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

# EN.601.419/EN.601.619 Assignment 1

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 specific 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 specific 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.**

_____

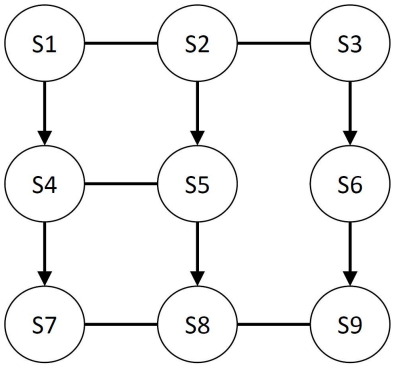
Name, Date

## 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.

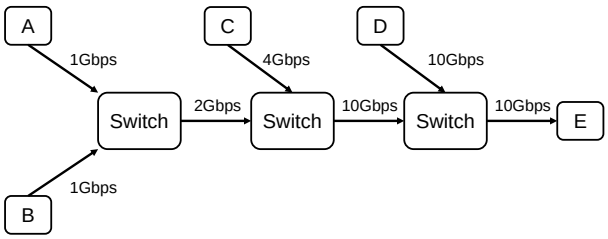
2. **[10 points]** Suppose ASes run standard BGP routing policies (Gao-Rexford model), an undirected link between $S_i$ and $S_j$ means that $S_i$ and $S_j$ are peers, and a link from $S_i$ to $S_j$ ($S_i \rightarrow S_j$) means that $S_i$ is a provider of $S_j$ (equivalently, $S_j$ is a customer of $S_i$). 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.

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 efficiency. 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.

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.



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.

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.
   - 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.

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?
   - Explain how a Ring failure detector works and why it is not complete.
   - Why off-the-shelf gossip-based failure detectors can have poor accuracy in datacenters?

## 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 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.
     (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 configuration.
     (d) Code did not run to completion (exceptions, infinite 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.

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.