ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communications
CS–202 Computer Systems
Argyraki K., Kashyap S. & Chappelier J.-C.

NOM : Hanon Ymous
(000000) #0000
Seat #: 0

CS–202 COMPUTER SYSTEMS

Final Exam

June 16th, 2025

INSTRUCTIONS (please read carefully)

IMPORTANT! Please strictly follow these instructions, otherwise your exam may be canceled.

1. You have three hours to complete this examination (3:15–6:15pm).
2. You must **use black or dark blue ink**, neither pencil nor any other color.
3. This is closed book exam.
   Personal notes, four times dual-sided A4 sheets (8 sides in total), allowed.

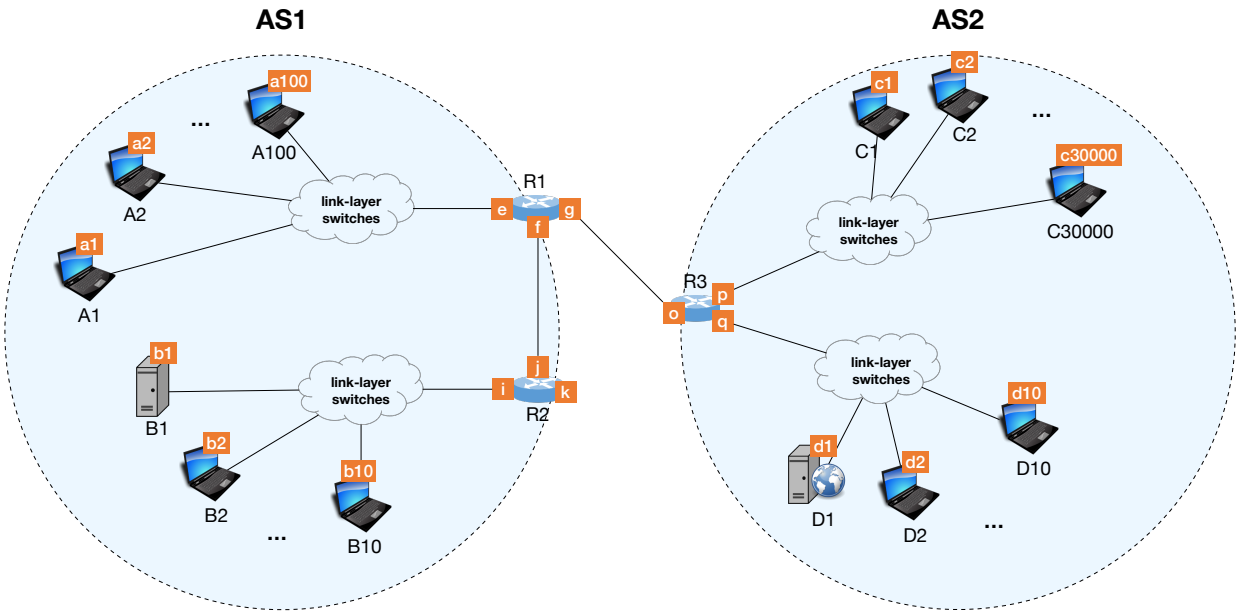
   On the other hand, you may not use any personal computer, mobile phone or any other electronic equipment.
4. Answer the questions directly on the exam sheet; do not attach any additional sheets; only this document will be graded.

   An additional blank page is provided at the end of the handout. If you use it, please clearly state the question number you are answering there.
5. Carefully and *completely* read each question so as to do only what we actually ask for. If the statement seems unclear, or if you are in any doubt, ask one of the assistants for clarification.
6. The exam consists of six independent exercises, which can be addressed in any order, but which do not score the same (points are indicated, the total is 180 points); all exercises count for the final grade.

LEAVE THIS EMPTY						
Networking		**OS**		**C**	**Project**	**TOTAL**
Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	
50	50	25	30	10	15	180

Question 1 – Subnets and packets [50 points]

Consider the following topology, which shows two Autonomous Systems (ASes), AS1 and AS2:



(For convenience, a copy is also provided on draft paper.)

Figure 1.1: Network topology for Question 1.

