International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Modified Transfer Mechanism over Multipath TCP

Sreejith N[1], Maya Mohan[2]

[1]S_4 M. Tech, Department of Computer Science and Engineering, NSS College of Engineering, Palakkad, Kerala, India

[2]Assistant Professor, Department of Computer Science and Engineering, NSS College of Engineering, Palakkad, Kerala, India

Abstract: ***With the presence multipath transport protocols (mptcp), a host can shift some of its traffic from more congested paths to less congested ones. Doing so it can compensate for lost bandwidth on some paths by moderately increasing transmission rates on other routes. But existing multipath proposals concentrate on a rough estimate of network congestion using packet losses, which is not so efficient. This project mainly focuses on multipath congestion control and proposes a modified mechanism for packet data transfer which is based on the packet queuing delay as congestion signals and thus achieves a good performance.***

Keywords: mptcp, round trip time, congestion

1. Introduction

Multipath TCP, as proposed by the IETF working group mptcp, allows a single data stream to be split across multiple paths. This has obvious benefits for reliability— the connection can persist when a path fails. A host is multihomed if it can be addressed by multiple IP addresses, as is the case when the host has multiple network interfaces. Though feasibility alone does not determine adoption of an idea, multihoming is increasingly economically feasible. Multihomed nodes may be simultaneously connected through multiple access technologies. For instance, a mobile user could have simultaneous Internet connectivity via a wireless local area network using 802.11b and a wireless wide area network using GPRS. [1]

2. Literature Survey

2.1 MPTCP

Hosts are often connected by multiple paths, but TCP restricts communications to a single path per transport connection. Resource usage within the network would be more efficient were these multiple paths able to be used concurrently. This should enhance user experience through improved resilience to network failure and higher throughput.

The two key benefits of multipath transport are the following:

- To increase the resilience of the connectivity by providing multiple paths, protecting end hosts from the failure of one.
- To increase the efficiency of the resource usage, and thus increase the network capacity available to end hosts.

Multipath TCP is a modified version of TCP that implements a multipath transport and achieves these goals by pooling multiple paths within a transport connection, transparently to the application. Multipath TCP is primarily concerned with utilizing multiple paths end-to-end, where one or both of the end hosts are multihomed. [2]

2.1.1 MPTCP Working Group

The Multipath TCP working group develops mechanisms that add the capability of simultaneously using multiple paths to a regular TCP session. The Key goals for MPTCP are: to be deployable and usable without significant changes to existing Internet infrastructure; to be usable by unmodified applications and to be stable and congestion-safe over the wide range of existing Internet paths, including NAT interactions. MPTCP assumes that both peers are modified and that one or both peers have multiple addresses, which often results in different network paths that are at least partially divergent (however, note there is no guarantee that the paths are divergent at all). [3]

2.2 CMT-SCTP

Concurrent Multipath Transfer Stream Control Transmission Protocol also has the properties of MPTCP. CMT is the concurrent transfer of new data from a source to a destination host via two or more end-to-end paths. The Stream Control Transmission Protocol (SCTP) is an IETF standards track protocol that natively supports multihoming at the transport layer. It is a general-purpose, connection-oriented, unicast transport protocol which provides the reliable transport of user messages and a multi-homing concept out of the box. SCTP multihoming allows binding of one transport layer's association (SCTP's term for a connection) to multiple IP addresses at each end of the association. This binding allows a sender to transmit data to a multihomed receiver through different destination addresses. Simultaneous transfer of new data to multiple destination addresses is currently not allowed due primarily to insufficient research. [4]

CMT also inherently adds to SCTP's fault tolerance, which is a major motivation for and benefit of multihoming. An SCTP sender gathers information about paths to alternate destination addresses through explicit probes. Since explicit probes are infrequent, a sender has inadequate information and consequently, is unable to make an informed decision about which destination to use when the primary destination becomes unreachable. A CMT sender avoids this problem because data sent concurrently on all paths act as frequent implicit probes, reflecting current conditions of paths to all destinations. This information will better assist a CMT sender in detecting and responding to network failures.

