

Portfolio-1 Guidelines

DATA 2410: Networking and Cloud Computing

Safiqul Islam

Organisational matters

- Individual submission
- 60% of your final grade
- Please note that you'll only get one Final grade after portfolio-2.
- Deadline: **Monday April 17 2023 at 12:00**
 - Hard deadline: no extension!!!!
- Submission: Inspira exam systems (access: 2 weeks before the deadline)

Prerequisites

- Socket programming
- Mininet (see all the lab tutorials)

Should you be missing lectures and lab sessions, it is your own responsibility to catch up to the competence level that you are supposed to get it from the lectures, assignments and lab sessions.

What you should have:

- Ubuntu/lubuntu
- Mininet

NOTE: You will not be able to complete the performance evaluation task without mininet.

Tasks

- Implement **simpleperf** (45%)
- **Performance evaluations** of a virtual network in mininet using simpleperf (45%)
- **Report** (10%)

Part 1: Simpleperf

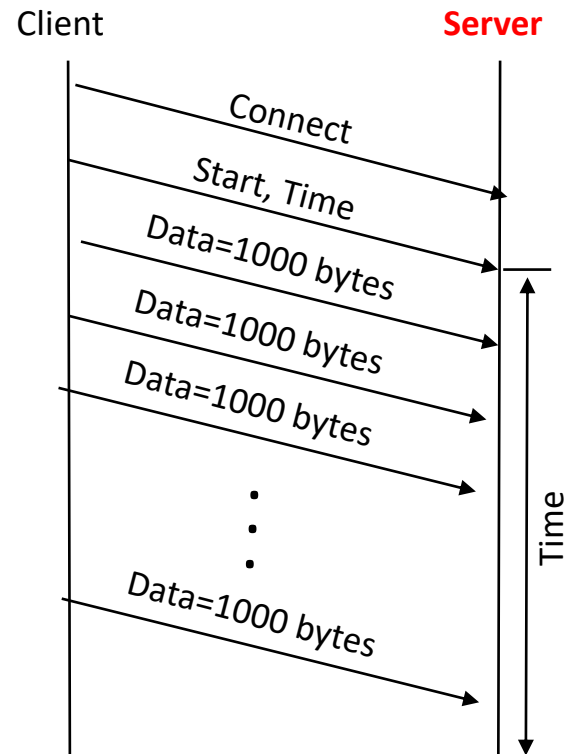
Implement simpleperf (45%)

- Design and implement simpleperf
 - A simplified version of iperf using sockets
- What you will implement:
 - **Server:** receives lots of data from client(s)
 - **Client:** sends lots of data to the server

Simpleperf **Server**

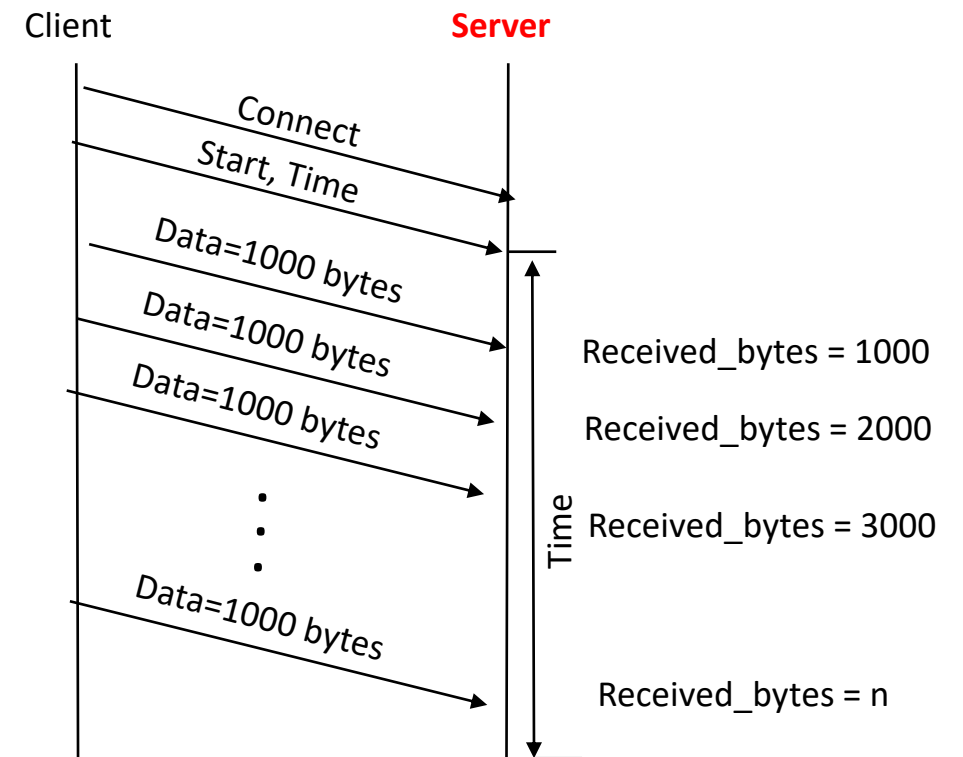
Server

- A server will accept connection and receive data in the chunks of 1000 bytes
 - A client will inform the start of the transfer!



Server

- It will also calculate how many 1000-bytes data it has received.
 - Track the number of bytes received
 - add them to the received_bytes variable!



Arguments

- You must use optional [arguments](#) using argument parser
 - Here are some examples from my lecture on 10.03.2023:
<https://github.com/safiquel/2410/tree/main/argparse-and-oop>
- The arguments for the servers are:
 - -b or --bind : allows to select the ip address of the server's interface
 - Default: 127.0.0.1
 - So you can run in your local machine
 - Must be in the dotted decimal notation (see check_ip.py [example](#) on github)
 - -p or --port : port number on which the server should listen
 - Default: 8088
 - Must be a positive integer
 - in the range [1024, 65535] (see [example](#) on github)
 - -f or --format: the format of the output data from [B, KB, MB]
 - Default : MB

Run the server

- Run with
 - Python3 simpleperf -s
- Enables server mode
 - With default IP, port and format
- The server should be ready now to receive tons of data
- Prints the following output

A simpleperf server is listening on port XXXX

Simpleperf Client

Arguments

- The arguments are:
 - -c or --client: enables the client mode
 - -l or --serverip : selects the ip address of the server
 - Default: 127.0.0.1
 - So, you can run in your local machine
 - -p or --port : allows to select the server's port number on
 - Default: 8088
 - Must be a positive integer
 - in the range [1024, 65535]
 - -f or --format: the format of the output data from [B, KB, MB]
 - Default : MB

Arguments

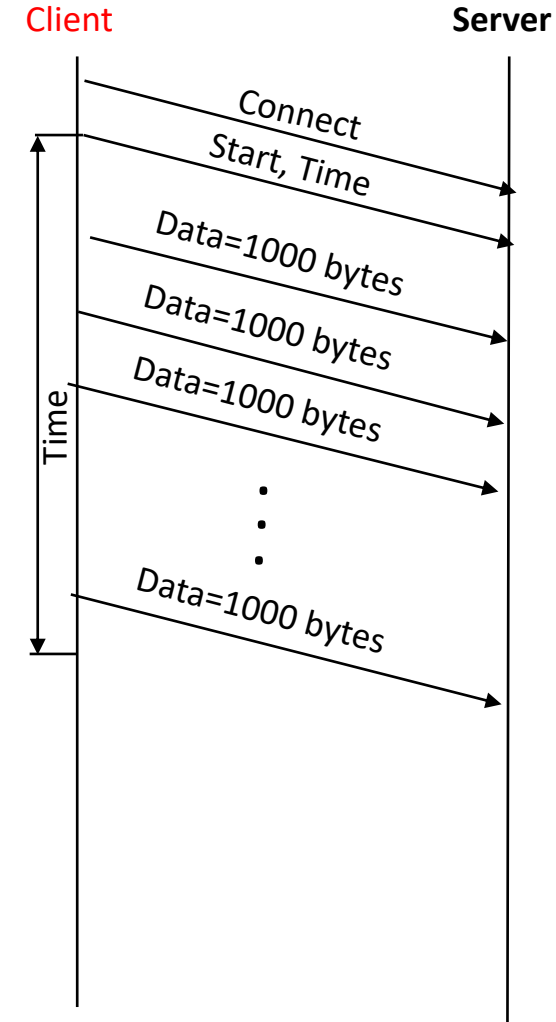
- -t or --time: the total duration in seconds for which data should be generated
 - Must be > 0
 - Default 25

NOTE:

