2015 The 5th International Workshop on Computer Science and Engineering
doi: 10.18178/wcse.2015.04.118

A Flexible Testbed for Datacenter Network Congestion Control Protocol

Xin Yue[1,a], Xing Li[1,b]

[1]Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China

[a]yuecn41@gmail.com；[b]xing@cernet.edu.cn

Keywords: testbed, congestion control protocol, datacenter network.

Abstract. With the prevalence of big data and cloud computing, both the network architecture and traffic pattern of the datacenter network changed. This brings challenge to the development of congestion control protocol for the datacenter network, as the traditional tools created for the wide area network and enterprise network don't meet the requirements of datacenter network well. In this paper, we will present our design and implementation of a flexible in-lab testbed for the development of datacenter network congestion control protocol. By using it to evaluate Transmission Control Protocol (TCP) and Data Center TCP (DCTCP), we demonstrate its flexibility and capability.

Introduction

During the first fifteen years of this century, we see the prevalence of web-based applications like Social Network Society (SNS) such as Facebook and Twitter, e-Bussiness such as Amazon and Taobao, and Cloud Computing such Amazon Web Service (AWS) and Aliyun. All these kind of services need powerful computing infrastructure to serve information to clients, needless to mention the traditional search service such as Google and Baidu. To lower cost and improve efficiency, companies tend to build their infrastructure based on server farm consisted of low-end commodity servers instead of supercomputers and one single datacenter of huge size instead of many ones of medium size. Today, the scale is so large that one single datacenter can accommodate about 10k or even 100k servers [1]. This brings new challenge to the state-of-art congestion control protocol TCP, which proved itself in Wide Area Network and Enterprise Network.

The challenge is twofold. On the one hand, the building block of datacenter network is different. 10Gbps and 40Gbps Ethernets are making their way into datacenter and replacing the 1Gbps Ethernet which TCP is tuned for. On the other hand, the traffic pattern of datacenter network is different. To meet the tight Service Level Agreement to satisfy user and ensure profit, compute paradigms like MapReduce [2] and Hadoop [3] are applied extensively. These cause the incast problem that TCP fails to maintain high utilization of bandwidth during this scenario.

There have already been several protocols proposed to address this challenge, like DCTCP [4], D3 [5], D2TCP [6], MPTCP [7], pFabric [8], and PDQ [9]. And we are sure to see more in the near future.

But the current tools for network protocol development fail to meet the requirements of congestion control protocol for datacenter network. The tools to evaluate network protocols fall into two categories: simulation and experiment. Simulation tools like NS2 [10], NS3 [11], OPNET [12], and NetSim [13] employ techniques to save memory and time consumed in the simulation. Thus, they are not strictly the same thing of reality. Experiment tools, like testbed PlanetLab [14] or Emulab [15], are not targeted at datacenter network. It could be time-consuming to build one such test case for datacenter network manually yet don’t fit the needs well.

In this paper, we design and implement a flexible in-lab testbed for the development of datacenter network congestion control protocol. Our contribution is threefold:

1. We describe and analyze the design principles of a flexible testbed for datacenter network congestion control protocol.
2. We compare the in-lab testbed design with other kinds of design like simulator-based and VPC-based (Virtual Private Cloud) and describe their pros and cons respectively.
3. We present the design and implementation of our testbed and present its capabilities by reproducing similar results of some published datacenter network congestion control protocols.

In the rest of this paper, we will proceed as following: first, in section 2, we will describe some design principles of such testbed that provides flexibility to the development of datacenter network congestion control protocol. Next in section 3, we will compare different ways to implement such kind of testbed according to these principles and analyze their pros and cons. Then we will introduce our implementation in section 4. And in section 5, we will present its capability by evaluating TCP and DCTCP. At last, we conclude this paper and discuss some future works in section 6.

Design principles

In this section, we will describe and analyze some design principles of a flexible testbed for datacenter network congestion control protocol.

We argue that a flexible testbed designed to assist the development of datacenter network congestion control protocol must provide flexibilities in following aspects.