Licensed Under Creative Commons Attribution CC BY

Many protocols were proposed in an attempt to transfer data through multiple paths in parallel.

pTCP, allows a connection to utilize the aggregate bandwidth offered by multiple paths, and it assumes the wireless link is the bottleneck to ensure fairness. The work Improving Throughput and Maintaining Fairness using Parallel TCP improves the fairness of parallel TCP in under-utilized networks by using a long virtual round trip time. [5]

mTCP focuses on detecting shared congestion at bottleneck links by computing the correlation between fast retransmit intervals on different paths. which can aggregate the available bandwidth of those redundant paths in parallel. By striping one flow's packets across multiple paths, mTCP can not only obtain higher end-to-end throughput but also become more robust under path failures. When some paths fail, mTCP can continue sending packets on other living paths and the recovery process normally takes only a few seconds. Because mTCP could obtain an unfair share of bandwidth under shared congestion, the paper integrates a shared congestion detection mechanism into our system. It allows us to dynamically detect and suppress paths with shared congestion so as to alleviate the aggressiveness problem. mTCP can also passively monitor the performance of several paths in parallel and discover better paths than the path provided by the underlying routing infrastructure. [6]

cTCP provides a single congestion window for all the paths and maintains a database at senders to record the relationship between packet sequences and the paths for the purpose of detecting losses. cTCP uses loss probability to estimate path capacity so as to put more packets on high bandwidth paths. CMT-SCTP improves SCTP for the purpose of multipath transfer in parallel. [7]

3. Background and Problem Statement

The current transport protocol workhorses, TCP and UDP, do not support multihoming; TCP allows binding to only one network address at each end of a connection. At the time TCP was designed, network interfaces were expensive components, and hence multihoming was beyond the ken of research. Changing Economics and an increased emphasis on end-to-end fault tolerance have brought multihoming within the purview of the transport layer. While concurrency can be arranged at other layers, the transport layer has the best knowledge to estimate end-to-end paths characteristics. [2]

A major objective of multipath congestion control is to couple all the subflows belonging to a flow together so as to achieve both fairness and efficiency. By this kind of coupling method, most of existing multipath proposals provide the ability of load balancing that can shift some traffic from more congested paths to less congested ones, thus compensating for lost bandwidth on some paths by moderately increasing transmission rates on other ones. However, these proposals achieve only coarse-grained load balancing, because they estimate network congestion and then trigger traffic shifting using packet losses that lack of the finer-grained information related to the extent of congestion. [8]

One example is about efficiency. In Figure 1 which was firstly presented in mTCP, every flow has two paths available for data transfer. If each flow transmits data at rate $0 \le x \le 9$ on its one hop path and at rate $(9-x)$ =2 on its two-hop path, then bandwidth sharing is always fair.

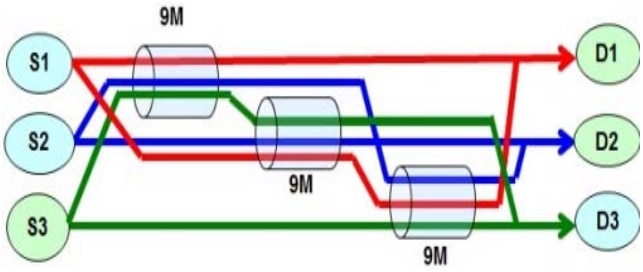


Figure 1: Efficiency Example

Among these fair outcomes, the most efficient one is $x = 9$, because every flow can obtain the maximum transmission rate [9]

4. The network utility Maximization Model

The network contain the set L of links with the finite capacities c = ($cl;\ l \in L$), which are shared by the set S of flows. A path $r \in R$ is defined as the subset $Lr \subseteq L$. The relationship between L and R is given by the routing matrix A, where $al;r = 1$ if $l \in Lr$, and $al;r = 0$ otherwise. Each flow $s \in S$ is associated with a subset $Rs \subseteq R$. This relationship is given by the matrix B, where $bs;r = 1$ if $r \in Rs$, and $bs;r = 0$ otherwise. Let $xs;r$ be the rate of flow s on path r, and $ys = \Sigma\ r2Rs\ xs;r$ be the total rate of flow s. Denote the vector ($xs;r;\ s \in S;\ r \in Rs$) by x, and the vector ($ys;\ s \in S$) by y. When flow s transmits data at rate ys, it obtains an utility $Us(ys)$. Suppose $Us(\cdot)$ is increasing, strictly concave and twice continuously differentiable in the nonnegative domain. Define $Us(0) = -\infty$. The objective of congestion control is to determine appropriate rates for the flows so as to maximize the total utility subject to link capacity constraints [11], [12].