- (-p or --port) and (-f or --format) arguments are common for both server and client!
- A server or client cannot run -s and -c at the same time:
 - Python3 simpleperf **-c -s**
 - Should report that you must run **either in client or server mode!**
 - HINT: play with args.server and args.client variables!
- Creating a packet of size=1000 bytes:
 - Data = '0' * 1000
 - Or, data = bytes(1000)
 - Many other ways! You decide!
- Time.time() gives you current time in seconds
 - you just need to make sure your run for currenttime + your_times and transfer tons of data
 - Module to import: **import time**

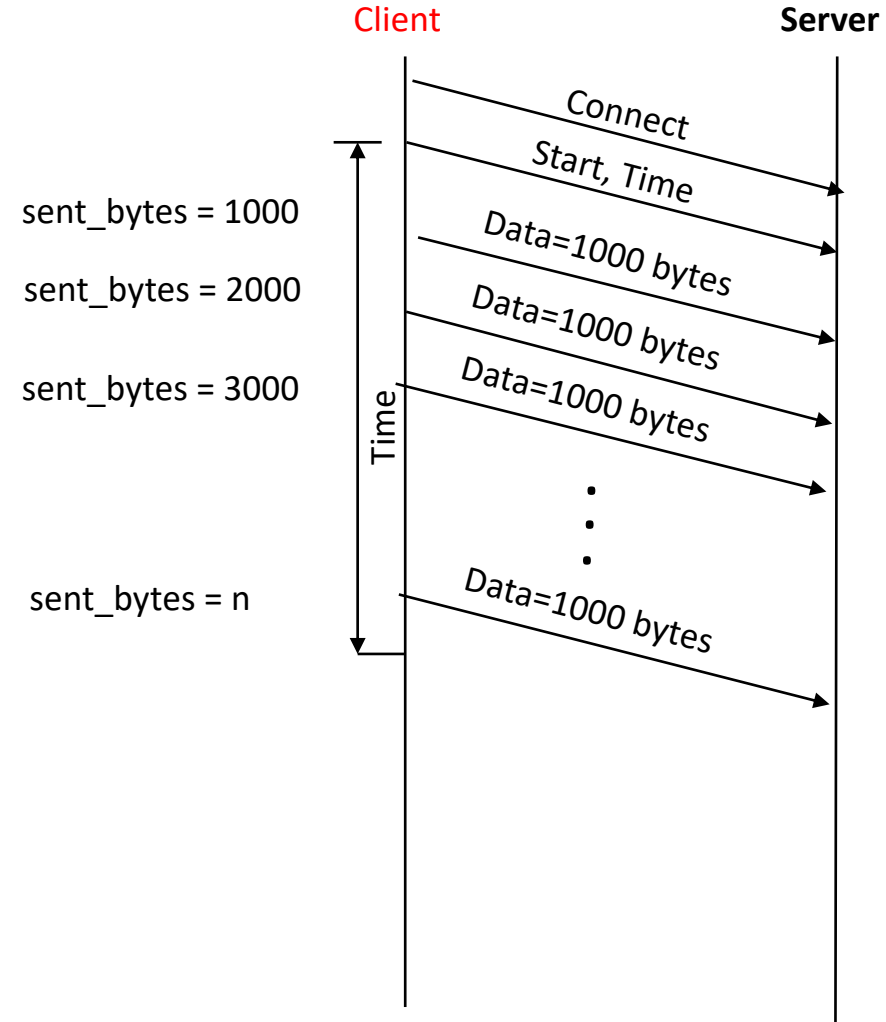
Client

- Client connects
 - Sends an indication to the server that a start of the message and also the duration (if specified by – t flag)
- A client will send data in the chunks of 1000 bytes for the time specified.



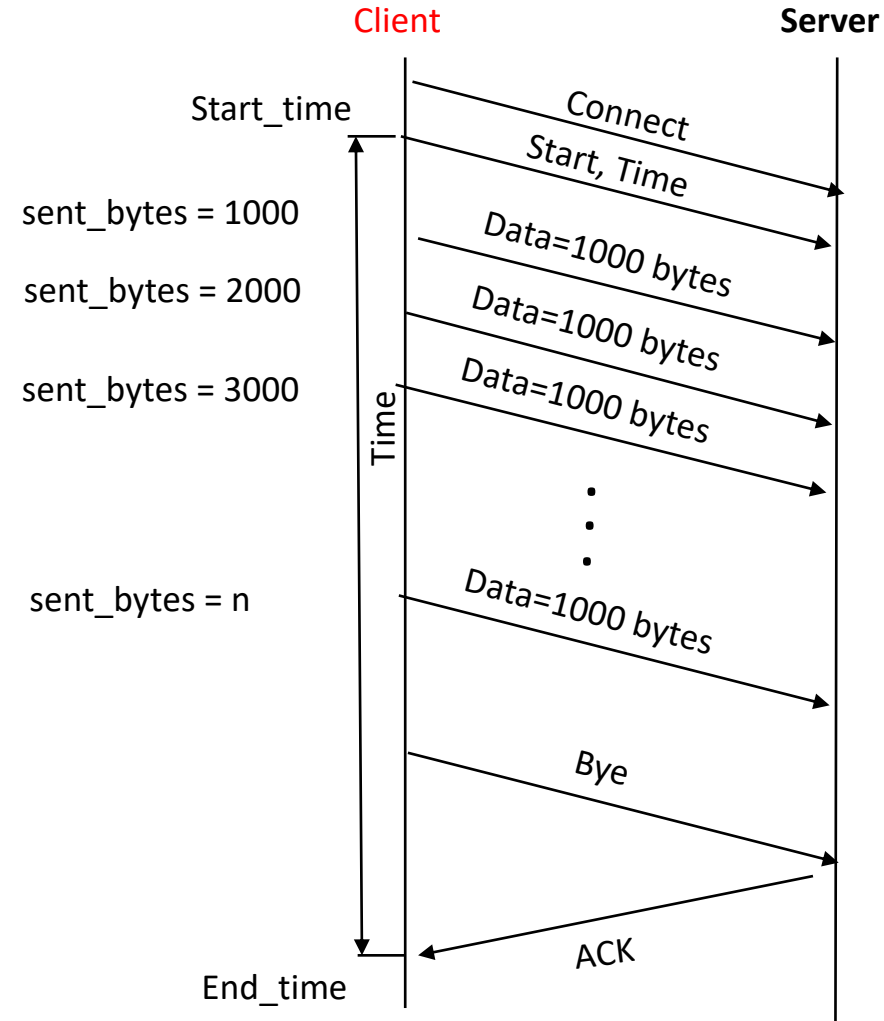
Client

- Calculates the number of bytes sent



Client

- Client sends a BYE message
 - You can send it whenever you want, but it should not be in the data transfer part.
- Server sends an ACK
- Client closes the connection
- Calculate the time difference ($\text{end_time} - \text{start_time}$)
 - Duration for bandwidth/rate calculation
 - Use `time.time()`
- Both server and client print the results!



Run the client

- Run with
 - Python3 simpleperf -c -l <server_ip> -p <server_port> -t 25
 - Replace server_ip and server_port with the correct IP and PORT
- Enables client mode
 - Connect with server (IP:PORT)
- The client should now send tons of data for 25 seconds
- Prints the following output