Traffic: Different than wide area network and enterprise network, there are mice flows and elephant flow in the datacenter network. Mice flow indicates the flow generated from applications that employ a MapReduce compute paradigm, such as rendering the search results of search product or the news feeds of SNS products, which has a typical size about several KB. Elephant flow indicates the flow generated by applications to synchronize storage, update system states, migrate virtual machine images, and conduct backups, which has a typical size over 1MB [4]. A flexible testbed for datacenter network congestion control protocol should be able to produce both kinds of traffic.

Topology: There are many different topologies in the datacenter network. There are conventional topologies like mesh when synchronizing storage, dumbbell when conduct virtual machine migration, and star when mapping and reducing computation. There are also novel topologies like BCube [16], DCell [17], Fat-Tree [18], and Jellyfish [19]. A flexible testbed should be able to support as many as possible different kinds of topologies, at least enough to conduct the development for datacenter network congestion control protocol.

Control: Datacenter network is a single management domain charged by a single management entity. Network-agnostic is not a necessary requirement. Active Queue Management (AQM) techniques are employed extensively in the design of datacenter network congestion control protocol, like DCTCP [4] and D2TCP [6]. Some protocols go one step further to use customized switch, for example D3 [5], pFabric [8], PDQ [9] and so on. A flexible testbed shouldn't restrict the datacenter network congestion control protocol to be host-centric or switch-centric.

Tracing: Due to the existence of different kinds of traffic, there are different performance metrics employed to measure their performance. Mice flows are mainly measured by flow completion time because search or other kind of interactive services always have a tight SLA less than 200ms [20]. Longer flow completion time will cause profit loss or service quality degradation. But elephant flows are measured by throughput because it typically occurs in the storage synchronization which has looser completion time requirement. Moreover, many datacenter network congestion control protocols also have the need to tracking the queue length of switch to monitor the microscopic behavior. A flexible testbed should be able to trace all these kinds of data.

Scalability: In the MapReduce compute paradigm, master node sends query to its worker nodes. Each worker node prepares the response independently and then sends it back to the master node. The master will wait a short period of time to collect as many as possible responses to produce the final result. The number of worker nodes could ranges from dozens to hundreds. A flexible testbed should offer an efficient and cost effective way to provide enough parallel senders to fulfill this kind requirement of scalability.

Comparison of different ways of implementation

In this section, we will compare different ways of implementing flexible testbed for datacenter network congestion control protocol. We will describe and discuss their pros and cons. We argue that the in-lab way to implement a testbed provides more flexibility to the development of datacenter network congestion control protocol.

To implement testbed for network protocol, there are already successful projects like PlanetLab or Emulab. They employ a VPC-based (Virtual Private Cloud) method and deploy many nodes around the world. As the prevalence of cloud computing, Amazon and Alibaba also provide similar VPC services that can be used to build testbed for network protocol. This way helps to develop wide area network protocol but not datacenter network congestion control protocol. Comparing with the in-lab way, they provide different level of support of flexibility. We summarize them up in the Table 1. For clarity, we also include the simulator-based way to implement testbed, represented by NS3, OPNET, and NetSim. The aspects of comparing is based on the design principles we discussed in section 2, including authenticity which describe how close the result will be compared with the production environment.

Table 1: Different Ways Provide Different Extent of Flexibility

Flexibility	Simulator	VPC	In-lab
Traffic	high	high	high
Topology	high	medium	high
Control	high	medium	high
Tracing	high	medium	high
Scalability	high	high	high
Authenticity	low	medium	high

Simulator-based testbed provides low level authenticity because it often uses techniques to save time and memory consumed by the simulation. For example, NS3 uses virtual application payload, which cause the packet contains no payload just a number to indicate its size. The packet isn't actually transmitted through the wire. The simulator calculates the time needed to complete the transmission and then put the event that packet is received into the schedule queue of the simulator.

VPC-based testbed provides medium level of flexibility in aspects of control, topology, and tracing because it is difficult to have full control of its network devices. Therefore, it is difficult to trace the switch queue and even impossible to deploy some switch-centric datacenter network congestion control protocols. The switch features that a deployed protocol can take use of are restricted to the features the cloud service provider can provide. This will greatly limit the development of datacenter network congestion control protocol.

VPC also uses overly network to provide the topology flexibility. Different nodes in the same topology may be placed on the same physical server, or across rack or pod. This may contradict the semantics of the topology and thus lower the level of authenticity to medium.

