,Acknowledgments & retransmissions for reliable delivery ,And Sequence numbers to detect losses and reorder data ,additionally to Flow control that prevent overflow of the receiver's buffer space ,furthermore Congestion control that mean adapt to network congestion for the greater good . Although all features above, TCP has a limitation namely the capability to open only a single logical channel between two communicating host, this characters is not suitable in the present-day Internet.

The maximum theoretical speed of a single TCP connection is bounded by the window size and the Round Trip Time, which often prevent the full utilization of high-speed links (unless in presence of specific settings)[6][8].

Additional to that TCP relies on the congestion avoidance algorithm (CAA) to measure the capacity of a network path, fairly share bandwidth between competing TCP streams, and to maximize the effective use of the network.

Unfortunately, the CAA prevents the effective and efficient use of wide area networks for these applications. The CAA relies on packet loss to indicate that the network is overloaded, and responds to packet loss by decreasing the transmission rate of the TCP stream by half. Van Jacobson assumed three things about packet loss when he designed the CAA – the fraction of non-congestion packet loss (”damaged packets”) is << 1%; network speeds (circa 1988) are low enough to prevent measurable sensitivity to low (< 5%) loss rates; and an implicit assumption that packet reordering would not affect the congestion avoidance algorithm. Jacobson found that the CAA is sensitive to packet loss when the packet loss rate is the same order of magnitude as the square of the congestion window in packets. This has become a problem for high-speed networks, since the maximum frame size is usually fixed at1500 bytes [5].

There is a substantial body of evidence that effects such as packet reordering and non-congestion packet loss can degrade TCP performance over wide area networks. The no congestion losses perceived by TCP are due to many factors unrelated to simple network congestion that include end-host and network infrastructure effects. These effects on wide-area high-speed networks can lead to situations in which it can require over 4 hours for a TCP stream to completely recover from one loss event. Non-congestion loss also has significant effects on satellite links. There are efforts within the networking community to design modifications to TCP congestion avoidance to overcome these effects.

Most of these efforts rely on mechanisms that aggressively compete for network bandwidth, which can lead to unfair behaviors at best, and congestion collapse at worst [5].

So we can see that the TCP in single form is not suitable for high speed networks, but also using a multi TCP generate many problem overmatch its penefitage, these faults crop up as fallowing : may perform worse if loss is due to congestion or may add to congestion also selecting the buffer size and number of streams is problematic additional to important point where exploits

TCP's fairness, so it be unfair to other flows, so we will clench this is affair further to performance and throughput .

III. The comparison

Assuming we have a network with topology and typical single bottleneck as following in figure (1)

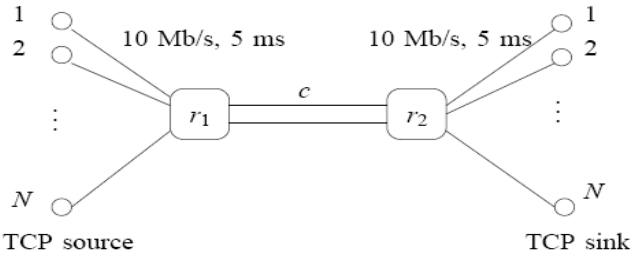

Figure1. The Network Topology.

Our performance metrics has been throughput and fairness between single-socket TCP and multi-socket TCP. So we need to remember two concepts, first is bandwidth utilization (throughput) is the amount of data moved successfully from one place to another in a given time period over a physical or logical link. The other is the fairness where the fairness metric used to determine whether applications which use the same link or connection are receiving a fair share of bandwidth. The results of this work show that the single socket TCP cannot emulate the parallel TCP in terms of performance throughput and throughput ratio , also since the multi socket occurs in the application layer that divides the data into small chunks and provide them to the lower layers in order to construct independent connections. Hence these flows will behave independently in the lower layers of the network and it will result a positive impact on fairness. In single socket TCP AIMD algorithm (Additive-Increase Multiplicative-Decrease process where the rate at which packets are sent increases linearly in time, but suffers a multiplicative decrease as soon as a loss is detected. Then the rate increases linearly again until the next loss, and so on.[7]) reduce its congestion window to the half which leads to decrease the total throughput. On the other hand, it leads to insufficient bandwidth utilization. Whilst, in multi socket TCP, the timeout which occurs in one of parallelized flows will not affect the congestion window of the other flows in the same set. For instance, suppose that

we have parallel of five flows and timeout takes places on two flows, this will result the reduction of 1/5 from the total throughput of this multi connection. From this comparing we demonstrated that multi TCP performs more efficient than single socket TCP in the high speed networks and too have produced better throughput than the single socket TCP while it maintains the fairness.

IV. Partial solution

We know that the sockets are present the interface to TCP connections, we can defined the socket as A way to write a computer program that can send data to the network. From a programmer's point of view, works very much like reading / writing a file. Within the socket we can tells the OS/Protocol Stack the behavior we want , Many socket options are Boolean flags indicating whether some feature is enabled (1) or disabled (0) , they are Available on many high level programming languages (C, C++, Java, Python, VB, etc).

