EN.601.419/EN.601.619: Cloud Computing Spring 2020

EN.601.419/EN.601.619 Assignment 2

Mou Zhang

Discussed with Yifeng Yin and Liwei Cui

Assignment 2 Due: 5pm ET, April 19, 2020

Goal. The goal of this assignment is to review some of the key concepts related to cloud networking and big data systems.

Submission instructions. This assignment is due at the time listed above by email to soudeh@cs.jhu.edu. In your submission, explain your answer (e.g., an answer with a numerical result should show how you derived it).

- Subject: Assignment 2 Your-Name.
- Attachment format: a PDF file for your answers to the questions.

Collaboration policy. You're encouraged to discuss the assignment and solution strategies with your classmates. Your submission must clearly mention the names of all the students that you have discussed/worked with, as well as all the resources you have used. Your solution and submission, however, must be written by yourself, in your own words. Please see the policy on academic honesty and cheating stated in the course syllabus.

1. [10 points] We saw that the additive-increase/multiplicative-decrease (AIMD) algorithm, the simple distributed feedback control algorithm used in almost all congestion control algorithms today, has some key strengths such as simultaneously optimizing fairness and efficiency. Despite inheriting those strengths from AIMD, TCP's congestion control algorithm is not ideal for datacenters and we saw in the Jupiter paper that even for lightly loaded datacenters, managing congestion is still a big open research challenge.

   (a) Give a few (at least three) reasons why TCP is not effective for managing congestion in datacenters.

   (1) TCP tends to fill the queues, so congestion is inevitable with TCP.

   (2) TCP uses the AIMD to increase the flow, makes it slow to converge. This will cause waste in the datacenter.

   (3) There is no priority between different flows, so we can not distinguish important flows and give them higher throughput.

   (4) The flow in the datacenter is a mix of short flow and long flow, short flows tends to have low latency while the long flow wants the high throughputs.

(5) TCP global synchronization. When a burst occurs, each sender will reduce the number of packets they transfer, and as a result, the total transmission rate becomes low.

(b) Describe at least one remedy in the context of datacenters for improving the performance of TCP.

(1) Using ECN(Explicit Congestion Notification) to detect the congestion and send this message to the sender.

(c) Is this remedy consistent with, at odds with, or orthogonal to the end-to-end principle?

The remedy is consistent with the end-to-end principle. This is because that this change has nothing to do with the content of each packet.

2. [10 points] Suppose we build a cluster with a fat tree topology using 24-port switches, following the topology design of the Al-Fares paper (that you read and reviewed).

(a) How many distinct end-to-end shortest paths are there in the physical topology between a source server A and a destination server B in a different pod? (Here, shortest paths are those that have the minimum number of switches along the path. This question concerns the physical topology, not routing or forwarding.)

There are in total 1 (Host) * 1(Edge) * 12(Aggregation) * 12(Core) * 1(only one route from core to target host) = 144 distinct end-to-end shortest paths.

(b) Suppose the cluster runs BGP, with each switch having its own AS number and each edge switch announcing a distinct attached IP pre x covering all its servers. (No other prefixes are announced.) Switches run standard IP forwarding with ECMP across all shortest paths. How many forwarding rules are in each edge switch? How many are in each core switch? (Each forwarding rule specifies a single egress interface for a set of matching packets.)

In each edge switch, there are 24 forwarding rules. Since there are 12 ports connected with aggregation layer and 12 ports with hosts.

In each core switch, t there are 24 forwarding rules. Since each core switch connects to 24 switches in aggregation layer.

(c) Suppose the switch hardware is capable of performing MPLS forwarding, with IP-in-MPLS encapsulation, and the hardware can also still perform IP forwarding. More specifically, a forwarding rule on a switch can either (1) perform IP/ECMP forwarding as above, (2) match a pre x of IP packets and direct it into an MPLS tunnel, with ECMP-style hashing if there are multiple matching rules; (3) perform MPLS label swap forwarding; or (4) match an MPLS label, pop the MPLS header and continue forwarding out an interface via IP. Can you use this hardware to deliver data along the same paths as ECMP, but with fewer forwarding rules? If so, describe your solution and how many forwarding rules it needs at each edge switch and at each core switch.

