Answer

1-Since user A and B have fixed request rate and each request has the fixed length of 1Mbit, the remaining link capacity on the link would be equal to:

20Mbit/sec-12Mbit/sec=8Mbit/sec

According to the question the probability of user A to have request probability of 4 equals to 0.5 so we will have:

$$0.5\sum_{k=0}^{4}\frac{\sqrt{\mathrm{e}}-1}{\sqrt{e}}e^{-0.5n}+0.5\sum_{k=0}^{8}\frac{\sqrt{\mathrm{e}}-1}{\sqrt{e}}e^{-0.5n}=$$

$$1-\frac{e^{-2.5}+e^{-4.5}}{2}$$

---

2)

a) Yes, the HTTP server is stateless so it sends requested files to clients without storing any state information about the client; this simplifies server design and has permitted engineers to develop high-performance Web servers that can handle thousands of simultaneous TCP connections.

b) The benefit of using cookies: It is often desirable for a Web site to identify users, either because the server wishes to restrict user access or because it wants to serve content as a function of the user identity.

Function: When the request comes into a specific Web server, the server creates a unique Identification number and creates an entry in its back-end database that is indexed by the identification number. The mentioned Web server then responds to the browser, which is requesting a file from the server, including in the HTTP response a Set-cookie: header, which contains the identification number. When the browser receives the HTTP response message, it sees the Set-cookie: header. The browser then appends a line to the special cookie file that it manages. This line includes the hostname of the server and the identification number in the Set-cookie: header. As the user proceeds to browse the Amazon site, each time he/she requests a Web page, his/her browser consults her cookie file, extracts her identification number for this site, and puts a cookie header line that includes the identification number in the HTTP request.

Components: cookie technology has four components: (1) a cookie header line in the HTTP response message; (2) a cookie header line in the HTTP request message; (3) a cookie file kept on the user's end system and managed by the user's browser; and (4) a back-end database at the Web site.

c)
Web caching has seen deployment in the Internet for two reasons. First, a Web cache can substantially reduce the response time for a client request. Second, Web caches can substantially reduce traffic on an institution's access link to the Internet.

3)

Note that each downloaded object can be completely put into one data packet. Let Tp Denote the one-way propagation delay between the client and the server.
First consider parallel downloads using non-persistent connections. Parallel downloads Would allow 10 connections to share the 150 bits/sec bandwidth, giving each just 15 Bits/sec . Thus, the total time needed to receive all objects is given by:

(200/150+*T*p + 200/150 +*T*p + 200/150+*T*p + 100,000/150+ *T*p)
+ (200/(150/10)+*T*p + 200/(150/10) +*T*p + 200/(150/10)+*T*p + 100,000/(150/10)+ *T*p)
= 7377 + 8**T*p (seconds)

Now consider a persistent HTTP connection. The total time needed is given by:

(200/150+*T*p + 200/150 +*T*p + 200/150+*T*p + 100,000/150+ *T*p)
+ 10*(200/150+*T*p + 100,000/150+ *T*p) =7351 + 24**T*p (seconds)

Assuming the speed of light is $300*10_6$ m/sec, then Tp=10/($300*10_6$)=0.03 microsec. Tp is therefore negligible compared with transmission delay.
Thus, we see that persistent HTTP is not significantly faster (less than 1 percent) than the non-persistent case with parallel download.

---

4)
a)
HTTP message format on the client side:

Get	**SP**	**Dorsapax/Data/Sub_27/ File/taghvim.amozeshi98.pdf**	**sp**	**HTTP/ 1.1**	**cr**	**lf**
Host:	**SP**	**http://www.kntu.ac.ir/**	**cr**	**Lf**		
Connection:	**SP**	**Close**	**cr**	**Lf**		
User agent:	**SP**	**Chrome/77.0.3865.120**	**cr**	**Lf**		
Accept-language:	**SP**	Fr	**cr**	**Lf**		

HTTP message format on the server side:

HTTP/1.1	**SP**	**200**	**sp**	**Ok**	**cr**	**lf**
Connection:	**SP**	**Close**	**cr**	**Lf**		
Date:	**sp**	**Mon, 14 Oct 2019 21:00:00 GMT**	**cr**	**Lf**		
Server:	**sp**	**IIS/7.5**	**cr**	**Lf**		
Content	**Sp**	**118**	**cr**	**lf**		

length:				
Content type:	**sp**	**pdf**	**cr**	**Lf**
(data,data,data,…..)				

b) If there is no such a file on the server the format of the server message would be like below:

HTTP/1.1	**SP**	**404**	**sp**	**Not found**	**cr**	**Lf**
Connection:	**SP**	**close**	**cr**	**Lf**		
Date:	**Sp**	**Mon, 14 Oct 2019 21:00:00 GMT**	**cr**	**Lf**		
Server:	**Sp**	**IIS/7.5**	**cr**	**Lf**		
Content length:	**Sp**	**118**	**cr**	**Lf**		
Content type:	**Sp**	**pdf**	**cr**	**Lf**		
(data,data,data,…..)						

