Computer Communications 62 (2015) 34–46

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Congestion-aware adaptive forwarding in datacenter networks

Jiao Zhang [a,b,*], Fengyuan Ren [c], Tao Huang [a], Li Tang [c], Yunjie Liu [a]

[a] State Key Laboratory of Networking and Switching Technology, BUPT, China
[b] School of Information and Communication Engineering, BUPT, China
[c] Dept. of Computer Science and Technology, Tsinghua University, China

ARTICLE INFO

Article history:
Received 20 June 2014
Received in revised form 24 January 2015
Accepted 28 January 2015
Available online 13 February 2015

Keywords:
Datacenters
Adaptive forwarding
Goodput

ABSTRACT

Datacenters employ the scale-out model to achieve scalability. This model requires parallelism in the underlying workload. Therefore, high bisection bandwidth is required to support intensive communications between servers. Several new datacenter architectures have been designed to provide redundant bandwidth. Currently, it is critical to design a mechanism to efficiently utilize the abundant bandwidth. In this paper, we propose a distributed Congestion-Aware Adaptive foRwording (CAAR) protocol to balance traffic load only depending on the local queue length information. CAAR allows flows to select under-utilized paths to forward packets. It is theoretically proved to be stable if the arrival rates are within the network throughput region. Simulation results under diverse datacenter topologies and communication patterns validate that CAAR achieves higher aggregate goodput compared with random and static routing protocols.

1. Introduction

Most datacenters employ the scale-out model to achieve scalability [1]. One task is usually completed by many servers together to achieve higher performance. It is stated that 80% of the services in datacenters require 10–100 servers to cooperate and 20% require more than 100 servers [2]. Therefore, intensive communications between servers exist in datacenter networks. Several architectures with high bisection bandwidth[1] have been designed to provide the bandwidth resource required by the intensive communications [4,5,3]. However, how to efficiently use the bandwidth resource is still an open problem.

Unbalanced traffic load across different links results in inefficient bandwidth utilization. It is possible that several paths are congested while some others are under-utilized or even idle. Such situation will lead to unfairness among different flows. Flows along under-utilized paths will finish earlier than other flows with the same flow size but along congested paths. Since a lot of services are cooperated by many servers together, the response time is decided by the slowest flow. In such kind of services, it is desirable that a set of flows are finished near the same time. For example, in the Partition/Aggregation workflow [6–8], the aggregators need to collect all the results from the lower-level workers and then generate the final results. Therefore, the latency of the final result will be dragged by the slowest flow, or the quality of the result degrades without aggregating the results sent from the slow workers.

There are mainly two research directions to utilize the redundant bandwidth in datacenters. One is designing multipath transport layer protocols. The typical work of this type is MPTCP [9]. MPTCP utilizes redundant bandwidth by splitting a flow into multiple subflows. Each subflow adjusts its own congestion window. However, it is an end-to-end protocol and thus can only response to congestion on the magnitude of Round Trip Time (RTT). While most of flows have quite short length in datacenters [7]. Therefore, balancing load across multiple paths at the transport layer is not fast enough.

The other direction is employing multipath forwarding protocols. Hop-by-hop forwarding mechanisms have the potential ability to quickly shift traffic bursts from congested paths to under-utilized ones and thus fully utilize the redundant bandwidth. Some work has been done to balance load in datacenters. ECMP [10] is a widely known flow-level random forwarding protocol that could utilize multipath bandwidth. However, since flow sizes are various and communication patterns are diverse in datacenters, possibly some flows will be randomly routed to some congested links even if there are some other idle links. To overcome the drawbacks of ECMP, Hedera [11] makes use of the feature that

* Corresponding author at: Office 738, 3rd Teaching Building, Beijing University of Posts and Telecommunications, 100876, China.
E-mail addresses: jiaozhang@bupt.edu.cn (J. Zhang), renfy@csnet1.cs.tsinghua.edu.cn (F. Ren), htao@bupt.edu.cn (T. Huang), tangli@csnet1.cs.tsinghua.edu.cn (L. Tang), liuyj@chinaunicom.cn (Y. Liu).

[1] Bisection bandwidth denotes the minimal total bandwidth of the links to be removed to partition a network into two parts of equal size [3].

http://dx.doi.org/10.1016/j.comcom.2015.01.020
0140-3664/

the number of elephant flows in datacenters is small, and employs a centralized controller to reselect under-utilized paths for the long flows if they encounter congestion. However, in Hedera, the centralized controller has to collect flow-level information to find the elephant flows in real time, which brings large overhead. Recently, packet-level random forwarding is proposed to reduce the tail of the flow latency [12]. Since each packet can choose its own path, traffic load can be well balanced across multiple paths. However, packet-level random forwarding possibly leads to a lot of out-of-order packets. Larger memory at the end servers is required to accommodate the out-of-order packets [12].

In this work, we propose CAAR to purposely forward packets to balance the traffic with many short bursts across multiple links. The main idea is that each flow selects the most under-utilized path. If the selected path becomes congested during transmission, then the flow will be redirected to another under-utilized path. CAAR can responsively adapt to the traffic variation in datacenters and avoid packets reordering when no congestion happens.

