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

EN.601.419/EN.601.619 Assignment 1

Mou Zhang

Final Exam	Due: 12pm ET, May 12, 2020

Instructions. Please carefully read the instructions below. Failure to comply with any of the instructions below may result in us being unable to accept or grade your exam. These instructions apply from 9am, May 12 until 12pm, May 12. If you agree and follow the instructions, please print your name below and turn this page in with your exam. If you do not agree or follow the instructions, please do not turn in this exam. Exams without a signature will not be graded.

- This exam is openbook, which means you are permitted to use any materials from your own notes from the course, the papers you reviewed, and anything on the course website and its Piazza page.

- The exam must be taken completely alone. Showing it or discussing it with anyone is strictly forbidden.

- You may not consult with any other person regarding the exam. You may not check your exam answers with any person. You may not discuss any of the materials or concepts with any other person.

- You may not consult any external resources. This means no Internet searches, materials from other classes or books or any notes you have taken in other classes, etc. You may not use Google or any other search engines for any reason. You may not use any shared documents including Google documents.

- Your work must be original and you may not solicit or obtain assistance from or provide assistance to other people for any speci c content on the exam. Activities considered cheating include (but are not limited to) copying or closely paraphrasing content from websites, discussing exam questions with other students, and asking for help with speci c questions on Internet forums. All exams are checked for originality and copied content and anyone found cheating will be assigned a failing score for the exam.

- The exam should be submitted via email by 12pm on May 12, 2020 to the instructor: soudeh@cs.jhu.edu. Please email the instructor if you are unable to submit the exam at that time.

I have read and followed the instructions above.

_____

Name, Date

Section 1

Please answer all the questions in this section (100 points). Please remember to explain all your answers (e.g., an answer with a numerical result should show how you derived it).

1. [15 points] We saw in the Jupiter paper that large-scale datacenters are built as Clos, a topology that we had seen earlier in the Fat-tree paper. Discuss two advantages and two disadvantages of Clos-based datacenters compared to datacenter designs based on random graphs such as Jellyfish.

    Clos Advantages Comparing to Jellyfish:

    (1) Jellyfish is hard for cabling. The connection is random so we may need to connect some hosts with very long distance.

    (2) Jellyfish is hard to do congestion control. Since the structure is a chaos, it is hard to find where the congestion is in the network and hard to know which path is selected by router.

    Clos Disadvantages comparing to Jellyfish:

    (1) It is too rigid and not suitable for incremental expansion. For example, you have a fully constructed Clos which every switch is full and you want to add 1 new host. Then you have to change the whole structure.

    (2) Total throughput is lower than Jellyfish. This is because the average path length is lower in jellyfish due to its random structure.

2. [10 points] Suppose ASes run standard BGP routing policies (Gao-Rexford model), an undirected link between S_i and S_j means that S_i and S_j are peers, and a link from S_i to S_j ($S_i \rightarrow S_j$) means that S_i is a provider of S_j (equivalently, S_j is a customer of S_i). In the network below, will S_5 be able to reach S_9?
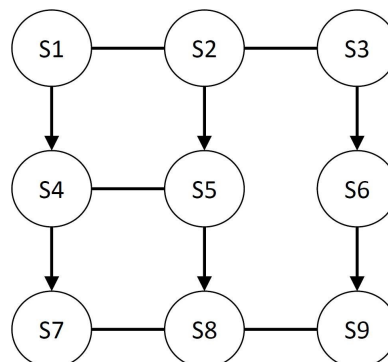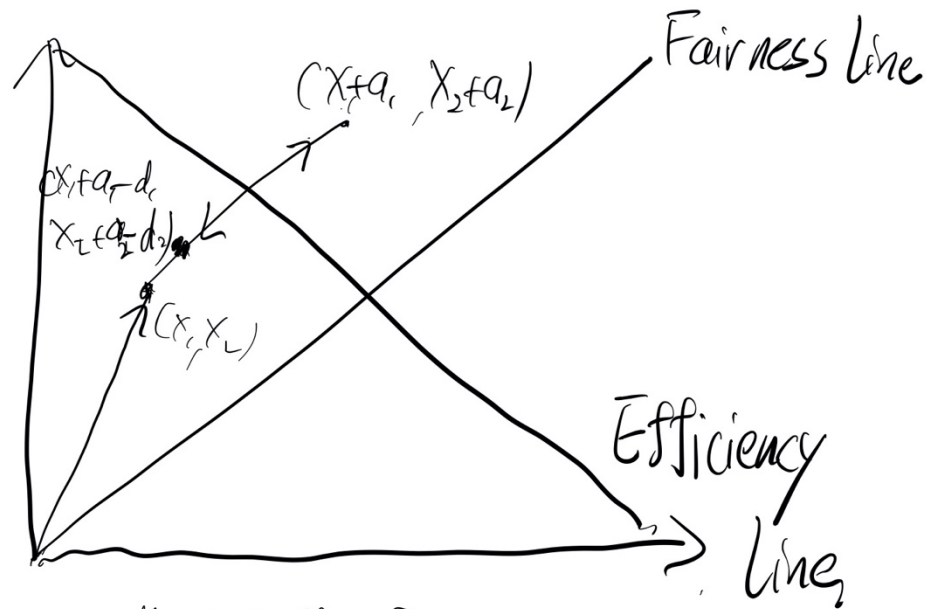

- If yes, what path does S_5 take to reach S_9? Compute the path stretch, defined as the ratio of the length of the BGP policy compliant path to that of the shortest path.