Implementation

In this section, we will describe the implementation of our in-lab flexible testbed for development of datacenter network congestion control protocol.

In our testbed, a test case is conducted following the workflow showed in the Figure 1. The architecture of our testbed is showed in Figure 1, where a star topology is used to mimic a single rack in the datacenter.

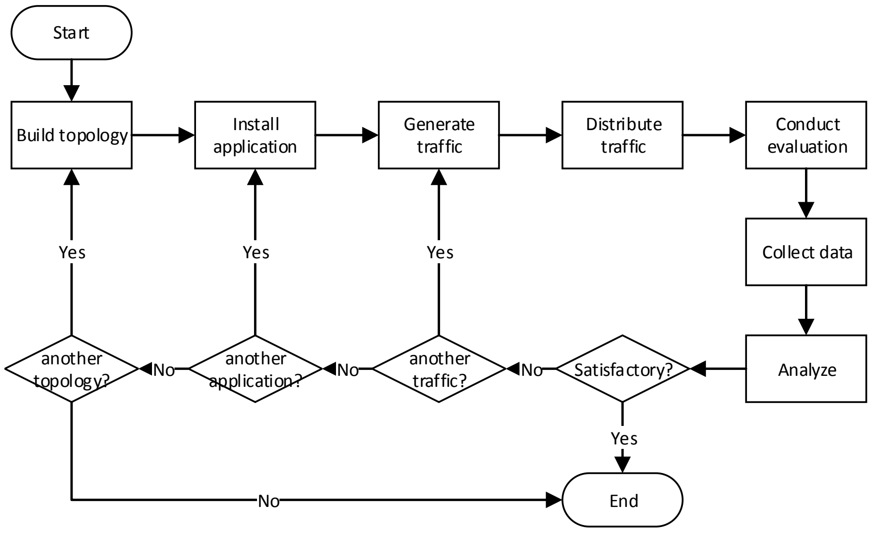

Fig 1: The Workflow of a Test Case

There are two kind of nodes in our testbed, the controller and worker. We use controller to manage the entire process of evaluation. For example, generate and distribute traffic t according to the specification of the test case. The evaluation of the datacenter network congestion control protocol is conducted among workers. The traced throughput and flow completion time are saved on corresponding workers. The traced data of queue length is saved on the host where the NetFPGA [21] card is installed. When the test finished, the controller will collect these traced data and proceed to do analysis. If the results are not satisfactory, the test can be conducted again by modifying the topology, application type, or traffic pattern.

We provide implementation of applications to mimic the MapReduce application which generates mice flow, and the storage application which generates elephant flow. The size and arrival interval of flow can be configured to be generated according to specific distribution, like uniform distribution, Poisson distribution, or lognormal distribution.

To get full control, we utilize NetFPGA as the platform to implement switch. It offers great flexibility to implement novel switch-centric datacentre network congestion control protocol. The NetFPGA offers four ports with 1Gbps or 10Gbps bandwidth. We cascade them to make more ports available. Different Ethernet is used to connect the controller to the switch to keep full access to it during the conducting of the test case. Thus, it is possible to retrieve the queue length dynamically.

In Figure 2, a single worker is used to mimic a single server in the datacentre. But to provide scalability, the worker can take use of a single process to mimic a single server. Thus, one worker is capable of mimicking hundreds of servers. It sacrifices authenticity, but it provides a more cost effective way to scale. When there is limitation on hardware, this method can be employed.

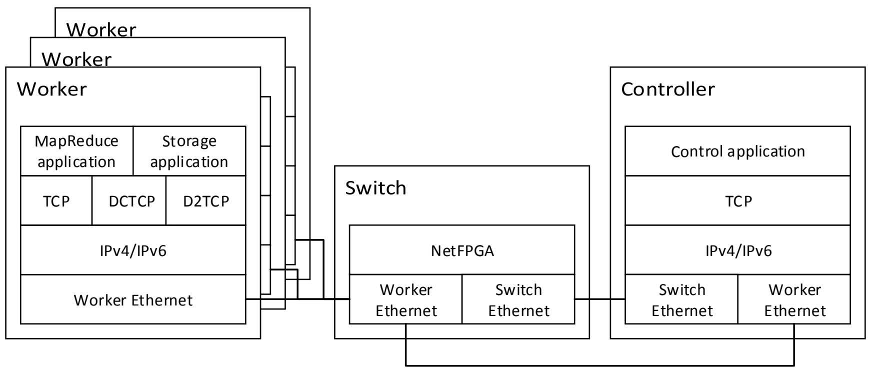

Fig 2: Testbed Architecture.