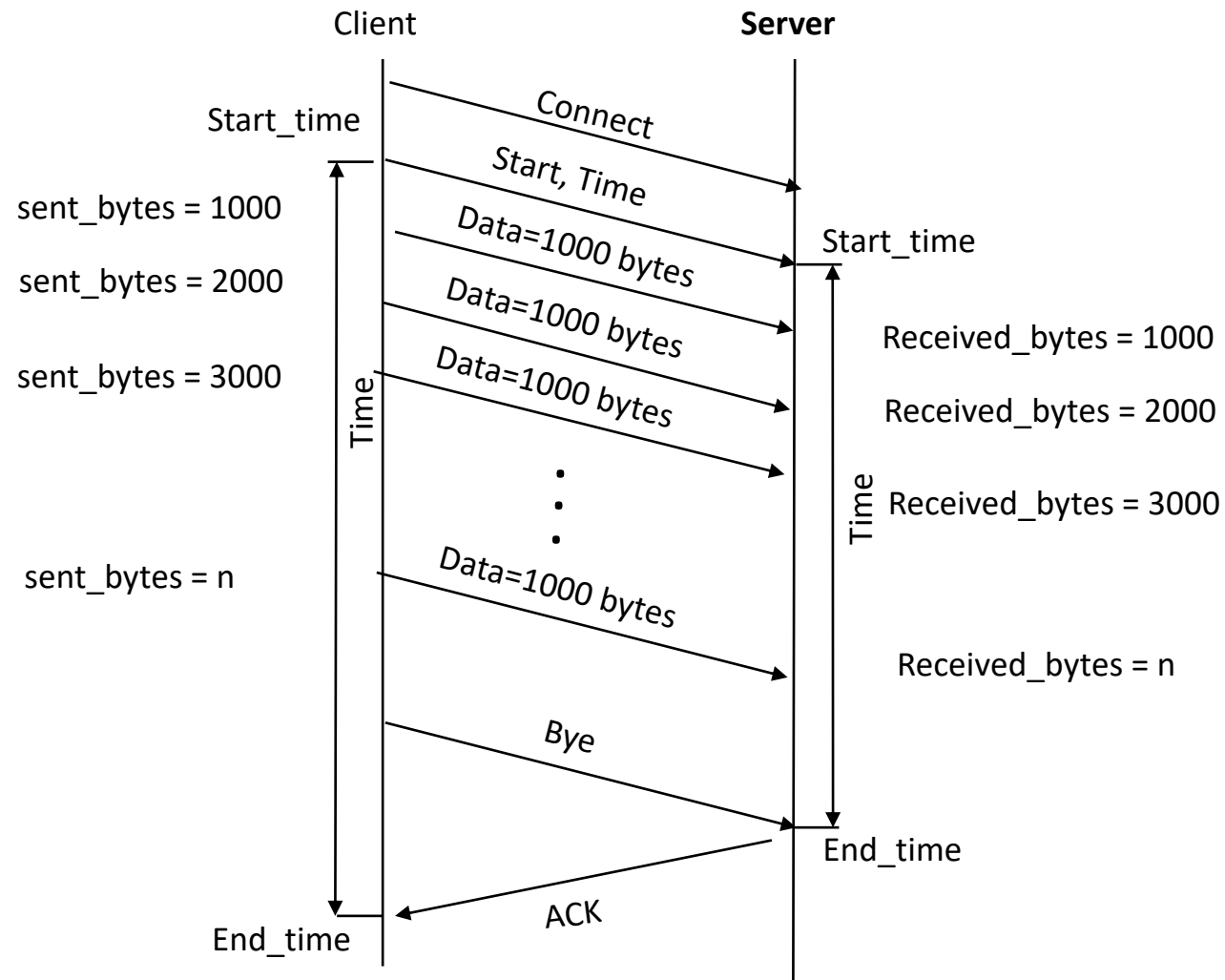
A simpleperf client connecting to server <IP>, port XXXX

Client connected with server_IP port XXXX

ID	Interval	Transfer	Bandwidth
IP:port	0.0 - 25.0	X MB	Y Mbps

Simpleperf Client and Server

Client and server!



Outputs

Server

When you start the server

A simpleperf server is listening on port XXXX

When a client joins

A simpleperf client with <IP address: port> is connected with <server IP:port>

ID	Interval	Received	Rate
IP:port	0.0 - 25.0	X MB	Y Mbps

End of the transfer after the
bye and ACK messages

Client

When you run the client

A simpleperf client connecting to server <IP>, port XXXX

When a client is connected

Client connected with server_IP port XXXX

ID	Interval	Transfer	Bandwidth
IP:port	0.0 - 25.0	X MB	Y Mbps

End of the transfer after the
bye and ACK messages

- $\text{Total_duration} = \text{End_time} - \text{start_time}$ (see the previous slide)
- $\text{Rate (server)} = \text{amount of bytes received} / \text{total_duration}$
- $\text{Rate (client)} = \text{amount of bytes sent} / \text{total_duration}$

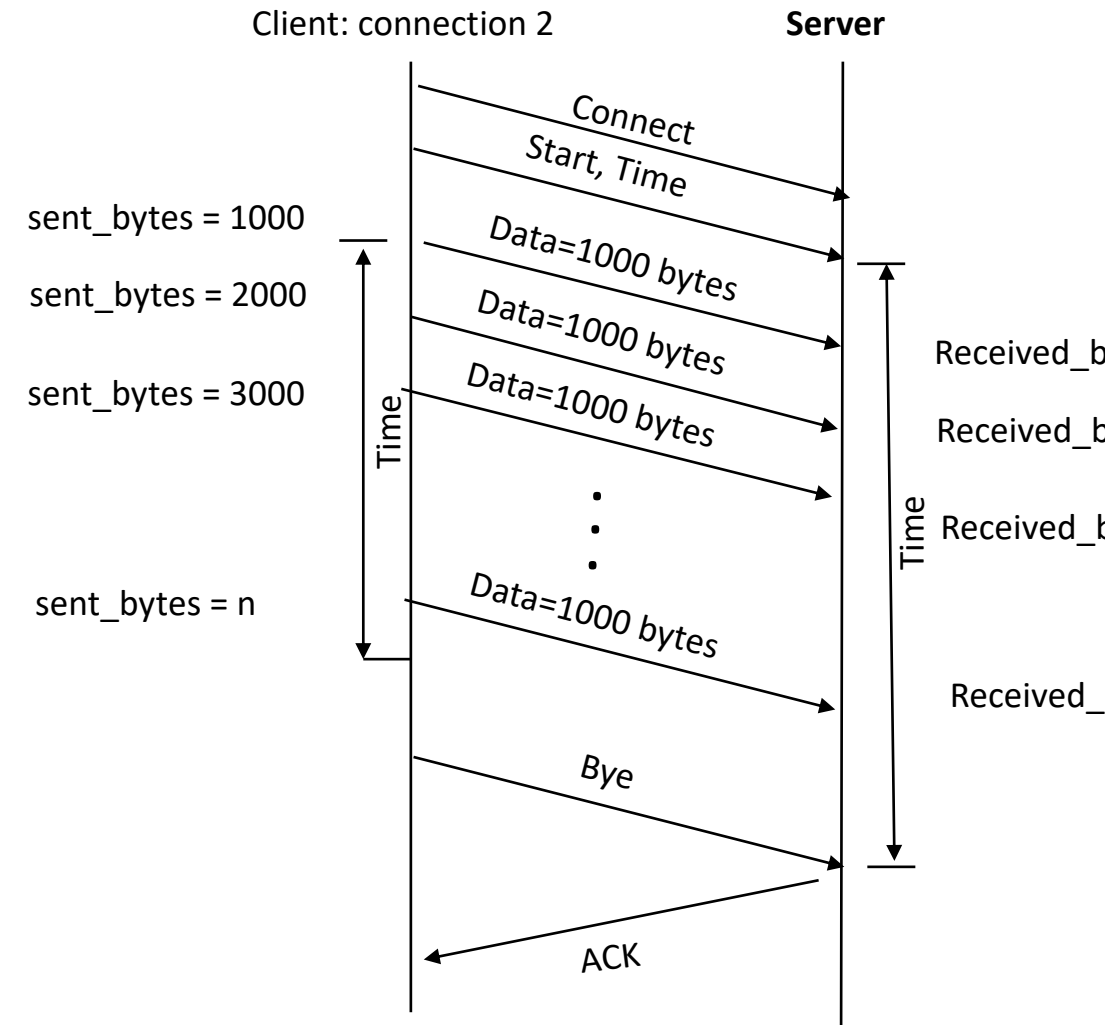
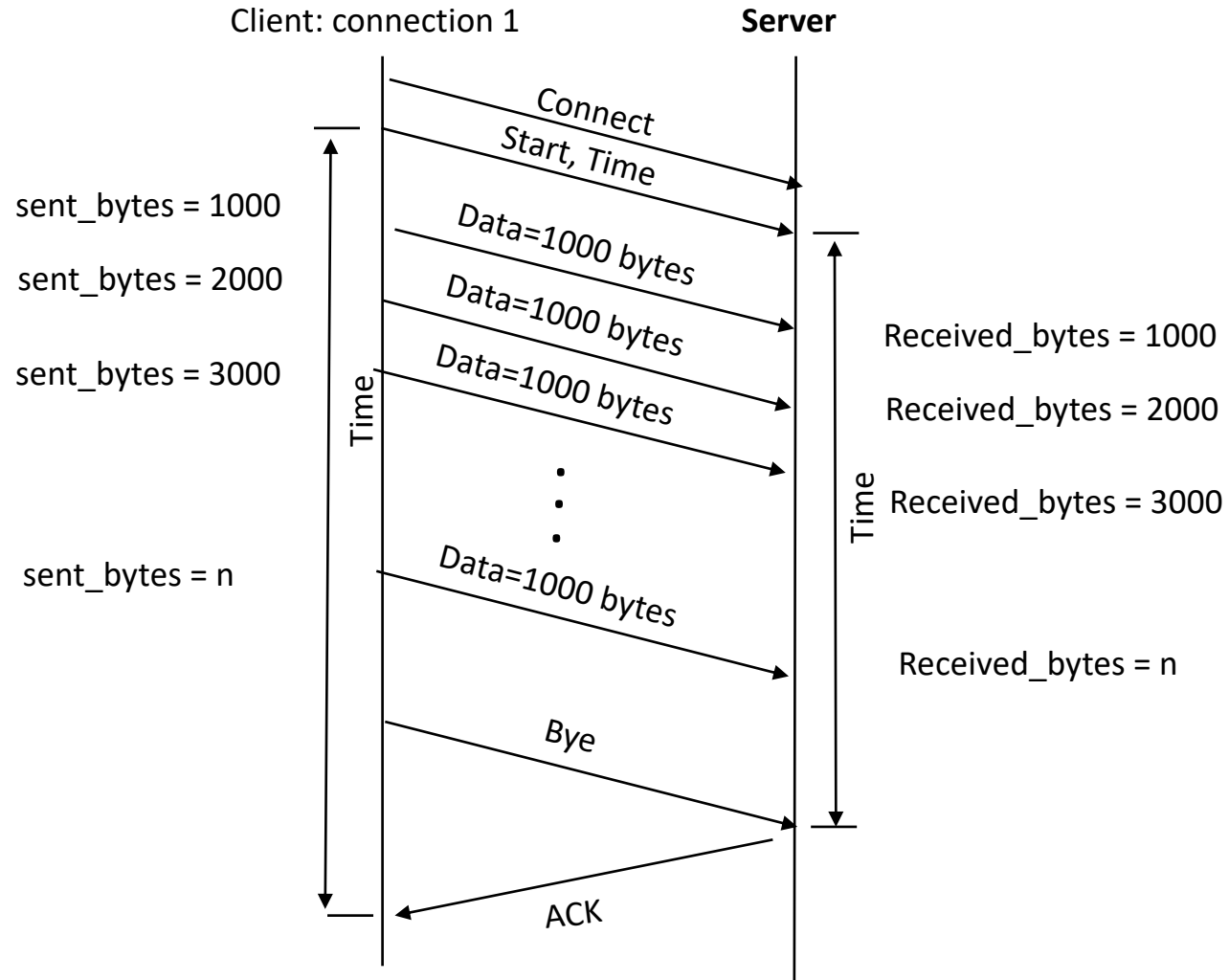
More arguments (client)

