Second assignment

Statement and concepts

INFO-0010

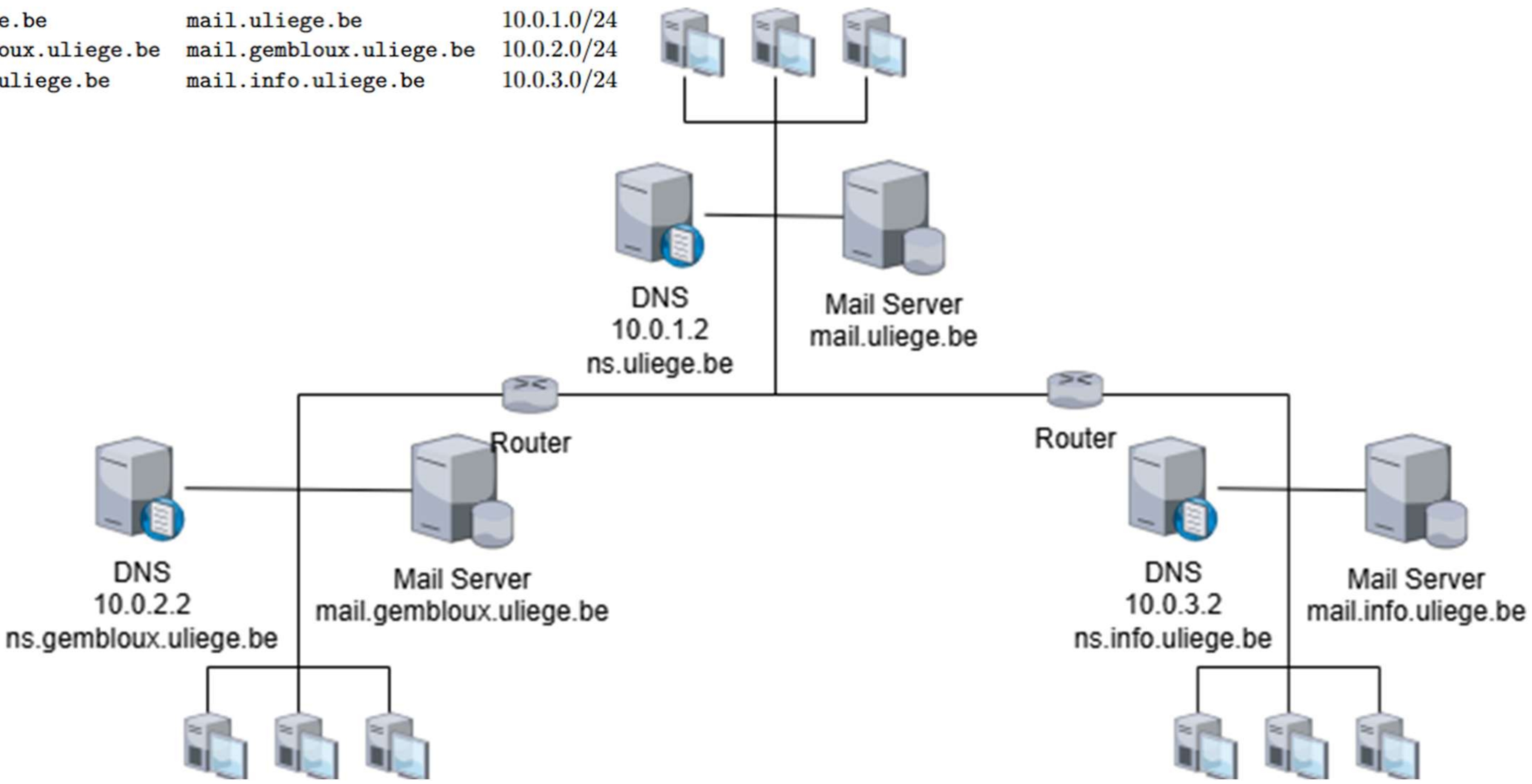
Outline

- Statement
- Implementation of concepts

Objective

Mail Server using SMTP, POP3 and IMAP

`uliege.be`	`mail.uliege.be`	10.0.1.0/24
`gembloux.uliege.be`	`mail.gembloux.uliege.be`	10.0.2.0/24
`info.uliege.be`	`mail.info.uliege.be`	10.0.3.0/24

