50.012 Networks (2021 Term 6)

Homework 1

Hand-out: 31 Jan

Due: 14 Feb 23:59

Name: Guo Ziniu Student ID: 1004890

1. Why two ISPs peer with each other? (Hint: consider the different types of peering: Regional ISP with Regional ISP, Tier 1 with Tier 1, and Regional ISP with content provider) What is the role of an IXP? How does an IXP generate revenue? (Hint: study some IXP, e.g., https://www.sgix.sg/)

ISP peering benefits:

Tier 1 ISP: It's their mutual interest to peer with each other since it makes more sense to connect users not only to their own networks but others as well.

Lower tier ISP / CDN:

increased reachability – internet is a network of networks, ISP has higher reachability when connected to more nodes

reduced latency – reduce latency caused by transiting through an intermediary IXP

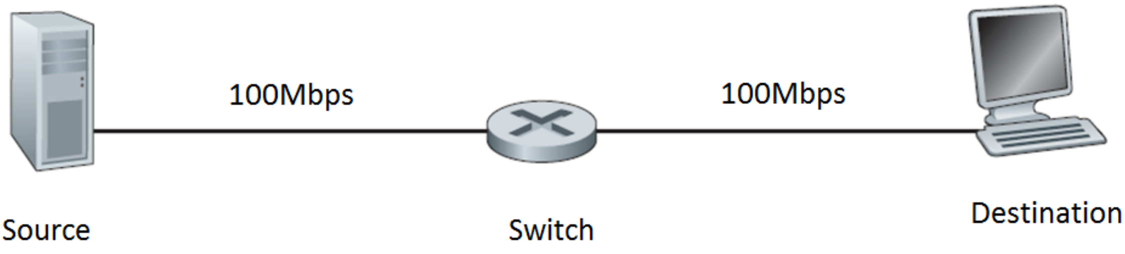
reduced costs – reduce transit cost through third party ISPs when exchanging traffic

The role of an IXP: provide connections between ISPs and CDNs.

IXP revenue: charge ISPs/CDNs for access to its service, which is connecting them together

2. (2019's mid-term exam question): Calculate the end-to-end delay (i.e., the duration from the first bit sent by the source to the last bit received by the destination) for a packet with size of 1500 bytes (12,000 bits) for the following settings:

2.1 The source and the destination are connected via a store-and-forward switch (as in the figure below). Assume that each link in the figure has a propagation delay of 12 µs (1 µs = 10^{-6}s) and operates at 100Mbps (consider 1M = 10^6). The switch begins forwarding immediately after it finishes receiving the whole packet. Assume zero processing and queueing delay.

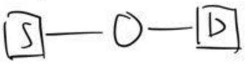


Single switch case

2.2 Same scenario as **2.1** above, calculate the end-to-end delay when there are four switches chained together (each switch and each link has the same setting as in scenario **2.1**) in the path between the source and the destination.

2.3 Same scenario as **2.1** above, i.e., there is only a single switch between the source and the destination, but assume the switch implements "cut-through" switching, i.e., the switch begins to forward the packet after the first 300 bits of the packet have been received. Calculate the end-to-end delay. (Hint: Draw a space-time diagram.)

(2.1)



Tranmitthion delay: $\frac{12000\ \text{bit}}{100 \times 10^{6}\ \text{bit/s}} \times 2 = 120 \times 2 \times 10^{-6}\,s = 240 \times 10^{-6}\,s$

propagation delay: $12 \times 10^{-6}\,s \times 2 = 24 \times 10^{-6}\,s$

Total delay: $2.64 \times 10^{-4}\,s$

(2.2) transmission delay: $\frac{12000\ \text{bit}}{100 \times 10^{6}\ \text{bit/s}} \times 5 = 120 \times 5 \times 10^{-6}\,s = 600 \times 10^{-6}\,s$

propagation delay: $12 \times 10^{-6}\,s \times 5 = 60 \times 10^{-6}\,s$

total delay: $6.6 \times 10^{-4}\,s$

(2.3) propagation delay: $24 \times 10^{-6}\,s$

transmithion delay: $\frac{12000\ \text{bit}}{100 \times 10^{6}\ \text{bit/s}} + \frac{300\ \text{bit}}{100 \times 10^{6}\ \text{bit/s}} = 120 \times 10^{-6} + 3 \times 10^{-6} = 1.23 \times 10^{-4}\,s$

total delay: $1.47 \times 10^{-4}\,s$

3. (adapted from textbook problem chapter 2, problem 4) Consider the following string of ASCII characters that were captured by Wireshark when the browser sent an HTTP GET message (i.e., this is the actual content of an HTTP GET message). The characters *<cr><lf>* are carriage return and line-feed characters (that is, the italized character string *<cr>* in the text below represents the single carriage-return character that was contained at that point in the HTTP header). Answer the following questions, indicating where in the HTTP GET message below you find the answer.