- -n or --num: transfer number of bytes specified by -n flag,
 - it should be either in B, KB or MB
- -i or --interval: print statistics per z second
 - Must be an integer sand > 0
- -P or --parallel: create parallel connections to connect to the server and send data
- If you do not implement these arguments
 - You only lose points for these.

Parallel connections!

- If you run simpleperf client with:
 - `Python3 simpleperf -c -l <server_ip> -p <server_port> -P 2 -t 25`
- It will open two separate client sockets, connect with the server, and send tons of 1000 bytes data from two connections at the client side
 - What you should do:
 - Create two sockets at the client side, and connect to server with two different sockets – send tons of packets from both connections
 - Hint:
 - Use separate a separate thread for each clientsocket
 - The outputs will vary based on your selections!

Parallel connections!



Output (client side)

```
-----  
-----  
A simpleperf client connecting to server <IP>, port XXXX  
-----  
-----  
  
Client IP:port connected with server_IP port XXXX  
Client IP: port connected with server_IP port XXXX  
  
ID          Interval    Transfer  Bandwidth  
  
IP:port     0.0 - 25.0    X MB      Y Mbps  
IP:port     0.0 - 25.0    X MB      Y Mbps
```

Server side out: should be similar (see the previous server outputs and this client-side output).

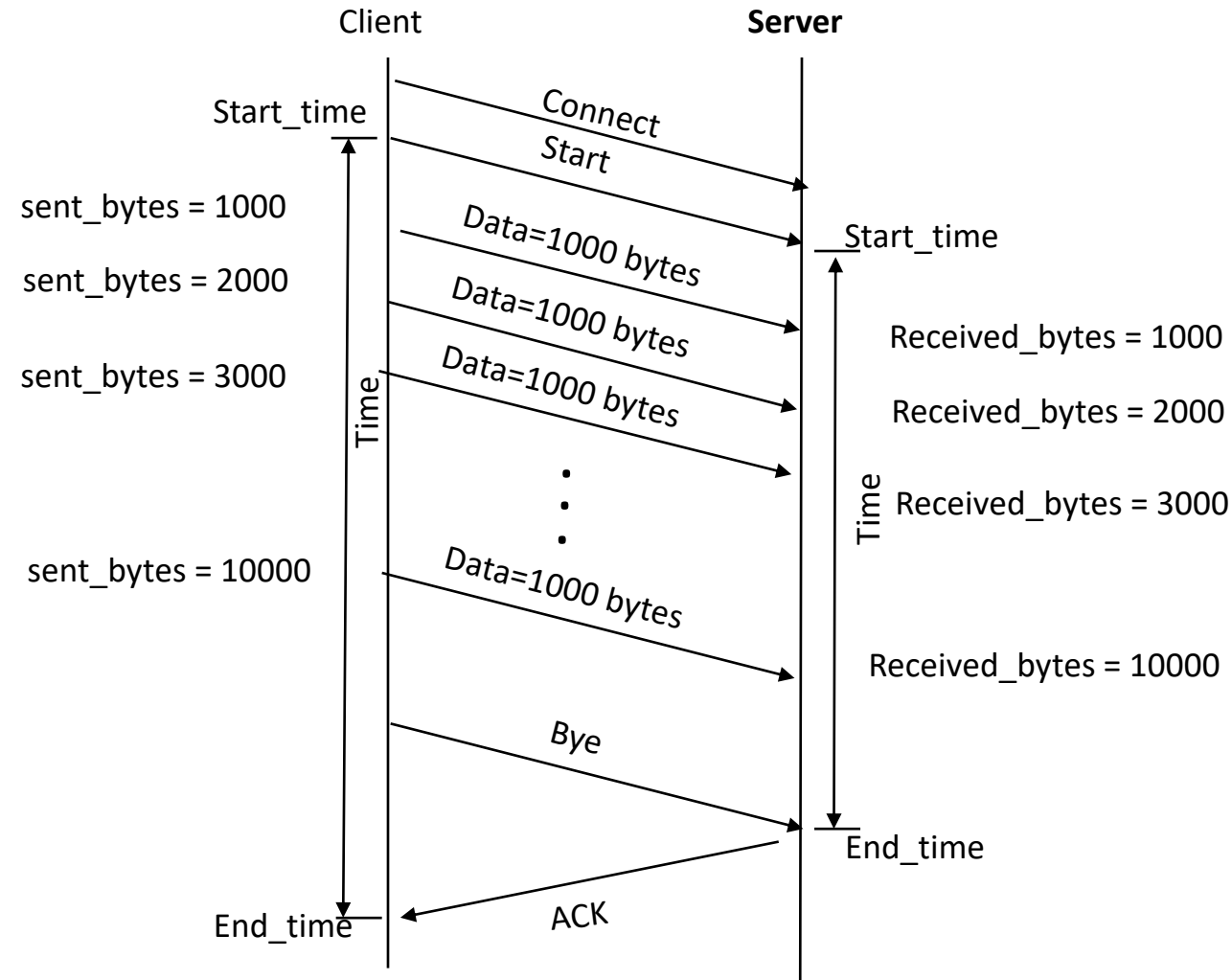
Num flag

- NOTE: do not run `-n` and `-t` at the same time
 - For your information, I am not going to test both at the same
 - You can handle this or skip it - your call!
- Run:
 - `python3 simpleperf -c -l <server_ip> -p <server_port> -n 10KB`

This will transfer 10 MB data from the client to the server!