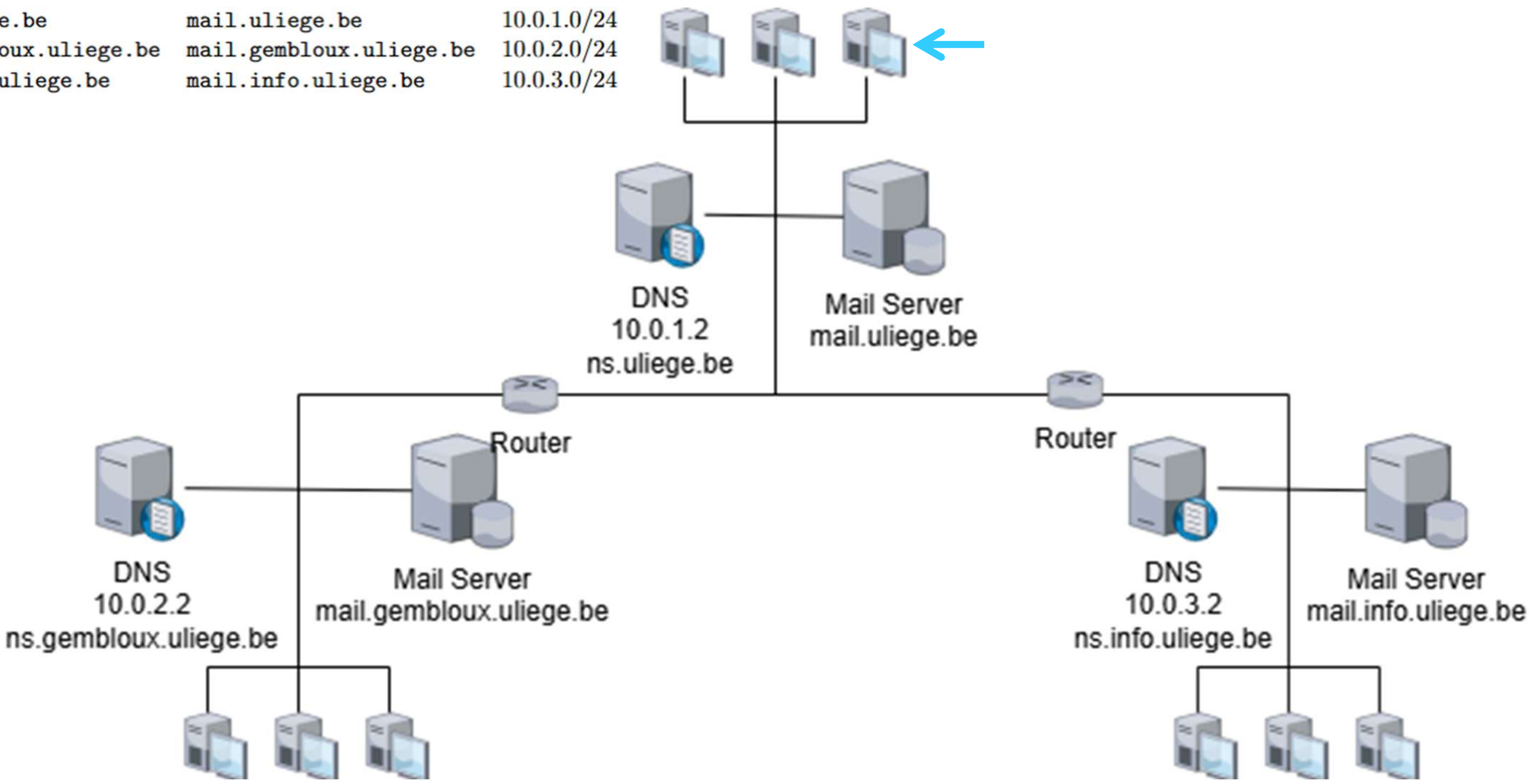


Overview

User dcd@info.uliege.be sends an email to vj@gembloux.uliege.be

1. User dcd@info.uliege.be connects to his email client

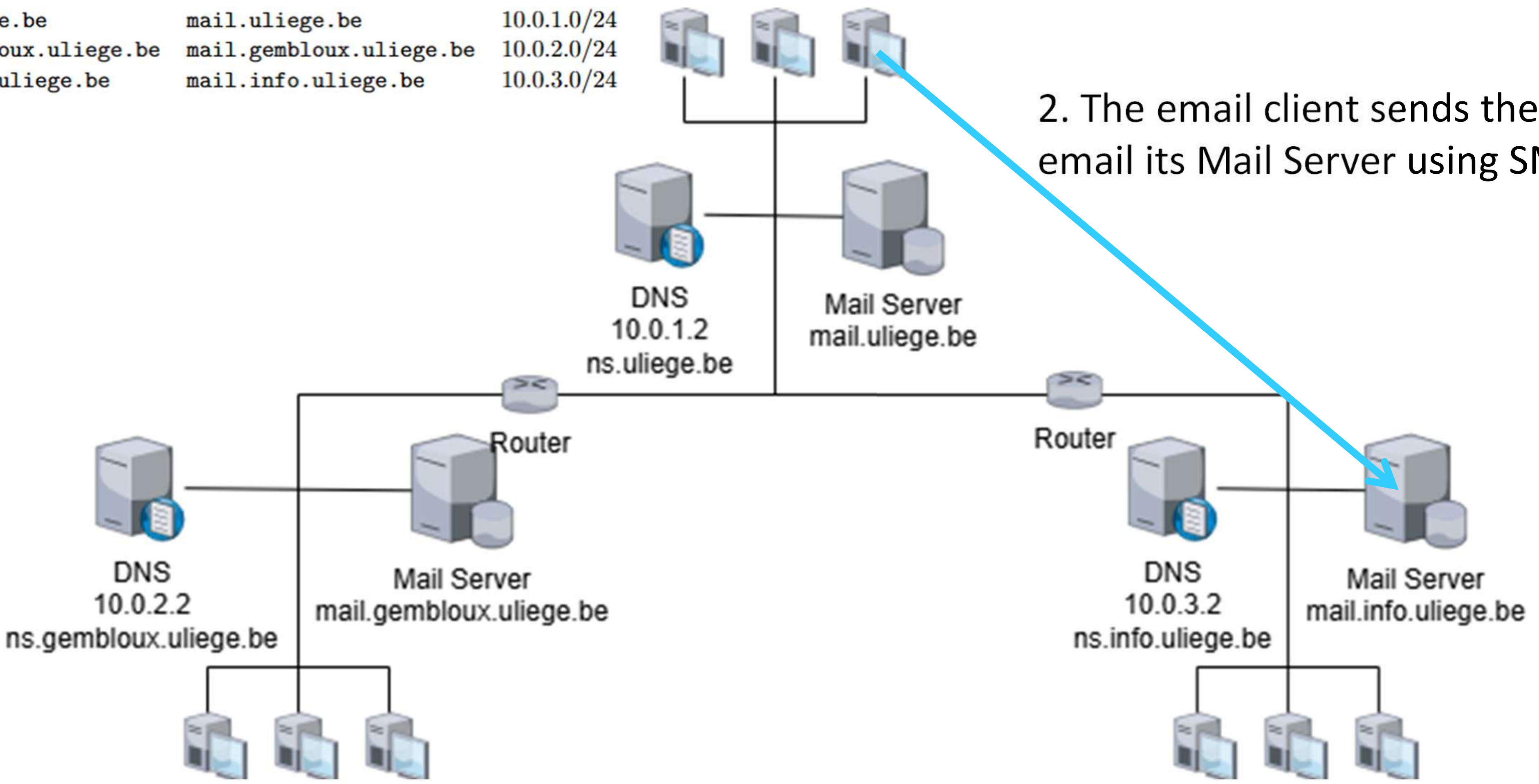
uliege.be	mail.uliege.be	10.0.1.0/24
gembloux.uliege.be	mail.gembloux.uliege.be	10.0.2.0/24
info.uliege.be	mail.info.uliege.be	10.0.3.0/24



Overview

User dcd@info.uliege.be sends an email to vj@gembloux.uliege.be

uliege.be	mail.uliege.be	10.0.1.0/24
gembloux.uliege.be	mail.gembloux.uliege.be	10.0.2.0/24
info.uliege.be	mail.info.uliege.be	10.0.3.0/24