- There are 3 routers, R_1, R_2, and R_3. All of them are border routers.
- There are several link-layer (Ethernet) switches (not explicitly shown).
  These switches do not have IP addresses.
- There are 100 end-systems A_1 to A_{100}; 10 end-systems B_1 to B_{10}; 30 000 end-systems C_1 to $C_{30\,000}$; and 10 end-systems D_1 to D_{10}.
- B_1 is a DNS server.
- All the end-systems in AS1 use B_1 as their local DNS server.
- D_1 is a web server with DNS name `d1.epfl.ch`.
- The small orange boxes are network interfaces; e.g., router R_1 has network interfaces e, f, and g.

Do NOT write anything here!

1.1 – IP subnets [20 points]

① Identify all the IP subnets inside AS1 and AS2 by **marking them on Figure 1.1 on the left** (not on the provided extra colored sheet).

② Of the following four scenarios, only one is correct and satisfies the needs of both ASes. Identify this one scenario and explain why you rejected the other three (in 1–2 sentences per scenario).

a) AS1 owns IP prefix 5.0.0.0/24.
AS2 owns IP prefix 5.0.1.0/24.

b) AS1 owns IP prefix 5.0.0.0/24.
AS2 owns IP prefix 5.0.0.0/16.

c) AS1 owns IP prefix 5.0.0.0/16.
AS2 owns IP prefix 5.0.1.0/16.

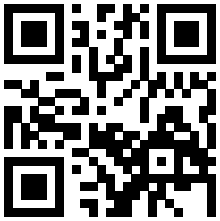
d) AS1 owns IP prefix 5.0.0.0/24.
AS2 owns IP prefix 5.1.0.0/16.

③ Assuming the correct scenario that you identified, assign an IP prefix to each IP subnet that contains end-systems. Each IP prefix you assign must have the smallest possible size. **Justify your answer.**

Answers and justifications:

more space to answer on the back ☞

Answers to subquestion 1.1 continued:



1.2 – Forwarding [4 points]

Assume that all routing protocols have converged. Draw the forwarding table of router R_1. Ignore any IP subnets that don't contain end-systems.

Answer:

continues on back ☞

1.3 – Messages [6 points]

All end-systems and packet switches have been rebooted. All caches (of all kinds) are empty. The user of end-system A_1 types in their web browser `http://d1.epfl.ch/hello` (this is the end-user's "first action"). This web page does not reference any embedded objects.

A short time later, the same user types in their web browser `http://d1.epfl.ch/news` (this is the end-users's "second action"). This web page references exactly one embedded object, an image with URL `http://d1.epfl.ch/logo.img`.

① Does end-system A_1 exchange any ARP requests/responses and/or any DNS messages as a result of the end-user's second action? If not, explain why not. If yes, explain what these messages are useful for.

② List the sequence of application-layer messages that are generated as a result of the end-user's second action. Answer by completing the table below. The first row shows an example (which is not part of the correct answer). You may not need to fill all the rows of the table.

Answers:

①

②

#	Source process	Dest process	Purpose
Example	DNS client process in A_5	DNS server process in C_1	DNS request for ...
1			
2			
3			
4			
5			
6			

Do NOT write anything here!

1.4 – Reasoning about TCP [20 points]

Consider again the scenario from subquestion 1.3. A_1 and D_1 use the same TCP connection for all their communications. From the moment A_1 starts sending to D_1 the first packet that results from the end-user's second action, it takes a bit more than 2 but less than 3 round-trip times (RTTs) for A_1 to receive the last packet from D_1.

Assume the following:

- The headers of all layers have insignificant size.
- Any message from A_1 to D_1 has insignificant size.
- `http://d1.epfl.ch/news` is 8000 bytes.
- `http://d1.epfl.ch/logo.img` is 16 000 bytes.
- The maximum segment size (MSS) is 1000 bytes.
- A_1's receiver window is always larger than D_1's congestion window.
- There is no segment loss, corruption, or reordering.

Based on this information, what can you tell about the size of `http://d1.epfl.ch/hello`?
Justify your answer.

Hint: You don't need any sophisticated delay computation. Sketch (on scratch paper) a standard TCP diagram that covers A_1–D_1 communication from the beginning. Don't worry about SEQ or ACK numbers. Focus on the number of segments transmitted in each round.