c)
To find the destination IP address using the destination host name, DNS servers should be consulted, to do so, The client first contacts one of the root servers, which returns IP addresses for TLD servers for the top-level domain com. The client then contacts one of these TLD servers, which returns the IP address of an authoritative server for kntu.ac.ir finally, the client contacts one of the authoritative servers for kntu.ac.ir, which returns the IP address.

d)
1)In the first case that there was no previous request for the same file, the exchanged messages between client, proxy server and the Web server could be as below:

The message sent from client:

GET /Dorsapax/Data/Sub_27/File/taghvim.amozeshi98.pdf/HTTP/1.1
Host: www.kntu.ac.ir
Connection: close
User-agent: Chrome/77.0.3865.120
Accept-language: fr

The message sent from proxy server to Web server:

GET /Dorsapax/Data/Sub_27/File/taghvim.amozeshi98.pdf/HTTP/1.1
Host: www.kntu.ac.ir

The message sent back to proxy server from Web server:

HTTP/1.1 200 OK
Date: Mon, 14 Oct 2019 21:00:00 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 7 Feb 2019 09:23:24
Content-Type: pdf
(data data data data data ...)

The message sent back from proxy server to the client:
HTTP/1.1 200 OK
Date: Mon, 14 Oct 2019 21:00:00 GMT
Server: IIS/7.5
Last-Modified: Wed, 7 Feb 2019 09:23:24
Content-Type: pdf
(data data data data data ...)

2.
Assuming another client request the same file from the proxy server:

The message sent from client:

GET /Dorsapax/Data/Sub_27/File/taghvim.amozeshi98.pdf/HTTP/1.1
Host: www.kntu.ac.ir
Connection: close
User-agent: Chrome/77.0.3865.120
Accept-language: fr

The message sent from proxy server to Web server:
GET /Dorsapax/Data/Sub_27/File/taghvim.amozeshi98.pdf/HTTP/1.1
Host: www.kntu.ac.ir
If-modified-since: Wed, 7 Feb 2019 09:23:24

The message sent back to proxy server from Web server:
HTTP/1.1 304 Not Modified
Date: Mon, 14 Oct 2019 22:00:00 GMT
Server: Apache/1.3.0 (Unix)
(empty entity body)

The message sent back from the proxy server to the client:
HTTP/1.1 200 OK
Date: Mon, 14 Oct 2019 22:00:00 GMT
Server: IIS/7.5
Last-Modified: Wed, 7 Feb 2019 09:23:24
Content-Type: pdf
(data data data data data ...)

---

5) a) Two scenarios could be possible in this case:

1. Using FTP protocol to put data in the server and getting the data from server using FTP protocol.
2. Using FTP protocol to put data in the server and getting the data from server using HTTP protocol.

In both scenarios at the first place we should put the date in the FTP server, obviously by the use of FTP protocol; so, we need to interact with FTP through an FTP user agent. We should first provide the hostname of the remote host, causing the FTP client process in the local host to establish a TCP

connection with the FTP server process in the remote host. The user then provides the user identification and password, which Are sent over the TCP connection as part of FTP commands:

USER username: Used to send the user identification to the server.

And then then server would probably reply with:
331 Username OK, passwords required

PASS password: Used to send the user password to the server.

these commands are sent over the TCP connection as part of FTP commands. Once the server has Authorized the user, the user copies one or more files stored in the local file system into the remote file system using below command:

STOR filename: Used to store (that is, put) a file into the current directory of the remote host.

And then on the second place our friend should get the file using RETR command in FTP or simply requesting the file by the use of HTTP protocol:

In the first scenario the exchanged messages would be like what has been explained above for putting data in the FTP server but the difference is that because our friend wants to get data from server, RETR command should be used, in the second scenario the exchanged messages would be like below:

HTTP request message:

GET /somedir/page.html HTTP/1.1
Host: www.xlightftpd.com
Connection: close
User-agent: chrome/75.0
Accept-language: Eng

Server may replies with:
HTTP/1.1 200 OK
Connection: close
Date: mon, 14 Oct 2019 21:00:00 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2015 15:11:03 GMT
Content-Length: 852
Content-Type: text/html
(data data data data data ...)

---

6)

For sending such an email the SMTP would be deployed as follow:

Imagine our friend's name is Sergio and his mail address is Sergio@yahoo.com :

First we should invoke our user-agent and input Sergio's mail address in it, the message format could be as follow:
From: B.gheysari04@gmail.com
To: Sergio@gmail.com
Subject: congratulations
Body: " Happy new year! "

And then our user agent sends the message to her mail server, where it is placed in a message queue. After that, the client side of SMTP sees the message in the message queue. It opens a TCP connection to an SMTP server, running on Sergio's mail server, when the connection is stablished successfully, some SMTP handshaking is needed and after that the SMTP client sends our message into the TCP connection. The exchanged messages at this stage are as follow:

S: 220 yahoo.com
C: HELO gmail.com
S: 250 Hello gmail.com, pleased to meet you
C: MAIL FROM: <b.gheysari04@gmail.com>
S: 250 b.gheysari04@gmail.com ... Sender ok
C: RCPT TO: <sergio@gmail.com>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: happy new year
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 yahoo.com closing connection

Which results Sergio's mail server to receive the message ? Sergio's mail server then places the message in Bob's mailbox.
finally Sergio could invoke his user agent to read the message by the use of protocols such as POP3 and IMAP, at his convenience.