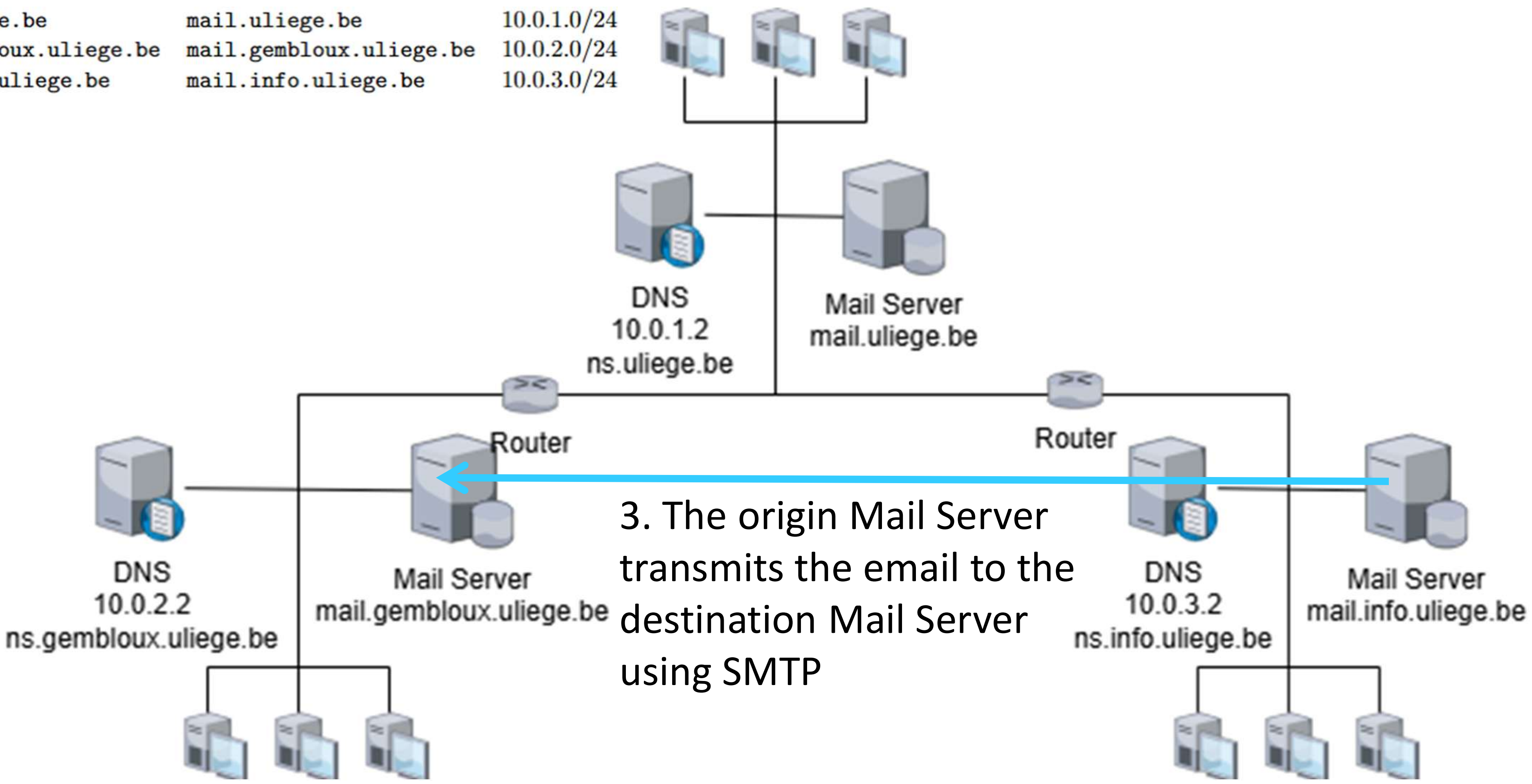


2. The email client sends the email its Mail Server using SMTP

Overview

User dcd@info.uliege.be sends an email to vj@gembloux.uliege.be

uliege.be	mail.uliege.be	10.0.1.0/24
gembloux.uliege.be	mail.gembloux.uliege.be	10.0.2.0/24
info.uliege.be	mail.info.uliege.be	10.0.3.0/24