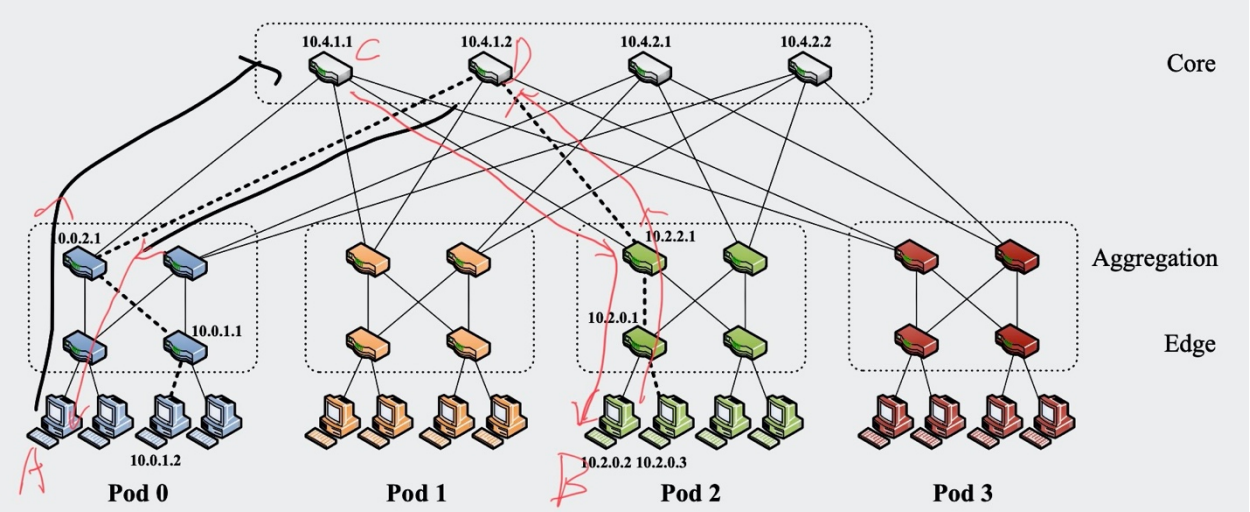
We can assign the label in MPLS the same way we do with hash in ECMP. We can just give the same label to a route in the cluster. Then if we are using this route, we don't need to remember the origin forwarding rules. We can just simply find out the switch with the same label and follow that label.

(d) Compare the resilience of your solution with that of ECMP. Specifically, suppose one core switch fails. What plausibly happens within about 1 millisecond, i.e., with only local reaction at each switch, in ECMP and in your solution? (Note: the goal isn't to produce a scheme which is necessarily better or worse than ECMP; the goal is just to compare.)

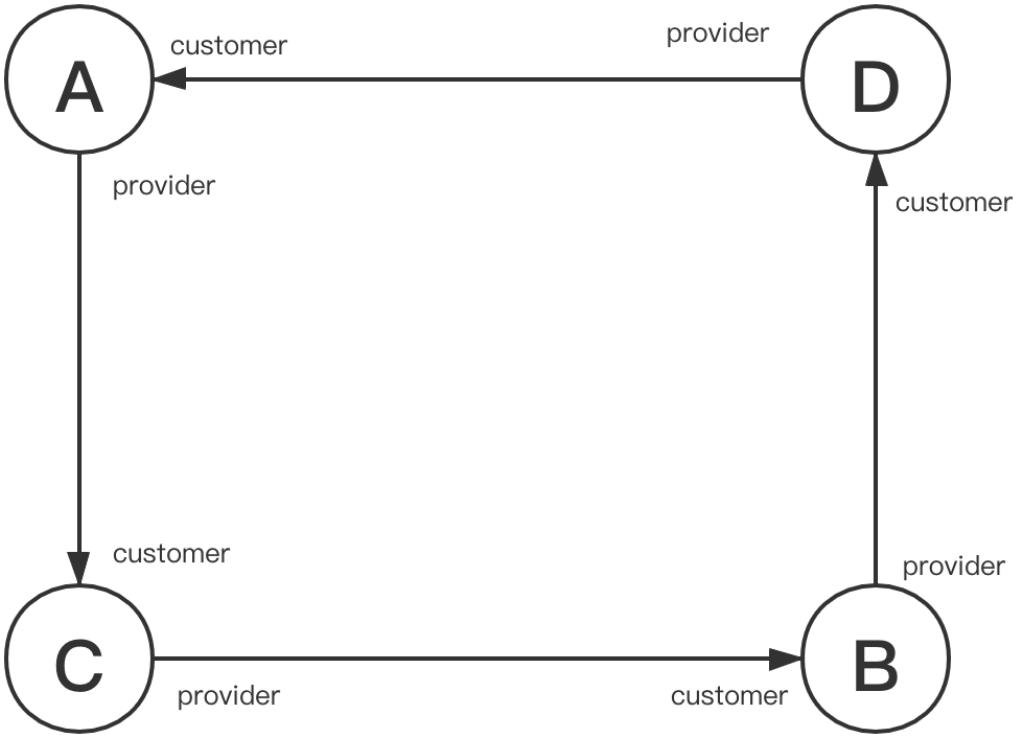
In ECMP, when a switch fails, it will detect the fault and send this packet to a new route. But in my algorithm, the switcha don't have this ability, so there will be a packet loss. ECMP has better resilience than my algorithm.