Num (-n) flag

- NOTE: you still need to calculate start_time before you send your first packet (before the while loop) `start_time = time.time()`
- When the transfer is finished
 - `end_time = time.time()`
- The difference will give you the **duration**
- Run:
 - `python3 simpleperf -c -l <server_ip> -p <server_port> -n 10KB`
- This will transfer 10 MB data from the client to the server side and stops!
- $\text{Rate} = \text{bytes_sent} / \text{duration}$




HINT

- If you use `-n 10MB`, you can:
 - Split the number (10) and UNIT (MB)
 - Convert MB to B
 - Send all the bytes
 - If it takes less than a second to transfer a file (`end_time - start_time`)
 - Set the duration to 1 second (for simplicity)
- Use regular expression:

```
import re
match = re.match(r"([0-9]+)([a-z]+)", val, re.I)
if match:
    items = match.groups()
    print (items)  #output: (10, 'MB')
```

interval flag

- If you run:
 - `python3 simpleperf -c -l <server_ip> -p <server_port> -t <time> -i 5`
- A client should print results every 5 seconds! 
- A server should just print “total” as before
 - It only affects the client side

A simpleperf client connecting to server <IP>, port XXXX

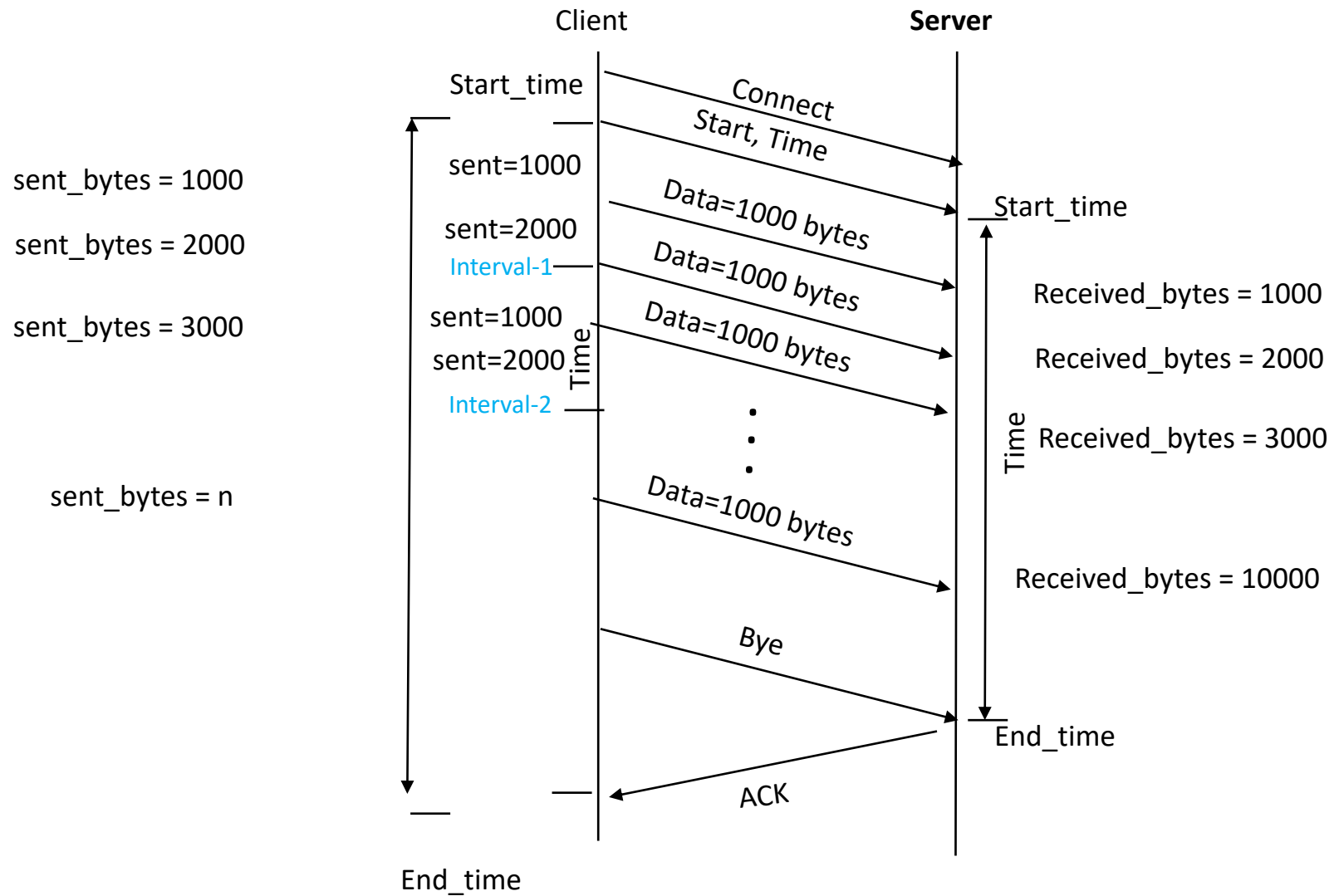
Client connected with server_IP port XXXX

ID	Interval	Transfer	Bandwidth
IP:port	0.0 - 5.0	X MB	Y Mbps
IP:port	6.0 - 10.0	X MB	Y Mbps
IP:port	11.0 - 15.0	X MB	Y Mbps
IP:port	16.0 - 20.0	X MB	Y Mbps
IP:port	21.0 - 25.0	X MB	Y Mbps

IP:port 0.0 - 25.0 X MB Y Mbps

HINT:

1. Calculate the number of bytes from the start time to first interval
 - Calculate rate and total_bytes sent
 - Print
2. Repeat step 1 from the first interval to the second interval!
3. Repeat for the other intervals