---

7)

For calculating the minimum distribution time for client-server distribution, we use the following formula:

D_{cs}=max {NF/u_s, F/d_{min}}, here we already know that d_{min}= d_i.

Similarly, for calculating the minimum distribution time for P2P distribution, we use the Following formula:

D_{P2P}=max {F/u_s, F/d_{min}, $NF/$ (u_s+$\sum_{i=1}^{N}$ ui)}

As they are given clearly in the question, we already know that

File Size, F = 15Gbits; Server upload rate u_s=30$Mbps$;

Each Peer download rate: d_{min}= d_i= 2Mbps. Putting all together:

CLIENT-SERVER:

N

Client Upload Rate	**10**	**100**	**1000**
u=300Kbps	7680	51200	51200
u=700Kbps	7680	51200	51200
u=2Mbps	7680	51200	51200

P2P:

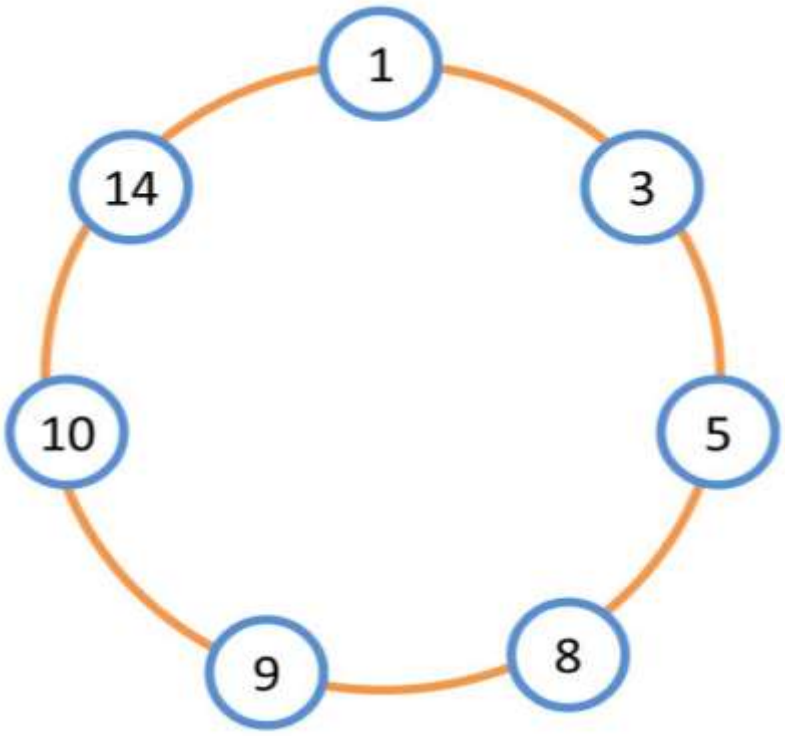
N

Client Upload Rate	10	100	1000
u=300Kbps	7680	25904	47559
u=700Kbps	7680	15616	21525
u=2Mbps	7680	7680	7680

---

8)

Assume there is T seconds to the time when each peer sends ping messages:



When node 5 and 14 leave the network their two predecessor nodes in this case, nodes 1 and 3 for node 5 and nodes 9 and 10 for node 14, will find it out by sending ping messages.

The amount of time needed for these ping messages to be sent from considered nodes, could be calculated as follow:

Propagation delay for each link= $50*10^3/10^8$=0.5ms

Ping message from node 1 to node 5= 1.5*2=3ms
Ping message from node 1 to node 3= 1*2=2ms
Ping message from node 3 to node 5= 1.5*2=3ms
Ping message from node 3 to node 8= 2*2=4ms
Ping message from node 9 to node 14= 2*2=4ms
Ping message from node 9 to node 10= 1.5*2=3ms
Ping message from node 10 to node 14= 1.5*2=3ms
Ping message from node 10 to node 1= 1*2=2ms
In the next step the time for updating the table should be calculated:

Node 1 should ask node 3 for its successor and node 3 should ask node 8 for successor the time required for these actions to be taken equals to:

Inquiry message from node 1 to node 3= 2*1ms=2ms

Inquiry message from node 3 to node 8= 2*2ms=4ms

Also Node 9 should ask node 10 for its successor and node 10 should ask node 1 for successor the time required for these actions to be taken equals to:

Inquiry message from node 9 to node 10= 2*1.5ms=3ms

Inquiry message from node 10 to node 1= 2*1ms=2ms

So obviously the time needed for all tables to be updated can be calculated as follow:

For node 1: 3.5ms

For node 1: 5.5ms

For node 1: 4.5ms

For node 1: 4ms

So the total time for updating the table would be equal to 5.5ms