Answer and justification:

continues on back ☞

Question 2 – TCP and delays [50 points]

Consider the following network topology:

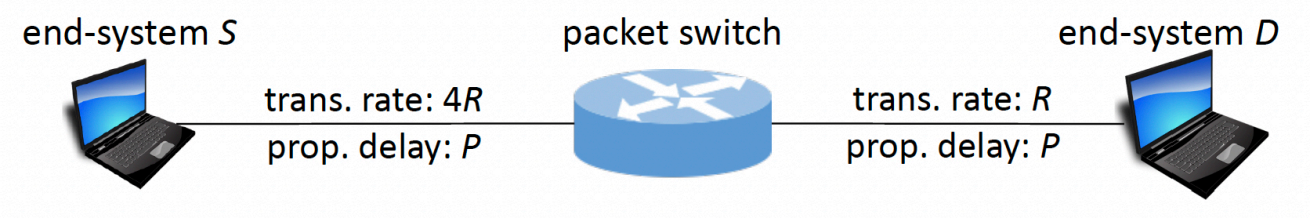


Figure 2.1: Network topology for Question 2.

- The first link has transmission rate $4R$, the second link has transmission rate R, in both directions.
- All links have propagation delay P, in both directions.
- The packet switch is store-and-forward (as we saw in class).

Assume that:

- The headers of all layers have insignificant size.
- End-systems S and D communicate over TCP Tahoe (there is no fast-retransmit or fast-recovery).
- D's receiver window is always larger than S's congestion window.
- S's TCP timeout is 3× RTTs (round-trip times).
- Processing delay at the packet switch is insignificant.

2.1 – TCP [20 points]

A process running on S establishes a TCP connection with a process running on D and sends 10×MSS bytes to it, where MSS represents the maximum segment size. D does not send any application-layer data to S.

The fourth segment sent by D to S (the SYN ACK counts as the first segment) is lost.
No other segment is lost, corrupted, or reordered.

Complete the diagram on the next page by showing:

- All the segments (including the TCP connection setup and the segments that carry only ACKs) exchanged until D receives all the data.
- The SEQ numbers of S's segments. Express them in MSS's, e.g., "MSS", "2 MSS," etc.
- The ACK numbers of D's segments. Express these in MSS's as well.
- The status of S's congestion-control algorithm.
- The values of S's congestion window and ssthresh.

Answer here: (do NOT answer on the provided DRAFT copy)

S's congestion window	S's ssthresh	state of congestion control algorithm for S	TCP diagram: Sequence number	TCP diagram: Acknowledgement number
			S	D

continues on back ☞

2.2 – Delays [30 points]

In the context of Figure 2.1, assume the following values:

- $P = 10$ msec.
- $R = 10$ Mbps.
- MSS = 1250 bytes (= 10 000 bits).

Consider the scenario described in Question 2.1 and the diagram you drew as your answer.

① From the moment S starts transmitting the first segment (the TCP SYN), it takes 300 msec until D receives the last bit of the last segment.

Based on this information, did the packet switch forward any traffic other than the traffic from S to D? **Justify** your answer.

Hint: What would the transfer time be if the switch did not forward any other traffic?

② Now assume the packet switch forwards only the traffic from S to D. Given that no segment from S to D was lost, what is the minimum size of the queue where the packet switch stores the packets addressed to D? **Justify** your answer.

Answers and justifications:

more space to answer on the next page ☞

Answers to Question 2 continued:

continues on back ☞

Question 3 – OS Basics [25 points]

3.1 – Memory accesses [5 points]

Together with a friend of yours, you look at the assembly code of a program, and you notice that it does not contain any explicit memory read (load) or write (store) CPU instructions: it does not contain any CPU instruction like "`mov rax, [0x0]`" or "`mov [0xb], rax`" that read the contents of some memory address into a register or write to a memory address the contents of a register.

Your friend reasons: "This means that executing this program does not trigger any memory access."

Do you think your friend's reasoning is correct? **Justify** your answer.

Answer and justification:

3.2 – Memory vs. I/O [8 points]

Consider two processes, A and B.
The only thing that Process A does is to read N bytes of data from memory.
The only thing that Process B does is to read N bytes of data from disk.

Assume the following:

- The N bytes read by Process A are already in memory.
- The N bytes read by Process B are not already cached in the file-system (block) cache.
- There is no page fault and no memory-mapped I/O.

If any of these assumptions does not make sense to you, don't worry about it.

Say whether each of the following statements is True or False. **Justify** each answer in 1–2 sentences.

① Once the CPU starts running Process A, it is possible that Process A completes reading its N bytes without the CPU executing any kernel code.

② Once the CPU starts running Process B, it is possible that Process B completes reading its N bytes without the CPU executing any kernel code.