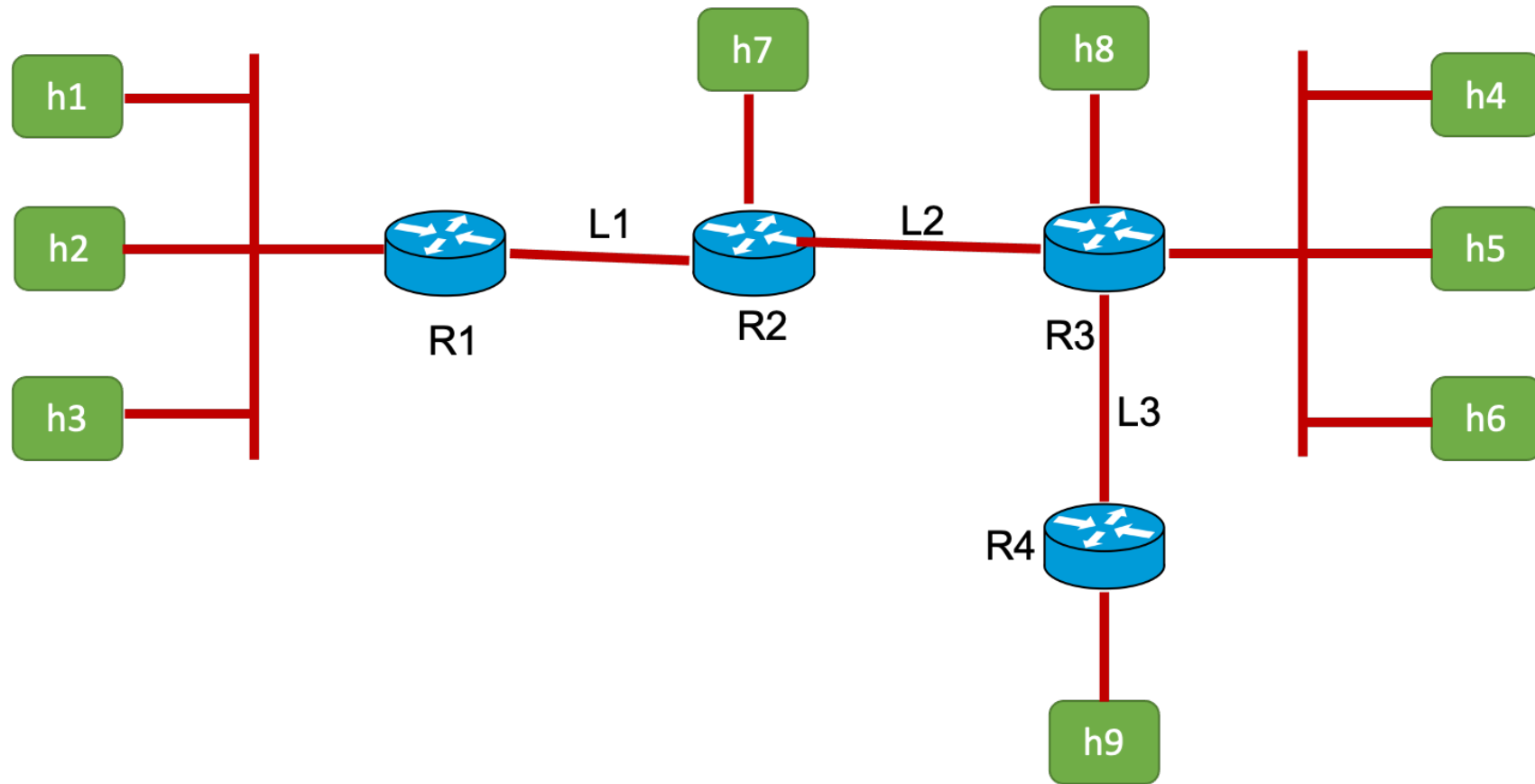


Performance Evaluation

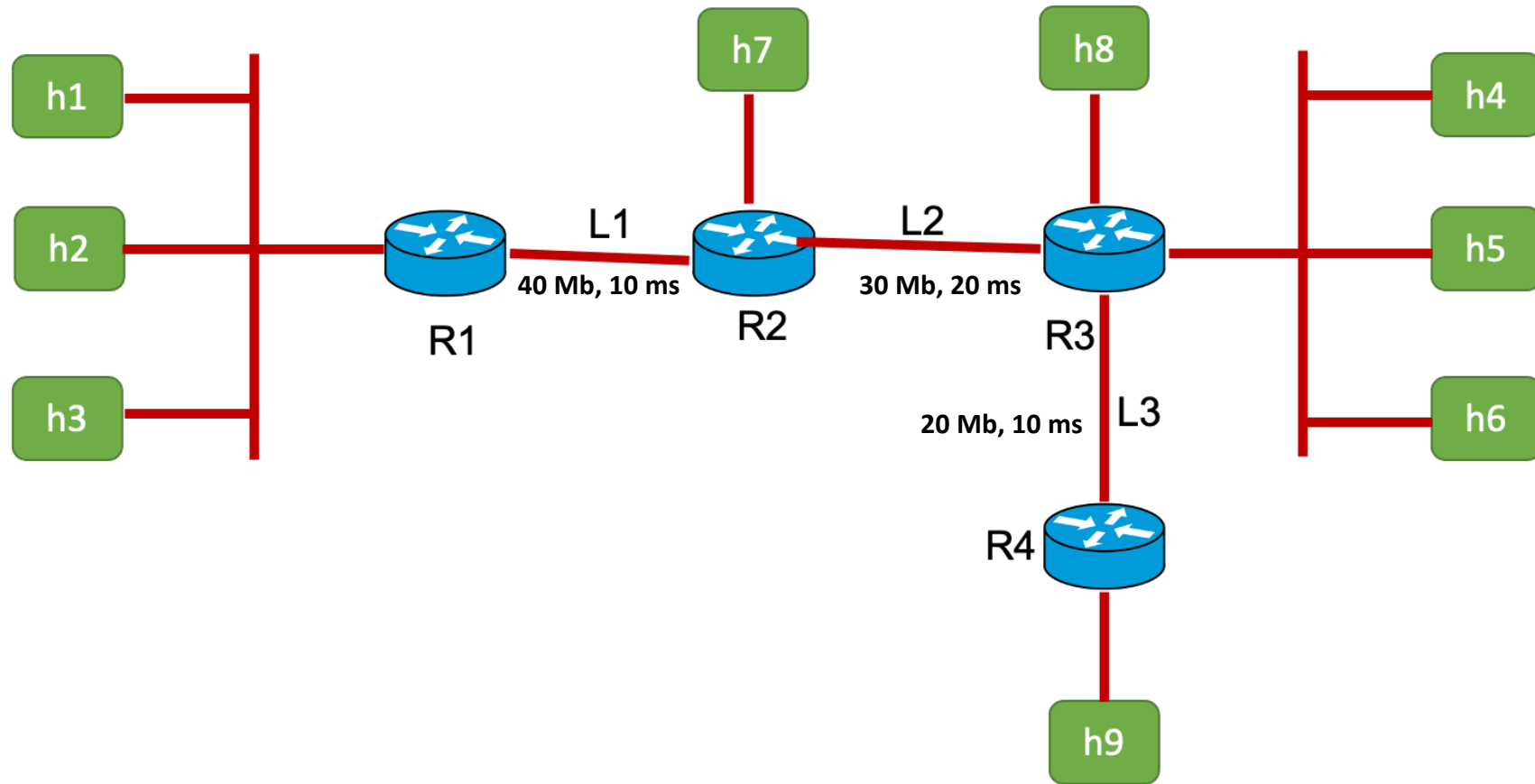
Performance Evaluation

- Evaluate the performance of a network using simpleperf, ping and iperf (only test case 1)
- What you should have:
 - ubuntu/lubuntu + mininet installed!
 - Finished lab tutorials and assignments
 - if you have missed the lectures/assignments/labs:
 - catch up on your own (see my tutorials, lectures, etc.)
 - come to the lab and ask Task
- See my performance evaluation lecture for:
 - performance metrics (bandwidth/throughput, delay/latency, loss, queuing delay (buffering at the router))

Topology



Topology



Case 1

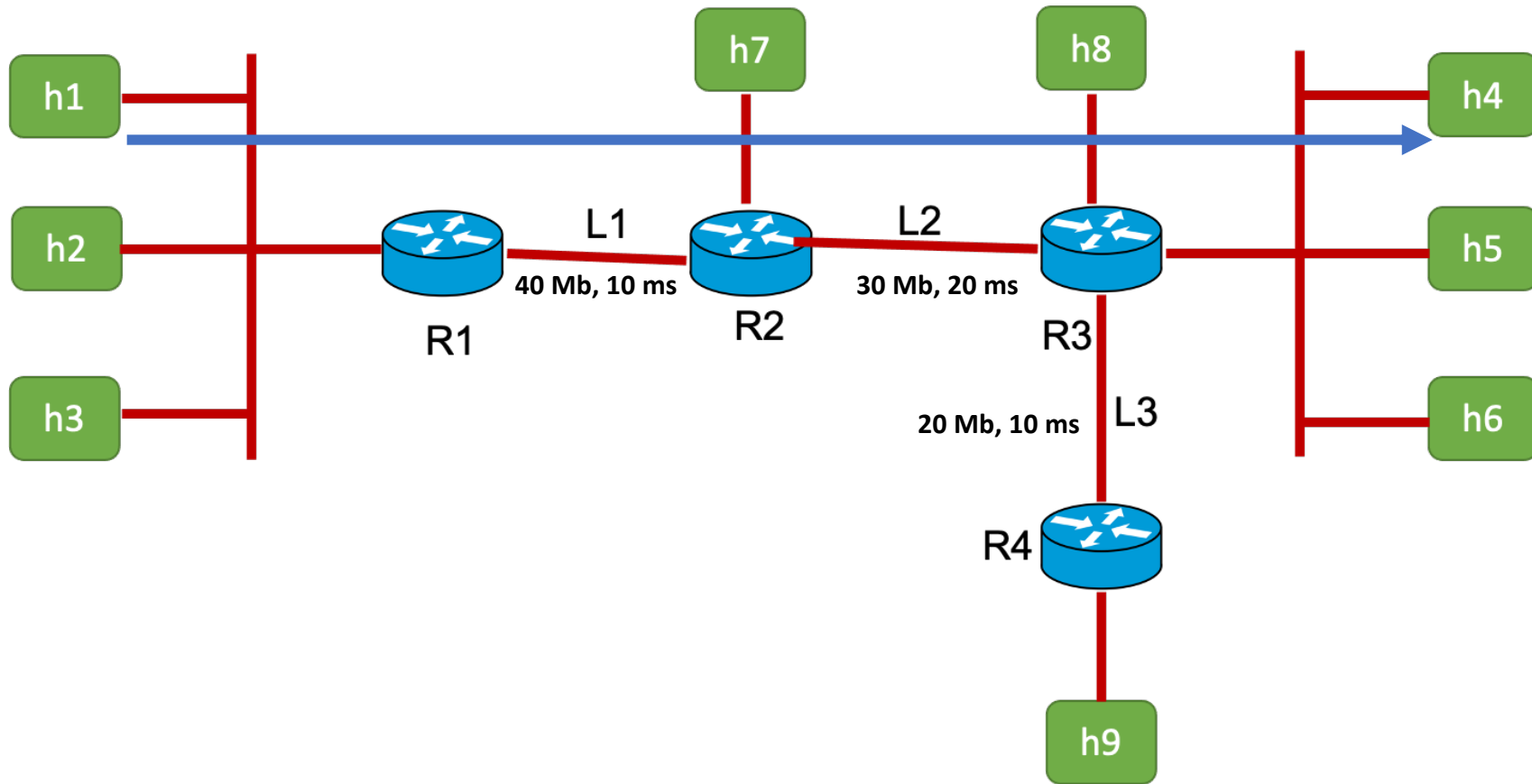
Measuring bandwidth with iperf in UDP mode

- Three separate iperf tests with UDP mode with -b 'X'M between the following client-server pairs:
 - 1.between h1 - h4
 - 2.between h1 - h9
 - 3.between h7 - h9
- Store the output of these 3 separate measurements in throughput_udp_iperf_h1-h4.txt, throughput_udp_iperf_h1-h9.txt, and throughput_udp_iperf_h7-h9.txt files.
 - **Only the client sides**

Test case 1

- See lab manuals: week 9 and week 10, and lab assignment 4
- E.g.,
 - Server: `iperf -s -u`
 - Client: `iperf -c 10.0.1.2 -u -b 90M`
- - u specifies UDP mode
- -b specifies the rate at which you want to send (90 Mbps here for example)

Topology



Measure the bandwidth between h1 and h4. See the mininet python script for the IP addresses or open terminals for h1 and h4 to find their ip addresses with ifconfig command.