Thus, we have

$$\max_{\mathbf{x} \ge 0} \sum_{s \in S} U_s(y_s)$$

$$s.t. \quad \mathbf{y} = \mathbf{B}\mathbf{x}$$

$$\mathbf{A}\mathbf{x} \le \mathbf{c}$$

The problem can be solved using Lagrangian functions and we can derive it as

$$\min_{\lambda \ge 0} \sum_{s \in S} L_s(\lambda) + \lambda \mathbf{c}^T$$

$$L_s(\lambda) := \max_{\{x_{s,r} \ge 0, \atop r \in R_s\}} U_s\left(\sum_{r \in R_s} x_{s,r} \right) - \sum_{r \in R_s} q_r x_{s,r}$$

5. Methodology

For a flow, the total sum of the parameters α of subflows is fixed, regardless of the number of subflows. This property contributes to the intra-protocol fairness of wVegas. Most significantly, wVegas adaptively adjusts the parameter α thereby influencing the transmission rate of the

Licensed Under Creative Commons Attribution CC BY

corresponding sub flow for the purpose of equalizing the extent of congestion on the path. We define the normalized α as the weight of subflows. Thus, in this sense, the weight quantifies the aggressiveness of competition for bandwidth. Incidentally, increasing the weight of a sub flow may not always push up the transmission rate, albeit making that sub flow more aggressive to compete for bandwidth, because other flows might also increase the weight of their own subflows.

The implementation phase gives a variable array equilibrium_rates [r] to store the transmission rate of path r, a queue_delay[r] to store the minimum queue delay of path r and alpha[r] to store the expected backlogged packets. The pseudo code and the analysis of the specific algorithm are shown as below [15]. In the Initialization phase:

- total_alpha <- fixed number---- The network would assign a fixed number of backlogged packages to the coming flow and save it in the total_alpha.
- Equilibrium_rates[r] <- 0, queue_delay[r] <- 0, alpha[r] <- random number
- Base_RTT[r]----Measure the minimum RTT of path with no queuing delay

During the congestion avoidance phase after each round of path r:

- RTT ← Sampled_rtts[r]/Sampled_num[r]. Calculate the average RTT of the previous round
- Diff ← cwnd[r] (RTT-Base_RTT[r]) / RTT----Calculate the actual number of backlogged packages of path r. It is the queuing delay indicates extension of congestion
- If Diff ≥ alpha[r] then
- Equilibrium_rates[r] ← cwnd[r]/RTT
- Adjust weight[r] ← Equilibrium_rates[r]/total_rate
- alpha[r] ← weight[r] multiply total_alpha
- alpha[r] ← max (alpha[r], 2) end if

This if block is used for updating the transmission rate, weight and expected backlogged packages of path r. The transmission rate and the weight is only updated when the number of actual backlogged packages is larger than the number of expected backlogged packages. The last instruction ensures that even if the path occupies no weight, it is still not abandoned and assigned a reasonable load.

- If Diff < alpha[r] then cwnd[r] ← cwnd[r] + 1
- If Diff > alpha[r] then cwnd[r] ← cwnd[r] - 1

Update the sending window size according to the relationship between expected backlogged packages and actual backlogged packages.