GET /cs453/index.html HTTP/1.1*<cr><lf>*Host: gaia.cs.umass.edu*<cr><lf>*User-Agent: Mozilla/5.0 (Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax) *<cr><lf>*Accept: ext/xml, application/xml, application/xhtml+xml, text/html;q=0.9, text/plain;q=0.8, image/png,*/*;q=0.5*<cr><lf>*Accept-Language: en-us,en;q=0.5*<cr><lf>*Accept-Encoding: zip,deflate*<cr><lf>*Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7*<cr><lf>*Keep-Alive: 300*<cr><lf>*Connection: keep-alive*<cr><lf><cr><lf>*

3.1 What is the URL of the document requested by the browser? (Hint: please look up the syntax rules of URL and includes all parts)

http://gaia.cs.umass.edu/cs453/index.html

3.2 What version of HTTP is the browser running?

HTTP 1.1

3.3 Does the browser request a non-persistent or a persistent connection? What are the pros and cons of using a non-persistent connection as compared to using a persistent connection?

keep-alive indicates that the connection is persistent.

Pros: conserve network resource, reduce network congestion and latency

Cons: for server, it requires more system resources. In most conditions, it requires content-length in the header as server needs to tell when the response ends

3.4 What is the IP address of the host on which the browser is running?

Not specified

3.5 What type of browser initiates this message? Why is the browser type needed in an HTTP request message?

Mozilla/5.0 (Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax)

It is included in the header so that the server can send response according to the browser's compatibility, including screen size and device type.

3.6 How many bytes are there for the request line (i.e., the first line in a request message, including *<cr><lf>*) in this message?

32 ASCII bytes

4. (textbook problem chapter 2, problem 5) The text below shows the reply sent from the server in response to the HTTP GET message in the question above. Answer the following questions, indicating where in the message below you find the answer.

HTTP/1.1 200 OK*<cr><lf>*Date: Tue, 07 Mar 2008 12:39:45GMT*<cr><lf>*Server: Apache/2.0.52 (Fedora)*<cr><lf>*Last-Modified: Sat, 10 Dec2005 18:27:46 GMT*<cr><lf>*ETag: "526c3-f22-a88a4c80"*<cr><lf>*Accept-Ranges: bytes*<cr><lf>*Content-Length: 3874*<cr><lf>*Keep-Alive: timeout=max=100*<cr><lf>*Connection: Keep-Alive*<cr><lf>*Content-Type: text/html; charset=ISO-8859-1*<cr><lf><cr><lf>*<!doctype html public "//w3c//dtd html 4.0 transitional//en">*<lf>*<html>*<lf>*<head>*<lf>* <meta http-equiv="Content-Type"content="text/html; charset=iso-8859-1"><lf> <meta name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT 5.0; U) Netscape]">*<lf>* <title>CMPSCI 453 / 591 / NTU-ST550A Spring 2005 homepage</title>*<lf>*</head>*<lf>* <much more document text following here (not shown)>

4.1 Was the server able to successfully find the document or not? What time was the document reply provided?

Yes, response status code is 200 OK.