Questions?

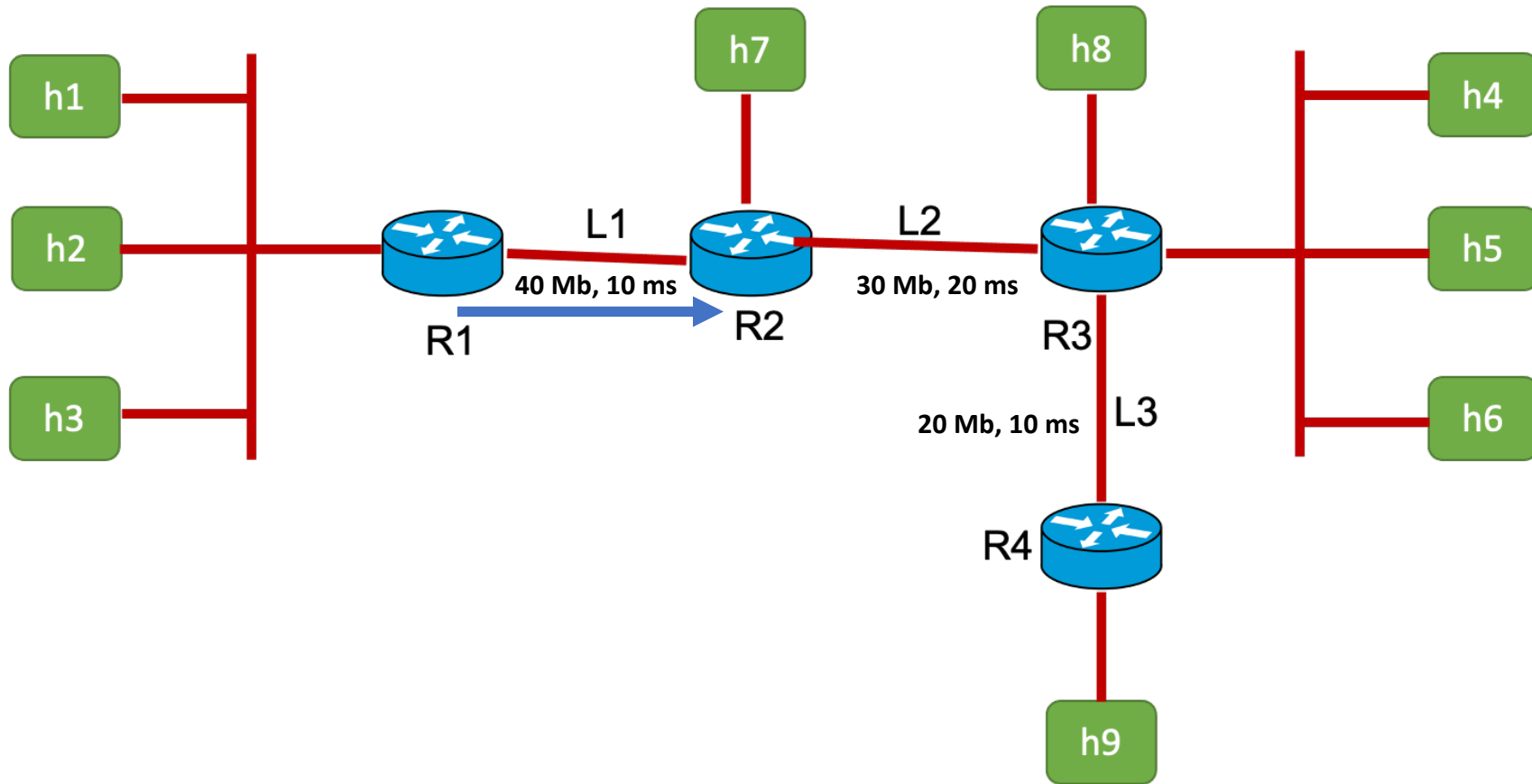
- Can you answer:
 - Which rate after `-b` flag would you choose to measure the bandwidth for h1-h4? Explain your answers.
 - Compare with what you expect and what you get. Do they match? Why or why not?
 - If you are asked to use iPerf in UDP mode to measure the bandwidth where you do not know anything about the network topology, what approach would you take to estimate the bandwidth? Do you think that this approach would be practical? Explain your answers.
 - For a small network (example script), we know the devices
 - But, what about in real life?
- Answer the questions between the other host pairs!

Case 2

Link Latency and Throughput

- RTT and bandwidth of each of the three individual links between routers (L1 - L3)
 - RTT with ping (see mininet labs and lab-4 assignment)
 - Bandwidth with simpleperf (see lab manual weeks 10 and 11)
- Let's consider L1 (r1-r2).
- For throughput test: run simpleperf for 25 seconds
 - For this I don't care about `-num`, `-interval` flags!
 - Only a simple test with the default flags
 - Store the output of the measurement on each link in a file called `Throughput_L1.txt`,
- Run ping with 25 packets
 - store the output of the measurement on each link in a file called `latency_L1.txt`
- Repeat for L2 and L3

Topology



See the mininet python script for the IP addresses or open terminals for r1 and r2 to find their ip addresses with ifconfig command

Test case 2

- Ping r2 from r1
 - Ping 10.0.1.2 -c 25
 - See (line 76-77) to find the IP addresses between r1 and r2
 - <https://github.com/safiquel/portfolio-1/blob/main/portfolio-topology.py>
 - Open terminals and find the ip addresses of the correct interfaces with ifconfig command
 - If you want to know how they are connected, use the net command on mininet> main window
 - See week 9, intro to mininet lab tutorial
 - To store the result in a file:
 - Ping 10.0.1.2 -c 25 > Latency_L1.txt
- Simpleperf:
 - Server: run server on r2
 - Python3 simpleperf -s -b 10.0.1.2 -p 8088
 - Client (r1):
 - Python simpleperf -c -l 10.0.1.2 -p 8088 -t 25

Answers

- Explain your results:
 - Compare with expected vs reality
 - i.e., by looking at the topology what you should get vs your performance evaluations

