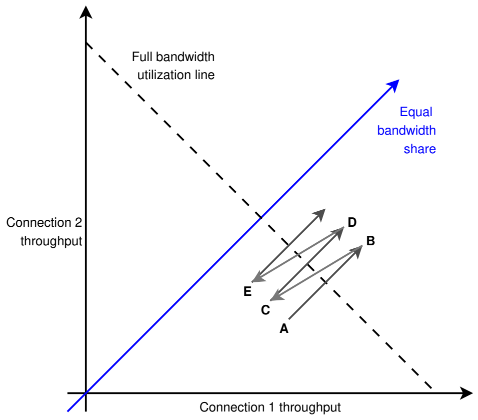
TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Intelligente Netze
Julius Schulz-Zander
Susanna Schwarzmann, Marcin Bosk

10th Assignment: Network Protocols and Architectures, WS 20/21



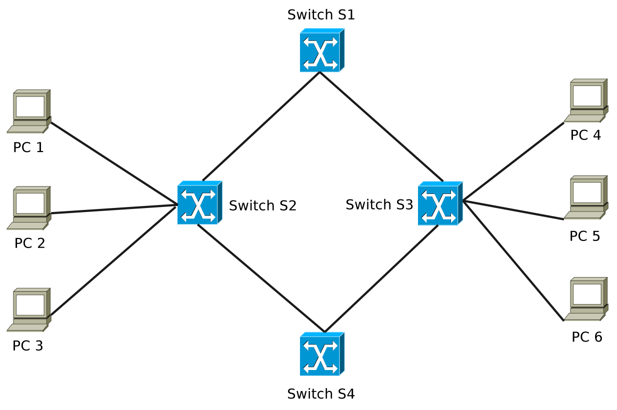
Question 1: (20 points) *Fairness of TCP: AIMD vs. AIAD*
Refer to the figure on the right which illustrates the convergence of TCP's additive-increase, multiplicative-decrease (AIMD) algorithm. The figure shows the throughput of the TCP connections 1 and 2. Suppose that instead of a multiplicative-decrease, TCP decreases the window size by a constant amount. Would the resulting **additive-increase, additive-decrease converge to an equal share** algorithm?
Justify your answer by drawing a **diagram similar to the figure** on the right. Additionally, **explain your diagram** in one or two sentences.

Question 2: (10 points) *Resource Allocation: Congestion Control*

Discuss what the Internet would look like if **every flow were based on UDP** and would not use congestion control.

Question 3: (10 points) *Switching Loops*

For redundancy, it might be a good idea to have a switch connected to several switches instead of only one. Have a look at the figure below and **explain why it can cause problems** having switches S2 and S3 connected to both switches S1 and S4 and **what can be done to prevent this**.



Please turn!

Question 4: (5 + 5 + 20 + 10 + 20 = 60 points) *A Web page load in detail*

In this exercise, you will have the opportunity to recap your knowledge of the Application Layer, Transport Layer, and Network Layer.

Assume that the ARP and IPv6 neighbor discovery caches are already filled, so you do not need to worry about the Link Layer in this question.

Consider a user who wants to **load the Web page** of `http://www.some-provider.org`.

The Web page consists of three resources:

- A base page hosted under `www.some-provider.org/`, with a size of 5KB.
- A stylesheet named `style.css`, with a size of 10KB, which is hosted on the same host as the base page. This file is referenced in the base page and contains some formatting information.
- An image named `banner.png`, with a size of 25 KB. This image is hosted on another Web server, `images.some-provider.org`, and is referenced in the base page.

Initially, the browser cache is empty. The browser first loads the base page using **unencrypted HTTP/1.1**. After it has finished loading the base page, it starts loading the two other resources, first the stylesheet and then the image. Again it uses unencrypted HTTP/1.1, and the transferred content is not compressed. The size of each **HTTP request is 200 Bytes**. The size of all **HTTP response headers** is **500 Bytes** in each response.

The **Maximum Segment Size is 1500 Bytes**, the **initial congestion window size is 4**. The hosts are using TCP Reno and sending cumulative acknowledgements, i.e., not every segment has to be ack'ed, but just the last of multiple segments. Assume that Reno is counting segments, not bytes. No packets are being lost. The receiver window size is not a limiting factor.