After this brief introduction about the sockets , we will return to some bad characters to multi TCP streams , we knew that multi TCP independent on multiplexing tcp streams within connection channels through the socket that contain the algorithms that lead TCP's streams and control in them , we knew also that from multi TCP main disadvantages that selecting the buffer size and number of streams is problematic and rising in congestion rate because might a number of streams are not suitable , and many times a number of streams take space without any payload of data , in consequence at will increasing in congestion and reduce in fairness level too much. So we can define an algorithm capable to low this bad effect and enhancing in multi TCP's global performance. This algorithm working as checker to multi TCP streams during its crossing from the socket will allows to only streams that has payloads to pass and freezing empty streams. This technique grants better performance for multi TCP.

The algorithm:

For i:=1 to n{If stream[i] =0 than pass this stream ;
Else Freeze this stream;}End

References :

[1] http://research.microsoft.com/˜milanv/socks.htm.
[2] http://www.planet-lab.org.

[3] Yutaka Sugawara ,Mary Inaba ,and Kei Hiraki . Flow Balancing Hardware for Parallel TCP Streams on Long Fat pipe Network . International Journal of Software Engineering and Its Applications ,Vol. 2, No. 2, April, 2008 .
[4] Thomas J. Hacker Brian D. Athey . The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network.
[5] Thomas J. Hacker , Brian D. Noble and Brian D. Athey . Improving Throughput and Maintaining Fairness using Parallel TCP .
[6] George Xylomenos, George C. Polyzos, Athens University of Economics and Business, Greece ,Petri Mähönen, University of Oulu, Finland, and VTT, Technical
Research Center of Finland and Mika Saaranen, University of Oulu, Finland . TCP Performance Issues over Wireless Links
[7] Bruno Tuffin and Patrick Maill´e IRISA-INRIA, Campus universitaire de Beaulieu
35042 Rennes Cedex, France . How Many Parallel TCP Sessions to Open: a Pricing Perspective .
[8] Andrea Baldini, Lorenzo De Carli, and Fulvio Risso . Increasing Performance of TCP Data Transfers Through Multiple Parallel Connections .
[9] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of tcp/ip with stationary random losses. In Proc. of the Sigcomm'00, pages 231–242, 2000.
[10] Thomas J. Hacker , Brian D. Noble and Brian D. Athey . Improving Throughput and Maintaining Fairness using Parallel TCP .
[11] E. Altman, D. Barman, B. Tuffin, and M. Vojnovi´c. Parallel tcp sockets: Simple model, throughput and validation. Microsoft Research Technical Report,http://research.microsoft.com/˜milanv/TR-2005-130.pdf, 2005.
[12] E. Altman, F. Boccara, J. Bolot, P. Nain, P. Brown, D. Collange, and C. Fenzy. Analysis of the tcp/ip flow control mechanism in high-speed wide-area networks. In 34th IEEE Conference on Decision and Control, pages 368–373, New Orleans, Louisiana, Dec 1995.
[13] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing Router Buffers. In Proc. of ACM Sigcomm 2004, Portland, Oregon, USA, Aug/Sept 2004.
[14] F. Baccelli and D. Hong. AIMD, Fairness and Fractal Scaling of TCP Traffic. In Proc. of IEEE Infocom 2002, New York, NY, http://www.di.ens.fr/_trec/aimd, 2002.
[15] J. Crowcroft and P. Oechslin. Differentiated End-to-End Internet Services using a Weighted Proportional Fair Sharing TCP. ACM Computer Communications Review, 47(4):275–303, 2004.

[16] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Trans. on Networking, 7(4):458–472, August 1999.
[17] S. Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. IEEE/ACM Trans. on Networking, 1(4):397–413, August 1993.
[18] T. Hacker, B. Athey, and B. Noble. The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network. In 16th IEEECSACM International Parallel and Distributed Processing Symposium, Ft. Lauderdale, FL, April 2002.
[14] Van Jacobson and M.J. Karels. Congestion avoidance and control. In Proc. of the ACM SIGCOMM'88, pages 314–329, Stanford, August 1988.
[19] Thomas J. Hacker Brian D. Athey . The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network.
[20] R. El Khoury and E. Altman. Analysis of Scalable TCP. In In Proc. Of Het-Nets'04, West Yorkshire, United Kingdom, July 2004.
[21] Dong Lu, Yi Qiao, Peter Dinda, and Fabian Bustamante. Modeling and Taming Parallel TCP on the Wide Area Network. In In Proceedings of the 19th IEEE IPDPS'05, Denver, Colorado, April 2005.
[22] J. Mahdavi and S. Floyd. TCP-Friendly Unicast Rate-Based Flow Control. Technical note sent to end-2-end interest mailing list, http://www.psc.edu/networking/papers/tcp friendly.html, January 1997.
[23] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno Performance: A Simple Model and its Empirical Validation. IEEE/ACM Trans. on Networking, 8(2):133–145, 2000.
[24] Thomas J. Hacker Brian D. Athey . The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network.
[25] T. Kelly. Scalable TCP Improving Performance in HighSpeed Wide Area Networks. In PFLDNet'03, Geneve, Switzerland, February 2003.
[26] E. Altman, R. El Azouzi, D. Ros, and B. Tuffin. Loss Strategies for Competing TCP/IP Connections. In Proc. of the Networking 2004, Athens, Greece, May 2004.