IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 13, Issue 4 (Jul. - Aug. 2013), PP 59-62
www.iosrjournals.org

A way of managing data center networks

[1]Shikha Soni, [2]Kreena Mehta, [3]Priyank Sanghavi,
[1]be-Extc, Djscoe., [2]be-It, Spit., [3]be-Extc, Djscoe.

***Abstract**: Evolutionary changes have occurred throughout the various elements of the data centre, starting with server and storage virtualization and also network virtualization. Motivations for server virtualization were initially associated with massive cost reduction and redundancy but have now evolved to focus on greater scalability and agility within the data centre. Data center focused LAN technologies have taken a similar path; with a goal of redundancy and then to create a more scalable fabric within and between data centres. The TCP's congestion control opens the network to denial of service attacks and performance interference.. We propose a natural evolution of data center transport from TCP to multipath TCP. We show that multipath TCP can effectively and seamlessly use available bandwidth, providing improved throughput and better fairness in these new topologies when compared to single path TCP and randomized flow-level load balancing.*

I. Introduction

Multipath transport protocols have the potential to greatly improve the performance and resilience of Internet traffic flows. The basic idea is that if flows are able to simultaneously use more than one path through the network, then they will be more resilient to problems on particular paths (e.g. transient problems on a radio interface), and they will be able to pool capacity across multiple links.

These multiple paths might be obtained for example by sending from multiple interfaces, or sending to different IP addresses of the same host, or by some form of explicit path control. Multipath-capable flows should be designed so that they shift their traffic from congested paths to uncongested paths, so that the Internet will be better able to accommodate localized surges in traffic and use all available capacity. In effect, multipath congestion control means that the end systems take on a role that is normally associated with routing, namely moving traffic onto paths that avoid congestion hotspots, at least to whatever extent they can given the paths they have available. When a flow shifts its traffic onto less congested paths, then the loss rate on the less congested path will increase and that on the more congested path will decrease; the overall outcome with many multipath flows is that the loss rates across an interconnected network of paths will tend to equalize. This is a form of load balancing, or more generally resource pooling. Multipath congestion control should be designed to achieve a fair allocation of resources. For example, if a multipath flow has four paths available and they all happen to go through the same bottleneck link, and if we simply run TCP's congestion avoidance independently on each path, then this flow will grab four times as much bandwidth as it should. In fact, the very idea of "shifting traffic from one path to another" there is some fair total traffic rate, and that extra traffic on one path should be compensated for by less traffic on the other.

II. Data center networking

From a high-level perspective, there are four main components to a data center networking architecture:

• Physical topology
• Routing over the topology
• Selection between the paths supplied by routing
• Congestion control of traffic on the selected paths

These are not independent; the performance of one will depend on the choices made by those preceding it in the list, and in some cases by those after it in the list. We will discuss each in turn, but it is worth noting now that MPTCP spans both path selection and congestion control, which is why it is able to offer benefits that cannot otherwise be obtained.

2.1 Topology

Traditionally data centers have been built using hierarchical topologies: racks of hosts connect to a top-of-rack switch; these switches connect to aggregation switches; in turn these are connected to a core switch.

Such topologies make sense if most of the traffic flows into or out of the data center. Uneven distribution of bandwidth takes place if most of the traffic is intra-data center, as is increasingly the trend. Recent proposals address these limitations. VL2 and FatTree are close [3] topologies that use multiple core switches to provide full bandwidth between any pair of hosts in the network. They differ in that FatTree uses larger quantities of lower speed (1 Gb/s) links between switches, whereas VL2 uses fewer faster (10Gb/s) links.

In contrast, in BCube [6], the hierarchy is abandoned in favour a hypercube-like topology, using hosts themselves to relay traffic. All three proposals solve the traffic concentration problem at the physical level — there is enough capacity for every host to be able to transmit out to another randomly chosen host. However the denseness of interconnection they provide poses its own problems when it comes to determining how traffic should be routed.

2.2 Routing

Many parallel paths between each pair of hosts are provided by dense interconnection topologies. We cannot expect the host itself to know which of these paths is the least loaded, so the routing system itself must spread traffic across these paths. The simplest solution is to use randomized load balancing, where each flow is assigned a random path from the set of possible paths. In practice there are multiple ways to implement randomized load balancing in today's switches. For example, if each switch uses a link-state routing protocol to provide Easy Cost Multi-Path (ECMP) forwarding then, based on a hash of 5 tuple in each packet, flows will be split roughly equally across equal length paths. VL2 provides just such a mechanism over a virtual layer 2 infrastructure. However, in topologies such as BCube, paths vary in length, and simple ECMP cannot access many of these paths because it only hashes between the shortest paths. A simple alternative is to use multiple static VLANs to provide multiple paths that expose all the underlying network paths [8]. Either the host or the 1st hop switch can then hash the 5 tuple to determine which path is used. In our simulations, we do not model dynamic routing; instead we assume that all the paths between a pair of endpoints are available for selection, whatever mechanism actually does the selection.