Controller node use different Ethernet network to control worker nodes and switch node. Thus, the control traffic don't interfere the generated traffic during evaluation.

Experiments

In this section, we will conduct the TCP incast [1,22,23] and DCTCP benchmark [4] experiments to demonstrate the capability and flexibility of our testbed.

The incast scenario compares performance of TCP Tahoe [24] and TCP NewReno [25]. One node requests a fixed block of data from multiple senders. As the number of senders increase, the throughput will drop. The result is showed in Figure 3. We set RTOmin as 200ms, use 1Gbps bandwidth, 1MB fixed block of data, and 64KB switch buffer.

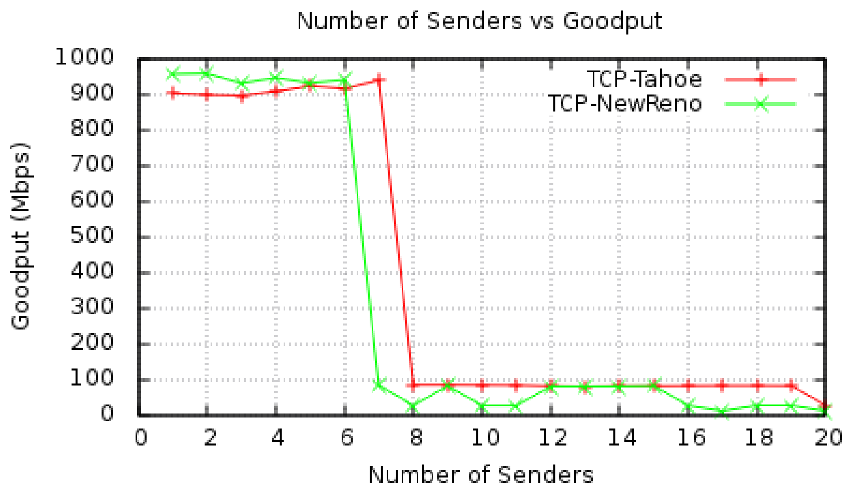

Fig 3: TCP Incast. RTOmin=200ms, Fixed Block=10MB, Bandwidth=1Gbps, Switch Buffer=64KB

DCTCP benchmark evaluates the performance of DCTCP. We generated the query traffic and background traffic according to the distribution described in [4]. Query flow spawned at every node will be sent to every other node in the rack and has a fixed size of 2KB. And background flow spawned at every node will be sent to a random node in the same rack. Its size is generated according the flow size distribution in [4]. The mean flow completion time is showed in Figure 4. We set K = 20 and RTOmin = 10ms, and use 12 senders.

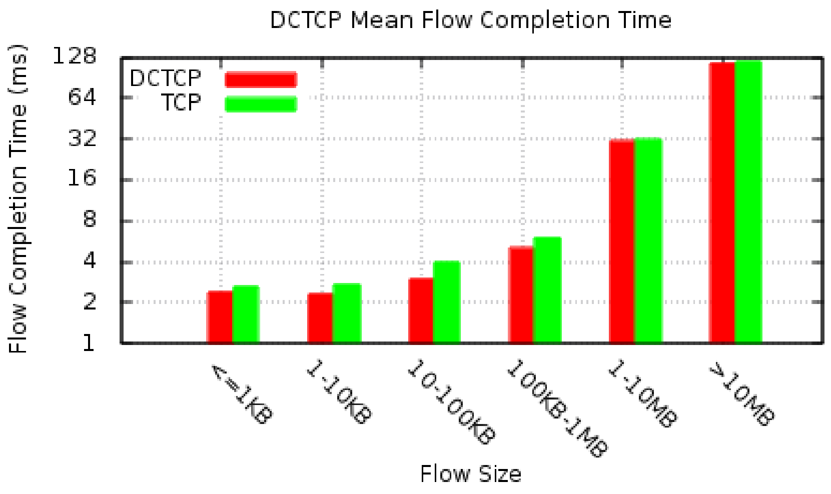

Fig 4: DCTCP Benchmark.