Case 3

Test case 3: Path Latency and Throughput

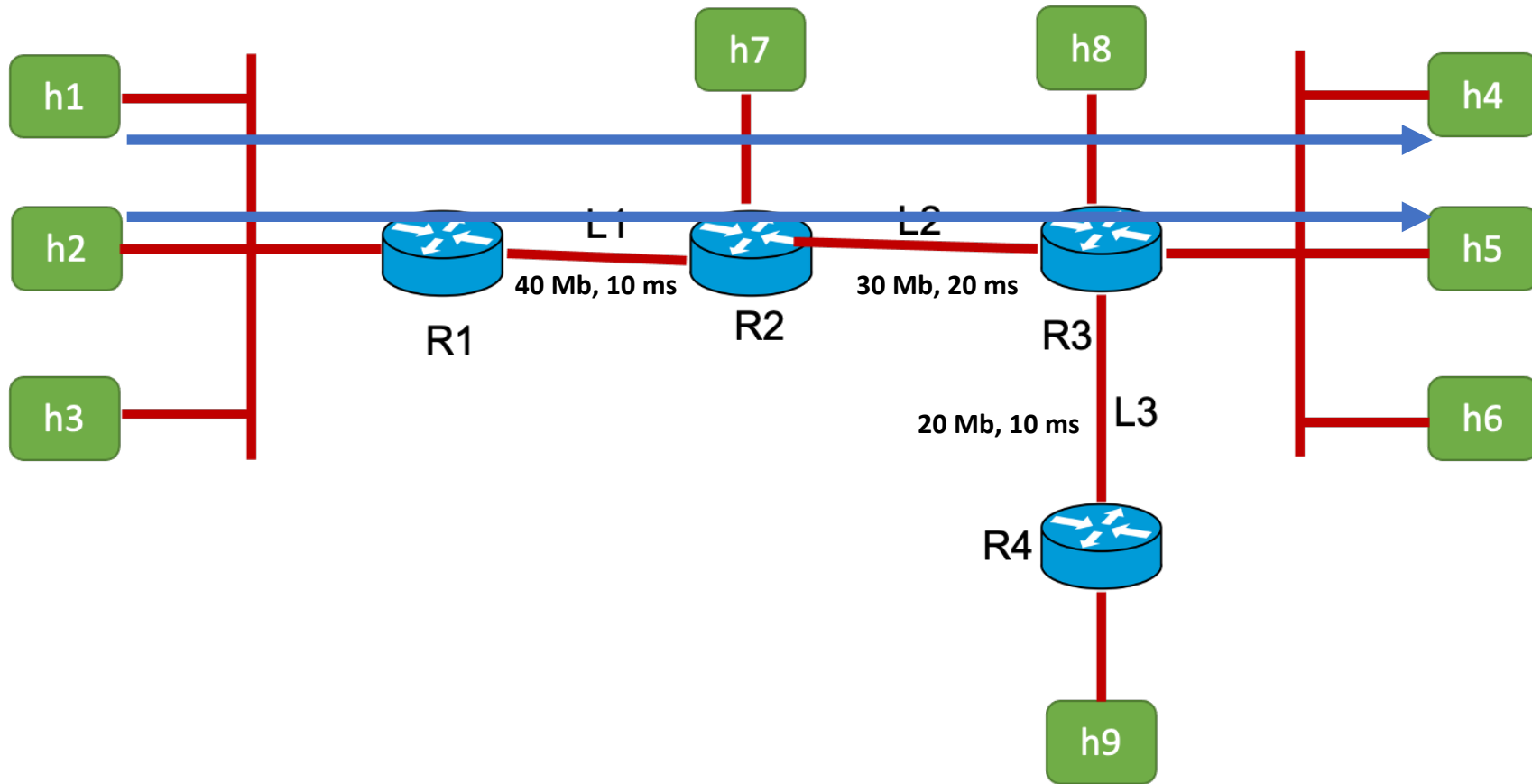
- Same as case 2: you will run between hosts instead of routers Read the description
 - For example: h1 wants to communicate with h4
 - Ping h4 from h1 (same as before)
 - Run simpleperf server on h4 and client on h1
 - Find h4's IP from the script (line 97) or the terminal with ifconfig command

Case 4

Effects of Multiplexing and latency

- In this experiment, you will look at the effects of multiplexing where many hosts will simultaneously communicate.
- E.g., Two pairs of hosts (h1-h4 and h2-h5) will simultaneously communicate using simpleperf.

Topology



Effects of Multiplexing and latency

- Two pairs of hosts (h1-h4 and h2-h5) will simultaneously communicate using simpleperf.
- Open terminals for h1 (twice), h2 (twice)
 - Xterm h1 h1 h2 h2
- Open terminals for h4 and h5
 - Xterm h4 h5
- Run simpleperf server on h4 and h5 (see the script to get the correct IP addresses)

Effects of Multiplexing and latency

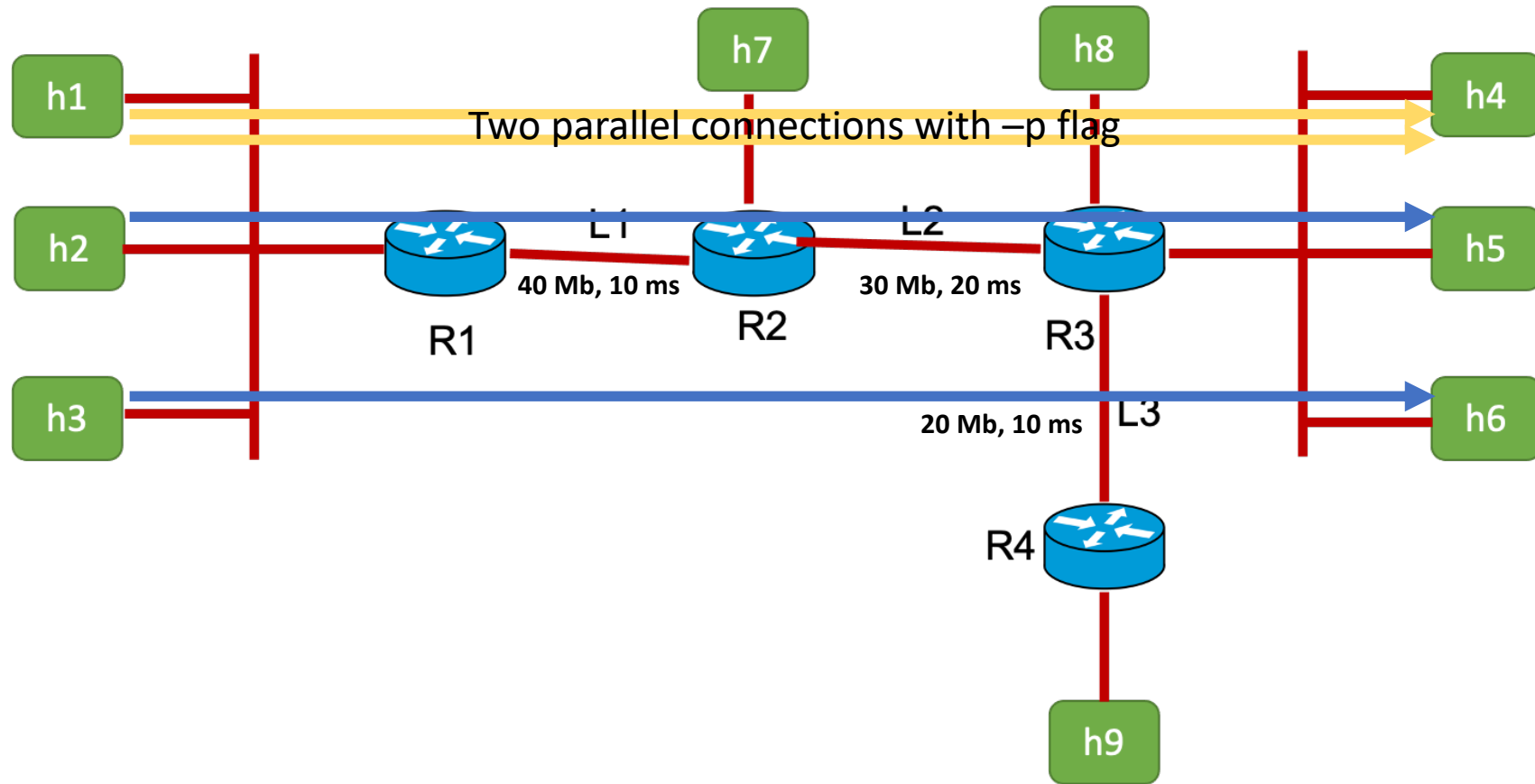
- Write `ping <h4's ip> -c 25` in the h1's first terminal
- Type in the `simpleperf client` command on h1 and h2 (without hitting ENTER)
- Type in the `ping` command on h1 and h2 (without hitting ENTER)
- Quickly hit ENTER to run ping commands on all client hosts so that they start at roughly the same time.
- Quickly hit ENTER to run `simpleperf` commands on all client hosts so that they start at roughly the same time.
- Store the output in the files (see the portfoslio description for the file names)
- Explain your answer (expected vs reality)
 - Note: do not put raw data in your document!

Case 5

Effects of parallel connections

- h1, h2 and h3 connected to R1 will simultaneously talk to hosts h4, h5 and h6 connected to R3
- In this case, you will only run throughput tests with simpleperf

Topology



Effects of parallel connections

- h1, h2 and h3 connected to R1 will simultaneously talk to hosts (h4, h5 and h6) connected to R3.
- In this case, you will only run throughput tests with simpleperf
- Open terminals for h1 h2 h3 h4 h5 h6
- Run servers on h4 h5 and h6
- Run simpleperf client with either with `–parallel 2` or `-P 2` on h1
- Run simpleperf client without the `-P` flag on h2 and h3
- You must start them at the same time
 - type in the simpleperf client command on each of the client hosts without hitting ENTER, and then quickly hit ENTER on all client hosts so that they start at roughly the same time.
- See the results, and explain in your answer
 - Expected vs reality

More about report: next week!

Thank you!

Questions!