- If no, explain why several of the paths in the network (at least three) from S_5 to S_9 violate Gao-Rexford policies.
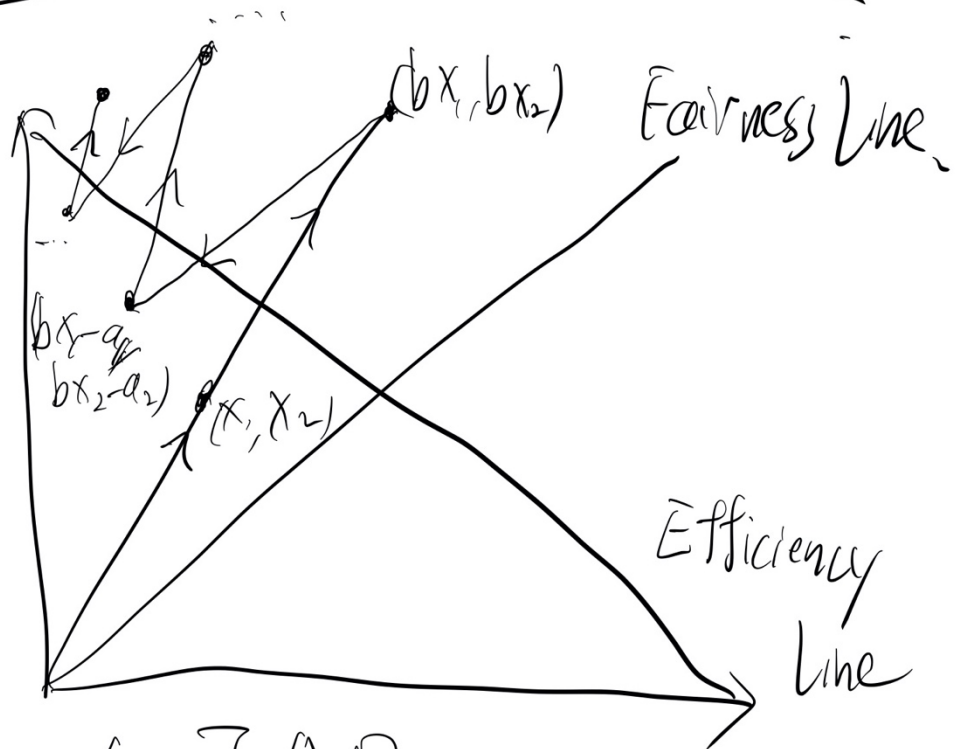
    Yes. The path is S5 -> s2 -> s3 ->s6 -> s9. The shortest path is S5 -> S8 -> S9. So the path stretch is 4 / 2 = 2.

3. [15 points] We saw that, similar to vanilla TCP, datacenter transport protocols such as DCTCP harness Additive Increase, Multiplicative Decrease (AIMD). This allows them to dynamically adjust their sending rates, in a fully distributed manner, while converging to both fairness and e ciency. From two possible alternatives to AIMD Multiplicative Increase, Additive Decrease (MIAD) and Additive Increase, Additive Decrease (AIAD) which one(s) converge to fairness and efficiency? Draw the Chiu Jain plots to demonstrate your answers.

AIAD converges to fairness and efficiency while MIAD not. The plots is shown below. The Chiu Jain plot, the cross of the two Fairness line and Efficiency line is the best condition with the best fairness and efficiency. As we can see from the plot. The AIAD serves similar to a loop and moves towards the cross while the MIAD moves away from the center.
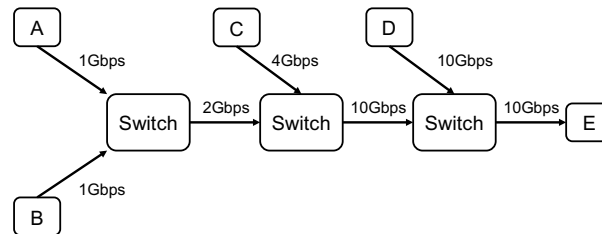
Fairness Line

$(X_1 + a_1, X_2 + a_2)$

$(X_1 + a_1 - d_1, X_2 + a_2 - d_2)$

(X_1, X_2)

Efficiency Line

AIAD

(bX_1, bX_2)

Fairness Line

$(bX_1 - a_1, bX_2 - a_2)$

(X_1, X_2)

Efficiency Line

MIAD

4. [15 points] What is the max-min fair bandwidth allocation in the network below? *A,B,C,D* are hosts that each wish to send at the maximum possible rate to *E*. The arrows are links and the numbers denote link capacities.