The benchmark lasts 10 minutes under the mixture of background traffic and query traffic, 12 senders involved.

Conclusion

In this paper, we design and implement a flexible in-lab testbed for the development of datacenter network congestion control protocol. We also describe and analyze the design principles of such kind of testbed. The datacenter network presents a different environment than wide area network and enterprise network, which cause the conventional tools inappropriate. By evaluating TCP and DCTCP, we demonstrate the flexibility and capability of our in-lab testbed.

References

[1] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen,G. R. Ganger, G. A. Gibson1, and B. Mueller, “Safe and Effective Finegrained TCP Retransmissions for Datacenter Communication,” in ACM SIGCOMM, pp. 303–314, Aug.2009..

[2] MapReduce: http://research.google.com/archive/mapreduce.html.

[3] Hadoop: http://hadoop.apache.org/.

[4] M. Alizadeh, A. Greenberg, D. A. Maltz, and J. Padhye, “Data Center TCP (DCTCP),” in ACM SIGCOMM, pp. 63–74, 2010.

[5] C. Wilson and T. Karagiannis, “Better Never than Late : Meeting Deadlines in Datacenter Networks,” in ACM SIGCOMM, pp. 50–61,2011.

[6] B. Vamanan, J. Hasan, and T. Vijaykumar, “Deadline-Aware Datacenter TCP (D2TCP),” in ACM SIGCOMM, 2012.

[7] C. Raiciu et al., “Improving Datacenter Performance and Robustness with Multipath TCP,” ACM SIGCOMM, 2011, pp. 265–76.

[8] M. Alizadeh et al., “Deconstructing Datacenter Packet Transport,” ACM Wksp. HotNet, 2012, pp. 133–38.

[9] C. Hong, M. Caesar, and P. Godfrey, “Finishing Flows Quickly with Preemptive Scheduling,” ACM SIGCOMM, 2012.

[10] NS2: http://www.isi.edu/nsnam/ns/.

[11] NS3: www.nsnam.org.

[12] OPNET: http://www.riverbed.com/products/performance-management-control/opnet.html.

[13] NetSim: http://www.boson.com/netsim-cisco-network-simulator.
[14] PlanetLab: http://www.planet-lab.org.
[15] Emulab: http://www.emulab.net/.
[16] C. Guo, G. Lv, D. Li, H. Wu, etc, “BCube: A High Performance, Server-Centric Network Architecture for Modular Data Centers”, ACM SIGCOMM 2009.
[17] C.Guo, H. Wu, K. Tan, L. Shi, etc “Dcell: A Scalable and Fault-Tolerent Network Structure for Data Centers”, ACM SIGCOMM 2008.
[18] M. Al-Fares, A. Loukissas, A.Vahdat, “A Scalable, Commodity Data Center Network Architecture”, ACM SIGCOMM 2008, pp. 63-74.
[19] A. Singla, C. Hong, L. Popa, P. Godfrey, “Jellyfish: Networking Data Centers Randomly”, NSDI, 2012, pp. 17-17.
[20] “Amazon Found Every 100ms of Latency Cost Them 1% in Sales,” http://blog.gigaspaces.com/amazon-found-every-100ms-of-latency-costthem-1-in-sales/, Aug., 2008..
[21] NetFPGA: http://netfpga.org/.
[22] Y. Chen, R. Griffith, J. Liu, R. Katz, and A. Joseph, “Understanding TCP Incast Throughput Collapse in Datacenter Networks,” in the 1st ACM workshop on Research on enterprise networking, pp. 73–82, 2009.
[23] J. Zhang, F. Ren, and C. Lin, “Modeling and Understanding TCP Incast in Data Center Networks,” IEEE INFOCOM, 2011, pp. 1377–85.
[24] M. Allman, V. Paxson, and E. Blanton, “TCP Congestion Control”, RFC 5681.
T. Henderson, S. Floyd, A. Gurtov, Y. Nishida, “The NewReno Modification to TCP’s Fast Recovery Algorithm”, RFC 6582.