③ Assuming neither process is interrupted, Process A completes significantly faster than Process B.

④ Process B occupies the CPU for significantly more time than Process A.

Answers and justifications:

continues on back ☞

3.3 – Virtual vs. physical memory [12 points]

Each of the following four scenarios (①–④) gives you information about a computer system whose memory management relies on paging and, in particular, linear (one-level) page tables.

Each scenario is independent from all the other scenarios.

For each scenario, **compute** the maximum amount of physical memory that the computer system — not just an individual process — can address, or **explain** why you do not have enough information to compute it.

Do not assume anything beyond what is explicitly given in each scenario; e.g., do not assume that a page table fits within a page, or that a page table entry (PTE) has any particular structure.

① You are given:

- Size of frame: 256 bytes.
- Size of virtual page number (VPN): 10 bits.

② You are given:

- Size of page offset: 6 bits.
- Size of physical frame number (PFN): 11 bits.

③ You are given:

- Size of page table: 2048 bytes.
- Size of page table entry (PTE): 16 bits.
- Size of physical frame number (PFN): 11 bits.

④ You are given:

- At any point in time, the maximum amount of memory that can be used (allocated to processes and the kernel) without triggering any swapping to disk: 1 GiB.

Answers and/or explanations:

more space to answer on the next page ☞

Answers to subquestion 3.3 continued:

continues on back ☞

Question 4 – Processes, threads, files [30 points]

4.1 – Opening and closing a file [10 points]

Consider the following program excerpt:

```
1  ...
2  const char* filename = "/usr/log/test.txt";
3
4  int main(void)
5  {
6     int fd = open(filename, O_RDWR, ...);
7     close(fd);
8     return 0;
9  }
```

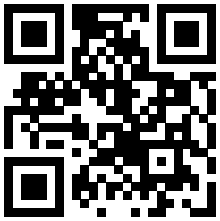
(This program does not accomplish anything, but one may have written it as a test.)

Suppose you compile this program and invoke it on your computer. Assume file "`/usr/log/test.txt`" already exists and you have the necessary permissions to open it in read/write mode.

① How many processes and how many threads does the Operating System (OS) of your computer create as a result of you invoking this program?

② How many block accesses does this program trigger? For each block access, state what is read from or written to the disk. **Justify** your answer.

Answers and justifications:



4.2 – With fork [10 points]

Answer the same two questions from subquestion 4.1, but for the following program excerpt:

```
1   ...
2   const char* filename = "/usr/log/test.txt";
3
4   int main(void)
5   {
6      int pid1 = fork();
7
8      int fd = open(filename, O_RDWR, ...);
9      close(fd);
10
11     if (pid1 > 0) waitpid(pid1, ...);
12
13     return 0;
14  }
```

Answers and justifications:

continues on back ☞

4.3 – With threads [10 points]

Answer the same two questions (repeated here for convenience) but for the program excerpt below.

① How many processes and how many threads does the Operating System (OS) of your computer create as a result of you invoking this program?

② How many block accesses does this program trigger? For each block access, state what is read from or written to the disk. **Justify** your answer.

Also answer this third question:

③ Does this program cause any resource to be accessed concurrently?
Justify your answer. If you answer yes, say which resource and which threads are involved.

```
1   ...
2   const char* filename = "/usr/log/test.txt";
3
4   void* fileAccess(void* unused)
5   {
6      int fd = open(filename, O_RDWR, ...);
7      close(fd);
8      return NULL;
9   }
10
11  int main(void)
12  {
13     int pid1 = fork();
14
15     pthread_t t1;
16     pthread_create(&t1, NULL, fileAccess);
17     pthread_join(t1, NULL);
18
19     int fd = open(filename, O_RDWR, ...);
20     close(fd);
21
22     if (pid1 > 0) waitpid(pid1, ...);
23
24     return 0;
25  }
```

Answers and justifications:

Question 5 – C-rching [10 points]

This question is FOR ALL STUDENTS.

Using the C language **and some pointer arithmetic**, write a function `find_all()` that takes a string and a function pointer as input and returns a list/set/array[1] **of pointers** to all the positions in the string for which the function passed as argument returns 1. These functions take an integer and return an integer which is either 0 or 1.

For instance, making use of the standard functions `int isupper(int c);` and `int isspace(int c);`

- `find_all("Hello! How are you?", isupper)` will return a list/set/array of 2 pointers pointing to each of the two 'H';
- and `find_all("Hello! How are you?", isspace)` will return a list/set/array of 3 pointers pointing to each of the three whitespaces.