The **IPv4 address** of the computer that the user is using is `192.168.178.54`, the **IPv6 address** is `2001:db8:8765::54`. The user is using the public **DNS resolver** `9.9.9.9` over UDP over IPv4. The computer resolves **each host name to both IPv4 and IPv6 addresses**. It loads the **actual Web page over IPv6**. You can freely assign global IP addresses to the Web servers involved. The server running at `images.some-provider.org` is a different Web server than `www.some-provider.org`, and both host names resolve to different IP addresses.

If you need to make any further assumptions because something in this setup is unclear or ambiguous to you, please **state your assumptions** in your solution.

(a) **How many host names** does the user's computer have to resolve? **What DNS records are queried** for each host name, if the host resolves both IPv4 and IPv6 addresses?

(b) What is the **minimum number of TCP connections** that the user's computer has to open when loading the page, assuming that no packets are lost? Why?

(c) We will now start emulating a **packet trace** that the user makes in parallel to loading the Web page, using Table 1.

Step 1: Name resolution. The user resolves the host name of the base page, `www.some-provider.org`, using UDP and IPv4. Each query and each answer is in one UDP packet.

Fill in a table like Table 1 with the **DNS queries and answers** for this first name resolution step. This is UDP, not TCP, so you can leave the TCP fields empty. But fill in the application-layer payload with the DNS queries and answers.

Note: DNS has a well-known port number, which you need to use. As source port, the user's computer picks a random number between 1025 and 65535. Both queries can have the same source port.

Please turn!

Source IP address	Destination IP address	Source port	Destination port	TCP Flags	TCP Seq	TCP Ack	TCP Len	Application-Layer payload
...								
...								

Table 1: Packet trace

(d) **Step 2: Establishing a TCP connection.** Now that the computer knows the IP addresses, it establishes a TCP connection over IPv6, using the well-known destination port for HTTP. Picking a source port works similar to the previous sub-question.

Fill in the **three packets of the TCP handshake** in another table like Table 1. Take care to fill in the correct sequence numbers, acknowledgement numbers, and payload lengths.

As there is no Application-Layer payload here, you can leave it empty.

Hint: You can look at a packet trace using Wireshark to double-check whether your solution matches what your own computer is doing.

(e) **Step 3: Loading the base page.** The browser now requests the base page, hosted at `/` on the server `www.some-provider.org`, using HTTP.

Calculate how many bytes have to be sent from the user's computer to the server and back from the server to the user's computer, using the HTTP header sizes and the file sizes given above.

Each packet contains one TCP segment. If possible, each host sends as many bytes as it can in one segment, up to the Maximum Segment Size (MSS), and as many segments as it can according to TCP Reno with an initial congestion window of 4. If there are less bytes left to send than the MSS, or less segments to send than the congestion window size, the host sends as many bytes or segments as it can, and then waits for a reply.

After receiving the HTTP request, the server first acknowledges it on the TCP layer. In the next packet, it starts sending a response with the base page. Then, the client can either send one acknowledgement for every TCP segment it receives, or only one cumulative Ack for all TCP segments of the base page (your choice).

Consider what segments (and, thus, what packets) with how many bytes have to be sent in what order. Calculate the TCP sequence numbers, Ack numbers, and lengths accordingly.

Fill in **all TCP packets to transmit the HTTP request and response** for the base page (including acknowledgements) in another table like Table 1. Take care to fill in the correct sequence numbers.

For the Application-Layer payload, fill in the HTTP request type, the request URI (path of the base page), the HTTP status code, and whether the response contains headers and/or parts of the body (the actual contents of the base page).

Hint: Again, looking at a packet trace using Wireshark might help you. But be careful because header and body sizes will most likely be different in your trace.

Due Date: Wednesday, February 3rd 2021 11.59 pm (end of day)

- **As PDF files (no MS Office or OpenOffice files)**, uploaded via ISIS: `https://isis.tu-berlin.de/course/view.php?id=21979`
- Put the names and Student ID numbers (Matrikelnummer) of **all** your group members **and** the tutorial slot on your solution!