Tue, 07 Mar 2008 12:39:45GMT

4.2 When was the document last modified?

Sat 10 Dec2005 18:27:46 GMT

4.3 How many bytes are there in the document being returned?

Content-length: 3874

4.4 What are the first 5 bytes of the document being returned? Did the server agree to a persistent connection?

<!doc

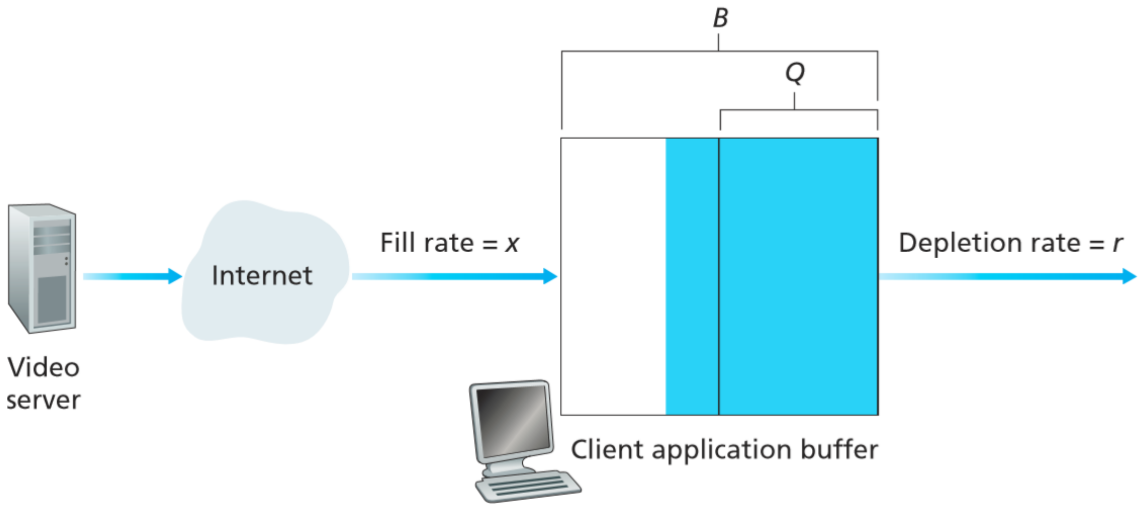
yes, but timeout seconds and max requests = 100

4.5 Explain how the "Last-Modified" and "ETag" information in the header lines can be useful.

Last-Modified: client may want to know whether the resource being looked for has changed since last time or not. So the client can add If-Unmodified-Since or

If-Modified-Since header to the request to let the server know, and the server can return status code 304 Not Modified. ETag works the same way, when If-Match or If-None-Match is specified in the request.

5. Recall the simple model for HTTP streaming shown in the Figure below. Let B denote the size of the client's application buffer, and let Q denote the number of bits that must be buffered before the client application begins playout. Also, let r denote the video consumption rate. Assume that the server sends bits at a constant rate x whenever the client buffer is not full.



5.1 Suppose that x < r. In this case playout will alternate between periods of continuous playout and periods of freezing. Determine the length of each continuous playout and freezing period as a function of Q, r, and x.

continuous playout duration: Q/(r-x)

freezing duration: Q/x

5.2 Now suppose that x > r. At what time does the client application buffer become full?

Assuming the initial condition is that the buffer is empty.

Q/x + (B-Q)/(x-r)

6. (2019’s midterm exam question) Consider distributing a file of F = 6 x 10^9 bits to N=100 peers. The server has an upload rate of u_s = 30 Mbps, and each peer has a download rate of d_i = 2 Mbps and an upload rate of u_i=1 Mbps. Assume 1M = 10^6. Calculate the minimum distribution time (i.e., to let every peer have a copy of the file) for:

6.1 the client-server distribution mode, and

NF/us = (100 * 6 * 10^9) / (30 * 10^6) = 2 * 10^4 (seconds)

F/min(d) = 6 * 10^9 / (2 * 10^6) = 3 * 10^3 (seconds)

minimun distribution time over client-server distribution mode is 2 * 10^4 seconds

6.2 the P2P distribution mode.

F/us = 2 * 10^2 = 200 (seconds)

F/min(d) = 3000 (seconds)

NF/(us + sum(ui))

= (100 * 6 * 10^9) / (30 * 10^6 + 100 * 1 * 10^6)

= 60/13 * 10^3 = 4615 (seconds)

minimum distribution time over P2P distribution mode is about 4615 seconds

7. Compare the basic design of the following application layer protocols: HTTP (you can focus on HTTP/1.1), MQTT, WebSocket, and SMTP.

7.1 List one common design aspect that all these application protocols adopt. Briefly explain the reason behind why they all choose this design.

They all use TCP.

Reason: All these connections need reliability

7.2 List a main design difference between HTTP and SMTP protocol, and briefly explain the reason behind this design difference.

HTTP is a pull protocol, because it is initially designed to retrieve data from a server, while SMTP is a push protocol that is used to send emails.

7.3 List a main design difference between MQTT and SMTP protocol, and briefly explain the reason behind.

MQTT is a pub/sub protocol. Producers push messages to consumers by MQTT.

7.4 List a main design difference between HTTP and WebSocket protocol, and briefly explain the reason behind.

HTTP is designed in request-response format while WebSocket is for real-time bi-directional communication, where data can be continuously pushed.

=== END ===