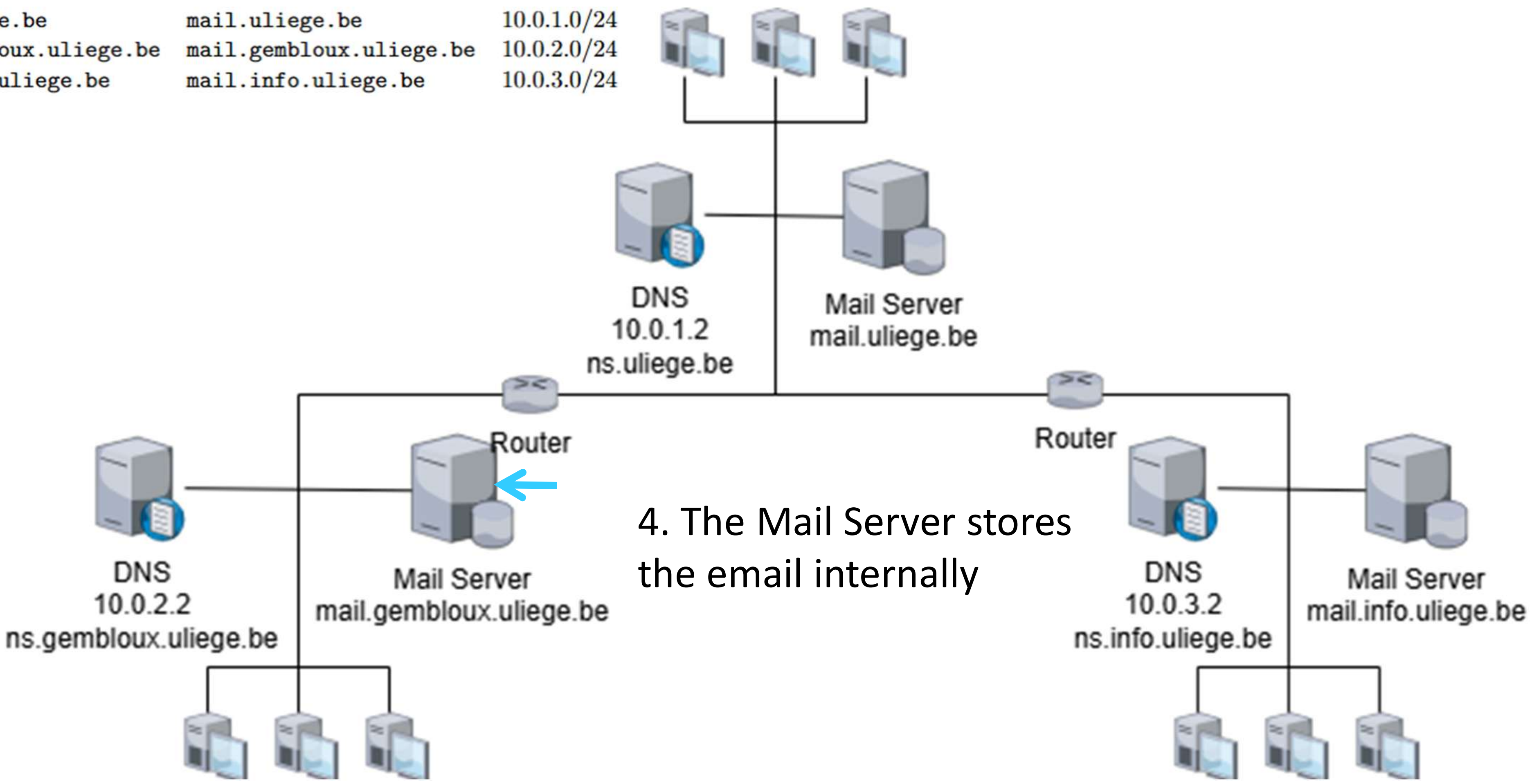


3. The origin Mail Server transmits the email to the destination Mail Server using SMTP

Overview

User dcd@info.uliege.be sends an email to vj@gembloux.uliege.be

uliege.be	mail.uliege.be	10.0.1.0/24
gembloux.uliege.be	mail.gembloux.uliege.be	10.0.2.0/24
info.uliege.be	mail.info.uliege.be	10.0.3.0/24