Using theoretical analysis, it is proved that CAAR protocol is stable, and is throughput-optimal in terms of that it can fully utilize the redundant bandwidth. This theoretical result explores the possibility of making a tradeoff between protocol complexity and bandwidth utilization.

There are possibly a few out-of-order packets in CAAR since a flow will change its path if the current path becomes congested. In datacenters, the traffic is mixed of few long flows and a large number of short flows [13]. It is not necessary to redirect short flows since they can be finished quite quickly, and the probability of two long flows selecting the same path is quite small. Thus, we propose CAAR without reordering mechanism which does not reselect paths for flows during their transmission.

We implement both CAAR and CAAR without reordering on the ns-2 platform, and evaluate their performance in different datacenter topologies, including FatTree [4], VL2 [14], and single-rooted tree. The results show that CAAR performs much better than static routing and ECMP. Besides, CAAR without reordering performs close to CAAR.

The main contributions of this work have threefold.

(1) A congestion-aware adaptive forwarding protocol, CAAR, is proposed to fully utilize the redundant bandwidth in datacenters.
(2) Using theoretical analysis, CAAR is proved to be stable and throughput optimal.
(3) Considering the traffic characteristics, we propose CAAR without reordering mechanism and compares its performance with CAAR, static, and ECMP in various datacenter topologies.

The remainder of the paper is organized as follows. In Section 2, the related work and motivation is described. The network model is presented in Section 3. Section 4 presents the proposed scheme CAAR in detail. In Section 5, we prove that CAAR is stable when the arrival traffic is within the throughput region. Section 6 evaluates the proposed CAAR algorithm through simulations. Finally, the paper is concluded in Section 7.

2. Related work and motivation

2.1. Related work

Numerous forwarding/routing protocols have been developed for sorts of networks, including Internet, interconnection networks and datacenters. Here, some most related work in datacenters are summarized.

Static routing. Static routing is calculated in advance and keeps unchanged. In [4], Al-Fares et al. proposed a FAT-tree architecture for datacenters and designed a deterministic routing for FAT-tree topology. Guo et al. in [3] also proposed a deterministic source routing for their DCell architecture. In [15], Yuan et al. theoretically studied the deterministic routing for FatTree interconnection. They analyzed the lower bound of the oblivious performance ratio for different fat-trees and developed optimal deterministic single-path and multipath routing schemes in terms of oblivious performance ratio.

Random forwarding. Static routing is simple, but it cannot fully utilize the redundant links in datacenters. Greenberg et al. in [14] proposed balancing load by randomly spreading traffic across all the available links without considering any other factors. In [16], redundant paths are computed and then merged into a set of VLANs. Each packet is randomly sent to one of the VLANs that can reach the destination. However, random forwarding cannot perform well under some communication patterns and flow size distributions due to its blindness. In DeTail [12], packet-level randomized routing is proposed to reduce the tail of the flow latency. Traffic load can be well balanced across multiple paths since each packet can choose its own path. However, large memory at the end hosts is required to accommodate out-of-order packets [12].

Adaptive forwarding. Adaptive forwarding algorithms make a tradeoff between static routing and random forwarding. They employ some local information to change forwarding decisions to adapt to the traffic variation. The adaptive forwarding mechanisms can be classified into centralized and decentralized.

Hedera [11], MicroTE [17] and Fastpass [18] are typical centralized mechanisms. In Hedera [11]. short flows are randomly forwarded, while each large flow is assigned a under-utilized path computed by a heuristic algorithm to balance load. However, to differentiate whether a flow is large or not, switches need to record the size of every flow. Besides, the scheduler is fundamentally limited in its response time since it has to retrieve statistics, comput routing paths and install them. Whenever a flow's size exceeds a threshold or it persists for some time, it is considered to be a large flow. At last, Hedera assumes exponentially distributed flow sizes and Poisson arrival, which does not comply with the traffic characteristics of datacenters [19,14]. MicroTE [17] also employs a centralized controller as Hedera does. The centralized controller is used to track the predictable traffic between servers connected with the same ToR switch and route the traffic optimally. The remaining unpredictable traffic is then routed along weighted equal-cost multipath routes, where the weights reflect the available capacity after the predictable traffic has been routed. Various services are being developed in datacenters. It is difficult to predict traffic between servers. Also, the latency caused by the centralized controller [20] could not be neglected. Fastpass [18] uses a centralized arbiter to determine the time at which each packet should be transmitted as well as the path to use for that packet. It mainly focuses on guaranteeing zero-queue. The scalability of Fastpass is limited by the centralized arbiter that needs to deal with all the packets.

Many decentralized dynamic routing algorithms are designed for ISP, such as MATE [21], FLARE [22], and TeXCP [23]. MATE [21] converges slowly and works on the premise of knowing a global network information [23]. TeXCP [23] works in the granularity of a packet rather than a flow [24], which might lead to lots of packets reordering. FLARE [22] mainly solves the reordering problem when splitting a flow across multiple paths. It measures the delay of each path and set the maximum delay difference between the parallel paths as a threshold. Only if the interval of two packets exceeds the threshold, they can be transmitted along different paths. Thus, the packet-reordering problem can be avoided. However, since it is difficult to exactly measure the path delay of