Notes:

1. It's up to you to **define** the return type of the `find_all()` function.
2. In C, a `char` can silently be cast to and cast from an `int`.

Answer:

[1] It's up to you to **define** the return type of the `find_all()` function.

continues on back ☞

Question 6 – Project-related questions [15 points]

These 3 subquestions are RELATED TO THE PROJECT.

6.1 – The ring [3 points]

Assume a ring with 5 servers and 7 nodes, the SHA of which are (here simplified to their first 2 bytes):

server	server 1	server 2	server 2	server 3	server 3	server 4	server 5
SHA	`6078`	`b0ff`	`f0ad`	`207f`	`70a9`	`c0af`	`40d1`

With the following request:

```
./dkvs-client get -r 2 -n 4 -- abc
```

(i.e. $R = 2$ and $N = 4$), we ask that ring for the value of the key "abc", the SHA of which starts with `a999`.

Assuming that all the servers are up, that for the key "abc", servers 1 and 2 store the value "val1", servers 4 and 5 store the value "val2" and server 3 does not store any value, **what value (or error) do we get from the ring?**

Justify your answer.

Do NOT write anything here!

6.2 – From network messages to key-value pair [5 points]

In the project, the key-value network messages, the size of which is known, were in the format: key, null-char, value; for instance: `hello\0folks`, of size 11.
They can thus be represented in C as:

```
struct message {
  char* content;
  size_t size;
};
```

We here want to extract from such a message, the key and the value, as a pair of C strings:

```
struct kv_pair {
    char* key;
    char* value;
};
```

i.e. write a function `message_to_kv_pair()` which takes a `struct message` and a `struct kv_pair` and extracts the key and the value from the message to put them in the key-value pair of C strings.
This function returns an `int`, which is -1 in case of error and 0 if everything went fine.

Define the `message_to_kv_pair()` function:

continues on back ☞

6.3 – Node content ownership [7 points]

At some point in the project, the node structure has to store its IP address as a C string. However, the ownership model for this address was not specified and we saw 3 different implementations:

- statically allocated ownership;
- dynamically allocated ownership;
- not owner;

which can be summarized by the following 3 codes:

```
struct node {
  char* address;
};
```

implementation:

```
struct node {
  const char* address;
};
```

implementation:

```
#define IP_ADDR_SZ 15

struct node {
  char address[IP_ADDR_SZ + 1];
};
```

implementation:

① Below each of the 3 above code, write which implementation it corresponds to: statically allocated ownership, dynamically allocated ownership, not owner.

② Here is how we would typically use the node structure:

```
struct node node1;
node_init(&node1, "127.0.0.1");
```

For each of the above 3 implementations, write (below) the code to initialize a node and the code to release it (add the comment "`// do nothing`" if there is nothing to do):

```
struct node {
  char* address;
};
typedef struct node node_t;



void node_init(node_t* n,
        const char* address)
{









}

void node_end(node_t* n)
{



}
```

```
struct node {
  const char* address;
};
typedef struct node node_t;



void node_init(node_t* n,
        const char* address)
{









}

void node_end(node_t* n)
{



}
```

```
#define IP_ADDR_SZ 15

struct node {
  char address[IP_ADDR_SZ + 1];
};
typedef struct node node_t;

void node_init(node_t* n,
            const char* address)
{









}

void node_end(node_t* n)
{



}
```

Do NOT write anything here!

③ Assume now we have to make some temporary reference copy of `node1`; here is a simplified example (but the general spirit would be the same):

```
1   struct node node1;
2   node_init(&node1, some_address);
3
4   // ...some use of node1...
5
6   // Now here we need a temporary copy to reference node1's content:
7   struct node tmp_node = node1;
8
9   // ...use of tmp_node without modifying its IP address...
10
11  // From here we don't need tmp_node anymore.
12  node_end(&tmp_node);
13
14  // ...rest of the code (still using node1)...
15
16  node_end(&node1);
```

For each of the 3 implementations, **say** whether the above proposed code excerpt would **(a) compile** and **(b) behave correctly**.
If not, **say which line(s)** should be corrected **and how** to have the above code excerpt compile and run properly.

Answer:

Empty page to answer whatever question if more space is needed.
PLEASE INDICATE THE QUESTION NUMBER YOU ARE ANSWERING.