2.3 Path Selection

Randomised load balancing cannot achieve the full cross-sectional bandwidth in most topologies even though solutions such as ECMP or multiple VLANs provide the basis for randomised load balancing as the default path selection mechanism. The problem is that often a random selection causes hot-spots to develop, where an unlucky combination of random path selection causes a few links to be under load and links elsewhere to have little or no load. To address these issues, the use of a centralized flow scheduler has been proposed.

Large flows are assigned to lightly loaded paths and existing flows may be reassigned to maximize overall throughput [2]. The scheduler does a good job if flows are network-limited, with exponentially distributed sizes and Poisson arrivals, as shown in Hedera [2]. As dictated by the small number of flows we can have a small scheduling cost, even if we only schedule the big flows we can fully utilize all the bandwidth. However, data center traffic analysis shows that flow distributions are not Pareto distributed [5]. In such cases, the scheduler has to run frequently (100ms or faster) to keep up with the flow arrivals. Yet, the scheduler is fundamentally limited in its reaction time as it has to retrieve statistics, compute placements and instantiate them, all in this scheduling period. We show through simulation that a scheduler running every 500ms has similar performance to randomised load balancing when these assumptions do not hold.

2.4 Congestion Control

The job of matching offered load to available capacity on whichever path was selected is fairly done by most applications which use single path TCP, and inherit TCP's congestion control mechanism. Recent research has shown there are benefits from tuning TCP for data center use, such as by reducing the minimum retransmit makes MPTCP incrementally deployable, as it is designed to be fair to competing single path TCP traffic, unlike simply running multiple regular TCP flows between the same endpoints. In addition it moves more traffic off congested paths than multiple regular TCP flows would. Our hypothesis is that given sufficiently many randomly chosen paths, MPTCP will find at least one good unloaded path, and move most of its traffic that way.

In doing so, it will relieve congestion on the links that got more than their fair share of RLB-balanced flows. This in turn will allow those competing flows to achieve their full potential, maximizing the cross-sectional bandwidth of the network and also improving fairness. Hence worst case performance matters significantly.

III. Analysis

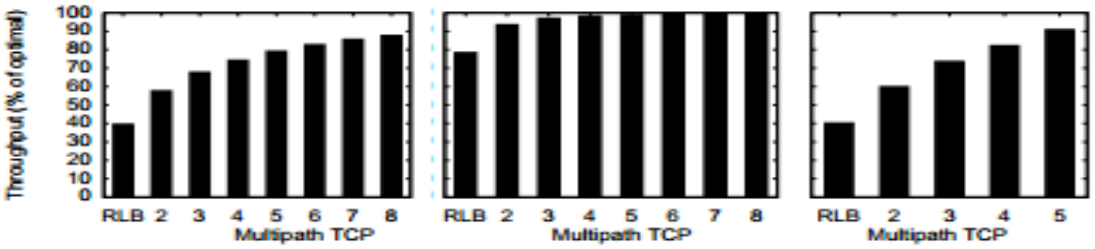
To validate our hypothesis, we must examine how MPTCP performs in a range of topologies and with a varying number of sub flows. We must also show how well it performs against alternative systems. To perform such an analysis is itself challenging - we really want to know how well such deployments will perform at large scale with real-world transport protocol implementations and with reasonable traffic patterns. Lacking a huge data center to play with, we have to address these issues independently, using different tools;

- Flow-level simulation to examine idealized large scale behaviour.
- Packet-level simulation to examine more detailed medium scale behaviour.
- Real-world implementation to examine practical limitations at small-scale.

3.1 Large scale analysis

The potential benefits of MPTCP with respect to the three major topologies in the literature are: FatTree, VL2 and BCube. The baseline for comparison is randomized load balancing RLB using single path TCP. MPTCP adds additional randomly chosen paths, but then the linked congestion control moves the traffic within each connection to the least congested sub flows. We use an iterative flow-level simulator to analyze topologies of up to 10,000 servers. The simulator computes the loss rates for each link based on the offered load, and adjusts the load accordingly. This simulator does not model flow start-up behaviour and other packet level effects, but is scalable to very large topologies. Fig. 1 shows the total throughput of all flows when we use a random permutation matrix where each host sends out (as determined by the TCP response function) to a single other host. In all three topologies, with the right path selection this traffic pattern should just be able to load the network to full capacity but not anything more than that. What we observe is that RLB is unable to fill any of these networks. It performs best in the VL2 topology, where it achieves 77% throughput, but performs much worse in FatTree and BCube. The intuition is simple: to achieve 100% capacity with RLB, no two flows should ever traverse the same link. With FatTree, when two TCP flows that could potentially send at 1Gb/s end up on the same 1Gb/s link, each backs off by 50%, leaving other links underutilized. With VL2, when eleven 1Gb/s TCP flows end up on the same 10Gb/s link, the effect is much less drastic, hence the reduced performance penalty. The benefits of MPTCP are clear; as additional sub flows are added, the overall throughput increases.