3. [15 points] Routes between A and B are asymmetric when the A B path is not the same as the B A path. How can asymmetric routing occur (a) inside a datacenter with a topology such as fat-tree that uses ECMP, and (b) between two datacenters even if all networks use BGP with common business relationship policies (prefer customer over peer over provider for route selection, and valley-free export)? Give two examples (one for (a) and one for (b)) of how this could happen.

(a)It is very common. For example, there are two switches in the core switches C and D. When A sends a packet to B, it goes through the switch C and get down to B. When B sends a packet to A, it goes through D and get down to A.

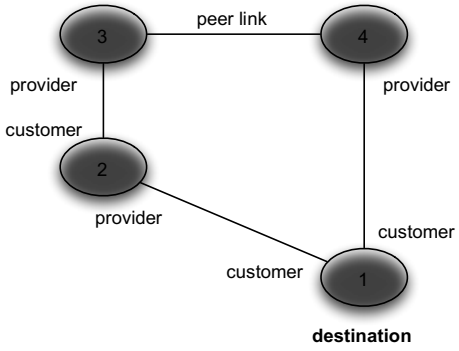


(b) For example, there are 2 nodes C and D. A is the provider of C while B is the customer of C. Therefore, when A goes to B, it goes through C. B is the provider of D while A is the customer of D. Therefore, when B goes to A, it goes through D. You can see a demo picture below.



4. [10 points] BGP routing can be formulated as a game where the selfish players are autonomous systems (ASes). It can be shown that this game has no Nash equilibrium (stable state) in some cases. That is, the control plane will keep switching routes even though the physical network is stable. In this problem, we'll see an example where there are two equilibria, and different sequences of events could lead to one or the other.

   Consider the network below:



   For simplicity, assume AS 1 is the only destination: it is the only AS that will originate an announcement message. The ASes have provider/customer/peer business relationships as shown. They follow the common route selection and export policies ... except that AS 1 is using AS 2 as a backup provider. This means AS 1 has instructed AS 2 to route to AS 1 via the link 2→1 only when no other path is available. (Incidentally, this is possible with BGP's community attribute.) The effect

is that AS 2 prefers to route through AS 3 in order to reach AS 1. Assume that at time 0, no routes have been announced by anyone. Shortly after time 0, AS 1 will begin announcing its IP pre x.

(a) Describe a sequence of events (BGP announcement messages and path selection decisions) that lead to one stable state.

(1) AS1 sends message to AS4, telling it that AS1 is its customer

(2) AS4 sends a message to AS3, telling it that AS1 is its customer.

(3) AS3 sends a message to AS4, telling it that AS2 is its customer.

(4) AS1 sends message to AS2, telling it that AS1 is its customer

(b) Describe a different sequence of events that lead to a different stable state.

(1) AS1 sends message to AS2, telling it that AS1 is its customer

(2) AS1 sends message to AS4, telling it that AS1 is its customer

(3) AS4 sends message to AS3, telling it that AS1 is its customer

(4) AS3 sends message to AS4, telling it that AS2 is its customer

(c) Suppose the network is now stabilized in state (a). A link fails; the BGP routers reconverge; the link recovers; BGP reconverges again; but now the network is in state (b) instead of state (a)! (This problem is sometimes known as a "BGP wedgie" because the system has gotten stuck in a bad state.) What sequence of events causes this story to happen? That is, which link failed, and which messages get sent?

The link between AS3 and AS4 failed. When this link fail, the AS2 cannot get to AS1 through AS 4, so as the backup plan, AS2 sends message to the AS1 directly. When the links recovers, AS2 does not know about the recover, so it's still going through the direct link between AS1 and AS2.

5. [15 points] Short questions.

(a) In the original OpenFlow paper that we read, a centralized controller receives the first packet of each flow and installs forwarding rules to handle that flow. However, a significant concern for any centralized design is scalability. Describe a feature of Google's Firepath SDN design that allows it to scale to large datacenters.

Google Firepath SDN has distributed the computing of the forwarding table with 2 main components. A Firepath client runs on each fabric switch and collects data. Firepath master collects data from clients and redistributes global link state to all switches. Switches locally calculate forwarding tables based on this current view of network topology. This has decreased the workload on the master, making it has better scalability.

(b) Consider a cluster of database servers. The time taken for any server to respond to any request is 10 milliseconds 99.8% of the time, and 200 milliseconds 0.2% of the time. Assume these times are sampled independently at random for each request. We have a front end web server that, when a user request arrives, queries 100 database servers in parallel. When all responses

are received, it can reply to the client. The web server itself is incredibly fast, so all its local processing always takes less than 1 millisecond. What is the probability that the web server needs ≥200 milliseconds to be ready to reply to the client? (As with all answers, briefy show how you calculated the answer.)