- q ← RTT - Base_RTT
- if queue_delay[r] = 0 or queue_delay[r] > q then queue_delay[r] ← q
- if q > 2 multiply queue_delay[r]
- backoff_factor ← 0.5 multiply Base_RTT[r] / RTT
- queue_delay[r] ← 0
- cwnd[r] ← max (cwnd[r], 2). Ensure that no path is abandoned totally

6. Evaluation

The proposed Algorithm is implemented using NS2.34 and evaluated the performance. wVegas can achieve the most efficient bandwidth sharing. wVegas can quickly complete traffic shifting, since it is more sensitive to changes of network congestion than other congestion control algorithms. wVegas attempts to backlog fewer packets in link queues, thus stabilizing links into a fully-utilized state with fewer losses. This property facilitates wVegas to cope with the variation of RTTs. [13] [16]

One of the key ideas of multipath congestion control is to couple the rate adjustment process on each sub flow together by means of a specially designed algorithm so as to achieve traffic shifting. Thus, the lost bandwidth on a path due to congestion events can be compensated by increasing rates of other subflows. As a result, a congestion event occurring in one place might cause flows in other places to change rates. This phenomenon is somewhat similar to the domino effect.

The computational overhead of wVegas is inexpensive, though it involves floating-point division. This is because the frequency of weight adjustment is one time per RTT for each subflow and the number of paths used by a flow is also small in most cases. Additionally, there exist many approximation methods that can convert floating-point calculation to integer operations. [14]

7. Conclusion and Future Scope

Based upon the network utility maximization model, we can prove the Congestion Equality Principle, and propose an approximate iterative algorithm for solving the problem of multipath congestion control. These two components together establish a general framework for designing an algorithm of multipath congestion control. Using this framework, we can develop wVegas and evaluate its performance in terms of fairness and efficiency.

A possible modification could be made on how to change the congestion control window according to the queuing delay. Actually the queuing delay could do more than just judging whether the path is congested or not, it could also tell how far it is from the situation of congestion. Besides, whether or not to shut down seriously congested paths is also an open issue. Hope this project can resolve the issues stated above.

References

[1] Yu Cao, Mingwei Xu, Xiaoming Fu “Delay based Congestion Control for Multipath tcp”, IEEE *International Conference on Network Protocols*, Dec.2012
[2] Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, “Architectural Guidelines for Multipath TCP Development,” RFC 6182, IETF, Mar. 2012.

Licensed Under Creative Commons Attribution CC BY

[3] M. Handley, U. Politechnica, A. Ford, C. Raiciu, of Bucharest, "TCP Extensions for Multipath Operation with Multiple Addresses draft-ietf-mptcp-rfc6824bis-00", Internet Engineering Task Force Oct.2013
[4] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, Oct. 2006.
[5] H. Y. Hsieh and R. Sivakumar, "pTCP: an end-to-end transport layer protocol for striped connections," in *Proc. of IEEE ICNP*, 2002, pp. 24–33
[6] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *Proc. of USENIX Annual Technical Conference*, 2004, pp. 99-112.
[7] Y. Dong, D. Wang, N. Pissinou, and J. Wang, "Multi-Path Load Balancing in Transport Layer," in *Proc. of 3rd EuroNGI Conference on Next Generation Internet Networks*, 2007, pp. 135–142.
[8] M. Handley, U. Politechnica, A. Ford, C. Raiciu, of Bucharest, "TCP Extensions for Multipath Operation with Multiple Addresses draft-ietf-mptcp-rfc6824bis-00", Internet Engineering Task Force Oct.2013
[9] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, Oct. 2006.
[10] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *Proc. of USENIX Annual Technical Conference*, 2004, pp. 99-112.
[11] Ermin Weiy, Asuman Ozdaglary, and Ali Jadbabaie "A Distributed Newton Method for Network Utility Maximization", Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, October 2012
[12] Steven H. Low, Senior Member, IEEE and David E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence", IEEE/ACM Transactions on Networking, 7(6), pp. 861-874, Dec. 1999.
[13] http://www.multipath-tcp.org
[14] http://en.wikipedia.org/wiki/Round-trip_delay_time
[15] http://www.nsnam.org
[16] http://en.wikipedia.org/wiki/Ns (simulator)

Author Profile



Sreejith N is doing M. Tech in Computer Science and Engineering at University of Calicut. He received B. Tech degree in Information Technology from University of Calicut in the year 2010.

Licensed Under Creative Commons Attribution CC BY