The max-min allocation is like follow:

A. 1Gbps

B. 1Gbps

C. 4Gbps

D. 4Gbps

5. [10 points] What does agility mean and why is it an important property in designing cloud infrastructures and systems? Discuss at least one technique that cloud providers deploy to maximize agility.

Agility means to use any server for any service at any time. It is important because it can increase utilization of servers and reduce costs. One technique is virtual machines. We use them for rapid installation of service's node. It makes use easier to set up the same environment rapidly so it increase agility.

6. [20 points] The OpenFlow paper that we read discusses the original design of SDN.

   • Analyze, in a qualitative manner, (a) scalability, (b) fault-tolerance, and (c) performance limitations of this design.

   (a) Scalability: The overall scalability is good. The scalability is limited by the memory of switches, since Openflow needs memory to store its property.

   (b) Fault-Tolerance:

   The fault tolerance is not good in Openflow. It is merely discussed in the original paper. It has no special mechanism for fault-tolerance. The controller is not designed to deal with fault tolerance well.

   (c) Performance:

   The overall performance of Openflow is good. Since all data in Openflow are controlled by the controller and does not interrupt with the original data, the performance of the network is based on the computation ability of the controller. If the controller runs slow, the performance is slow.

   • Discuss two approaches that cloud providers deploy to target and mitigate any of these limitations. You can discuss any SDN-based cloud systems and their solutions to SDN's original shortcomings.

- (1)The Scalability is increased in the later work like Jellyfish, it is easy to increase a new host by randomizing the path. And this has also increase the performance by lowering the average path length with random.

- (2)For 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.

- (3)The fault tolerance is much better in the later work. For example, In the Clos structure, it is very easy to find where fault happens because the structure is very clear. And every host has its own IP following the same rule. This makes it easy to find the host and switch to another route.

7. [15 points] Failure detectors are key building blocks of many cloud systems including key-value stores. We saw that it is impossible to build failure detectors that guarantee both completeness (each failure is eventually detected by at least one non-faulty node) and accuracy (no mistaken detection) in unreliable networks that occasionally drop packets.

- Which of these two properties completeness and accuracy practical cloud systems guarantee and why?

  The completeness means that each failure is eventually detected by at least one non-faulty node. The accuracy means that there is no mistaken detection. In practical cloud systems, the completeness is guaranteed because it is very important to know that there is a problem so the completeness has a higher priority. We tends to choose ways that guarantees the compleness.

- Explain how a Ring failure detector works and why it is not complete.

  Completeness means that each failure is eventually detected by at least one non-faulty node. It is not complete because if you can't receive heartbeating from 3 continuous nodes, you can never know whether the middle node of these 3 nodes is alive because no one can get its heartbeating even if it's alive.

- Why off-the-shelf gossip-based failure detectors can have poor accuracy in datacenters?

  Because when we are using gossip-based failure detectors, the failure message tends to go through many nodes before get detected. So the accuracy is poor.

Section 2: Peer Validation

8. [10 points] The final task for the project is to evaluate a peer project. Similar to the second assignment, we need you to validate and comment on the reproducibility of the results of the group project assigned to you. The group-validator assignments has been posted on Piazza.

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

- Write a comment or question for that project.

- Please use the following rubric to leave a score on the reproducibility of their results. Note that this is not their nal grade. The teaching sta 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.
  - (b) Code ran to completion and results matched fairly well with the report.
  - (c) Code ran to completion but results did not match the ones in the report, or all results in the report were not produced with the default run con guration.
  - (d) Code did not run to completion (exceptions, in nite loops, no progress/debug output/log output, etc.)
  - (e) Instructions were so vague or incomplete that the setup could not be completed to even start the code.
  - (f) Instructions not found.
- [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.

Section 3: Extra Credit

You can optionally answer at most one question in this section. If more than one question in this section is answered, we will grade only the first one.

9. [5 points] Describe a desired property in cloud computing that cannot be guaranteed with either Consistent Updates (CU) or Consensus Routing (CR). Give an example that shows a violation of this property despite using CR or CU. You can alternatively show how deploying CU (or CR) itself can occasionally violate this property.

    They both don't have the ability to check if there's any problem in the given new routing method. They can only deploy the new routing method. For example, I give out a routing algorithm to router C to let it sends directly to router D, but there is actually no router D. Then the router is stuck and confused even using CR and CU.

10. [5 points] In class, we saw the complete membership protocol in which each node retains the complete list of all other members of the group. An alternative membership protocol that is deployed in systems such as Cyclon is retaining a partial membership list at each node that is selected independently (for each node) and uniformly at random from all members. Compare, asymptotically, the expected propagation time of gossiping in a complete membership protocol and in a partial one.

11. [10 points] We analyzed the scalability and fault-tolerance of the push epidemic multicast protocol. In push epidemic multicast protocols, once a node is infected, it starts transmitting the message. Analyze the scalability of the pull epidemic multicast protocol where each node periodically polls a few randomly selected nodes for new multicast messages.