Web server needs ≥200 milliseconds only when responding 200 milliseconds occurs. The probability = 1 – the probability of all searching ends in 10 milliseconds = 1- $(99.8\%)^{100}$ = 1 – 0.8186 = 18.14%

(c) In the setting of the previous question, suppose the web server needs to perform two phases of processing before replying to the client. Phase 1 queries 100 database servers as described above. But now, after receiving all responses from phase 1, it can begin the second phase, where it queries another 200 database servers in parallel. Finally, when all responses are received, it can reply to the client. What is the probability that the web server needs to wait ≥200 milliseconds for its requests to complete?

Web server needs ≥200 milliseconds only when responding 200 milliseconds occurs. The probability = 1 – the probability of all searching ends in 10 milliseconds = 1 – the probability of all searching ends in 10 milliseconds in phase1 * the probability of all searching ends in 10 milliseconds in phase1 = 1 - $(99.8\%)^{300}$ = 1 – 0.5485 = 45.15%

6. [15 points] In a regular Clos datacenter, among the four types of resources (compute, memory, storage, and network), which ones are likely to become scalability bottlenecks for MapReduce and Spark? You will need to explain the workflows of these systems to answer this question.

(1) MapReduce: Network. The workflow of MapReduce is to split the work into separate map parts. Then send each map work to a worker, or mapper. Each worker computes its own part and return the result key-value pair to the reducers. The reducers aggregate/summarize/filter/transform with the result of mappers, and return the final results to the client. In the whole process, the transmission of data requires a lot disk IO and network IO, so the storage and network is the bottleneck of MapReduce.

(2) Spark: Memory. The workflow of spark is as follow: At first, the user's code containing RDD transformation forms DAG(Direct Acyclic Graph). Then the DAG will be split into stages of tasks by DAGScheduler. After that, the task scheduler will combine/shuffling/repartitioning the tasks and assign tasks to the workers. Finally tasks run on workers and return the result. Because in the whole process the data is in the memory, which is expensive and limited, memory is the main bottleneck of the spark.

7. [25 points] Peer-validation: One of the final tasks for the project is to evaluate a peer project. As you read the final reports, we need you to validate and comment on the reproducibility of the graphs in it. You will practice validating a project for this assignment using the Checkpoint 2 slides of a group assigned to you. The group-validator assignments will be posted on Piazza after the deadline of the second checkpoint.

- Write a brief summary (1 paragraph) of the project assigned to you: What problem are they going to solve? Why is it important? Their approach? Their progress so far?

  The problem they are solving is to set the connection and configuration of large networks. This is very important because a bad connection and configuration may lead to inefficiency and waste of resources. Their approach is to use genetic algorithm to automatically find out the best way to set the configuration of the network. They have almost achieved their goal. They have already implemented the algorithm with python code and they have already got a good result.

- Write a comment or question for that project.

  In the real datacenter, the flow of data may change rapidly. And the numbers of flows is very large. What are you going to do about the massive and rapidly changing data?

- Please use the following rubric to leave a score on the reproducibility of their results. Note that this is not their final grade. The teaching staff will be evaluating the projects and reproduce the results separately. This is just to give us another perspective and practice peer-validation.

  (a) Code ran to completion and results matched very well with the report.

- [Optional] Feel free to write some additional feedback in addition to the reproducibility score. We will also appreciate if you can comment on the quality of analysis and if you thought it was interesting.

8. [5 points] Optional/Extra credit. One of the reasons that MapReduce is slow is the barrier between the Map and the Reduce phases, i.e., no Reduce task can start before all Map tasks finish. Suppose a programmer writes a variant of MapReduce without a barrier, i.e., where Reduces can start before all Maps are done computing.

   - Demonstrate with an example that this MapReduce run may be incorrect, e.g., it may generate an incorrect value.

     Eg. The reducer tends to calculate the average value of numbers. If we have 3 mappers, and the output of each mapper is A, B, C. In the origin MapReduce, the reducer will calculate only when all mappers are done, so the result is (A + B + C) / 3. But in this variant of MapReduce, the B and C may be calculated first and reduced first, then A comes in the reducer and reduce again. So the result will be (A + (B + C) / 2) / 2, which is definitely incorrect.

   - Show how you can resolve this correctness issue while (partially) preserving the performance improvement of this barrier-less MapReduce.

     Let the controller set whether to use barrier-less MapReduce. The controller will make sure whether there will be error if using barrier-less MapReduce by analyzing the problem.