How many sub flows are needed depends on the topology; the intuition is as before - we need enough sub flows to overcome the traffic concentration effects of the random path allocation. One might think that the power of two choices [7] might apply here, providing good load balancing with very few sub flows. However it does not because the paths are not disjoint. Each sub flow can encounter a congested bottleneck on a single link along its path, causing the other links along the path to be underutilized. Although such bottleneck links are load balanced, with FatTree in particular, other links cannot be fully utilized, and it takes more than two sub flows to spread load across sufficient paths to fully utilize the network. This raises the question of how the number of sub flows needed scales with the size of the network. 90% of the cross sectional bandwidth is chosen as an arbitrary utilisation target, and then progressively the number of sub flows used are increased for different networks.. Fig. 2 shows the minimum number of sub flows that can achieve 90% utilization for each size of network. The result is encouraging: beyond a certain size, the number of sub flows needed does not increase significantly with network size. For VL2, two sub flows are needed.



(a) FatTree (b) VL2 (c)BCube

Figure 1

For FatTree, eight are needed. This might seem like quite a high number, but for an 8192-node FatTree network there are 256 distinct paths between each host pair, so only a small fraction of the paths are needed to achieve full utilization. From the host point of view, eight sub flows is not a great overhead. We also care that the capacity is allocated fairly between connections, especially for applications where the final result can only be returned when the last node running a part of a calculation returns its results. At very light load, there are few collisions, so MPTCP gives little benefits over RLB on FatTree or VL2 topologies. However on BCube, MPTCP excels because a single flow can use all the host interfaces simultaneously. At the other extreme, under overload conditions even RLB manages to fill the network, but MPTCP still gives better fairness. FatTree show substantial improvements over single-path RLB, even for very light or very heavy loads. This shows the performance benefits of MPTCP are robust across a wide range of conditions. The improvements for BCube are even greater at lower traffic loads. This is because BCube hosts have multiple interfaces, and MPTCP can use them all for a single flow at light loads the bottlenecks are the hosts themselves. Many flows collide on either the sending or receiving host, and MPTCP has no path diversity here. The 10Gb/s links are then not the bottleneck for the remaining flows under these load levels.

IV. Experimental evaluation

To test with our Linux implementation of MPTCP we used the topology below, which is essentially a trimmed down FatTree with two paths between each pair of endpoints Our aim is not to verify the simulation results (we don't have access to enough machines), but rather to examine the potential downsides to MPTCP that might arise through practical implementation issues.

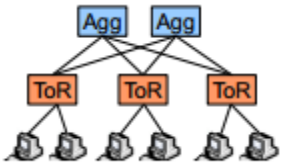


Figure 2

V. Conclusions

The modern data centers need adequate efficiency to function, and here MPTCP is a simple solution that can effectively utilize the dense parallel network topologies which significantly improves throughput and fairness compared to single path TCP. MPTCP unites path selection and congestion control. This flexibility allows the creation of cheaper and better network topologies: for instance, cheap multiport network interface controller (NIC) can be used between hosts allowing much higher throughputs when the network core is underutilized. For regular single path TCP to use such capacity it is very hard. As with any system, there may be some costs too: under heavy load, per-packet latency may increase due to timeouts; a little more memory is needed for receive buffers; and some complexity will be added to OS implementations. Our congestion control scheme is designed to be compatible with existing TCP behaviour. However, existing TCP has well-known limitations when coping with long high-speed paths. To this end, Microsoft incorporates Compound TCP in Vista and Windows 7, although it is not enabled by default, and recent Linux kernels use Cubic TCP [9]. We believe that Compound TCP should be a very good match for our congestion control algorithm. Compound TCP kicks in when a link is underutilized to rapidly fill up the pipe, but it falls back to New Reno-like behaviour once a queue starts to build. Such a delay-based mechanism would be complementary to the work described in this paper, but would further improve a multipath TCP's ability to switch to a previously congested path that suddenly has spare capacity. We intend to investigate this in future work.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In Proc. SIGCOMM 2010.
[2] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In Proc. Usenix NSDI 2010.
[3] C. Clos. A study of non-blocking switching networks. Bell System Technical Journal, 32(5):406–424, 1952.
[4] A. Ford, C. Raiciu, M. Handley, and S. Barre. TCP Extensions for Multipath Operation with Multiple Addresses. Internet-draft, IETF, 2009.
[5] A. Greenberg el al. VL2: a scalable and flexible data center network. In Proc. ACM Sigcomm 2009.
[6] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. Bcube: a high performance, server-centric network architecture for modular data centers. In Proc. SIGCOMM 2009.
[7] M. Mitzenmacher. The power of two choices in randomized load balancing. IEEE Trans. Parallel Distrib. Syst., 12(10):1094–1104, 2001.
[8] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul. Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In Proc. NSDI 2010.
[9] C. Raiciu, D. Wischik, and M. Handley. htsim. http://nrg.cs.ucl.ac.uk/mptcp/implementation.html. [10] V. Vasudevan et al. Safe and effective fine-grained tcp retransmissions for datacenter communication. In Proc. SIGCOMM 2009.