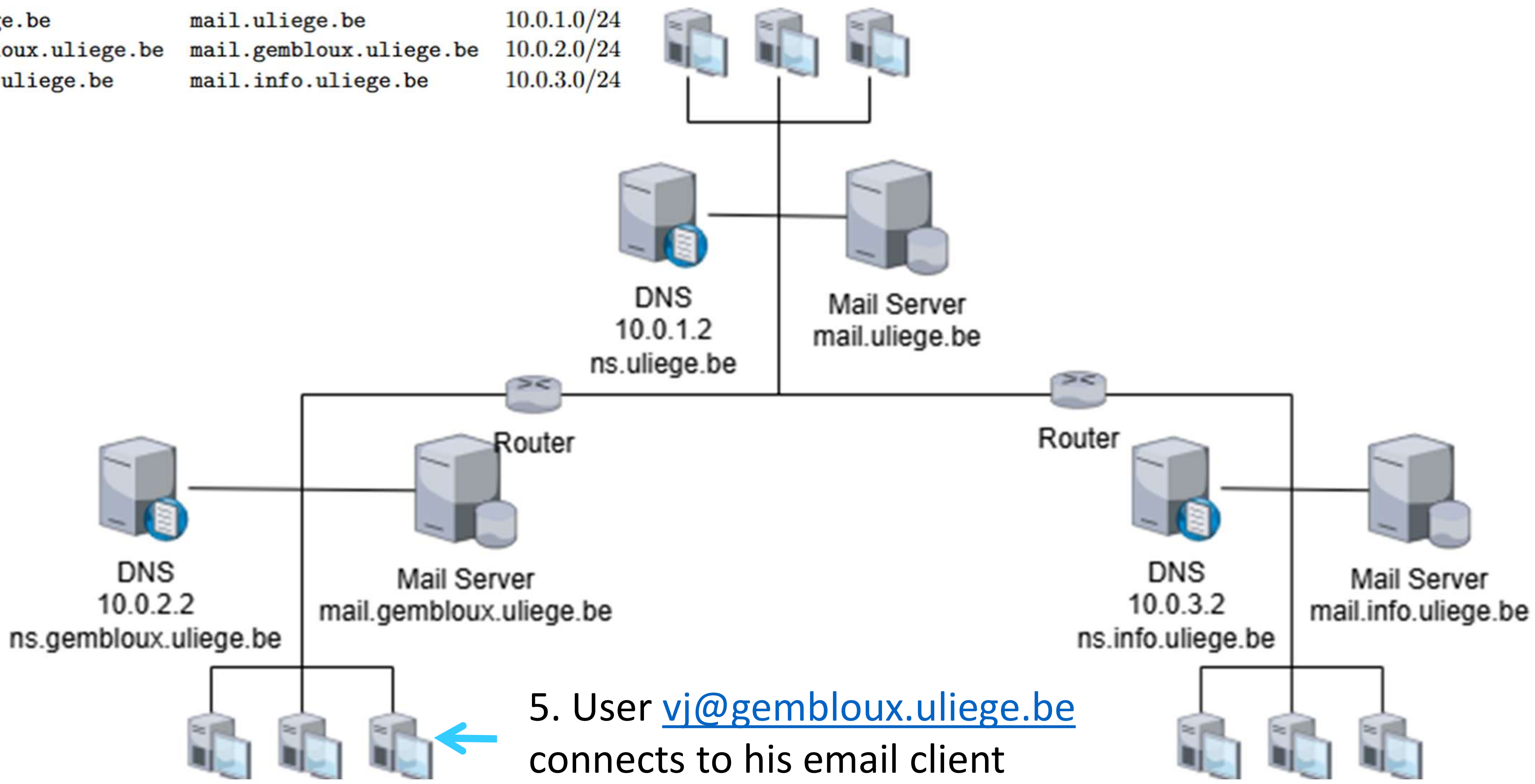


4. The Mail Server stores the email internally

Overview

User dcd@info.uliege.be sends
an email to vj@gembloux.uliege.be

uliege.be	mail.uliege.be	10.0.1.0/24
gembloux.uliege.be	mail.gembloux.uliege.be	10.0.2.0/24
info.uliege.be	mail.info.uliege.be	10.0.3.0/24

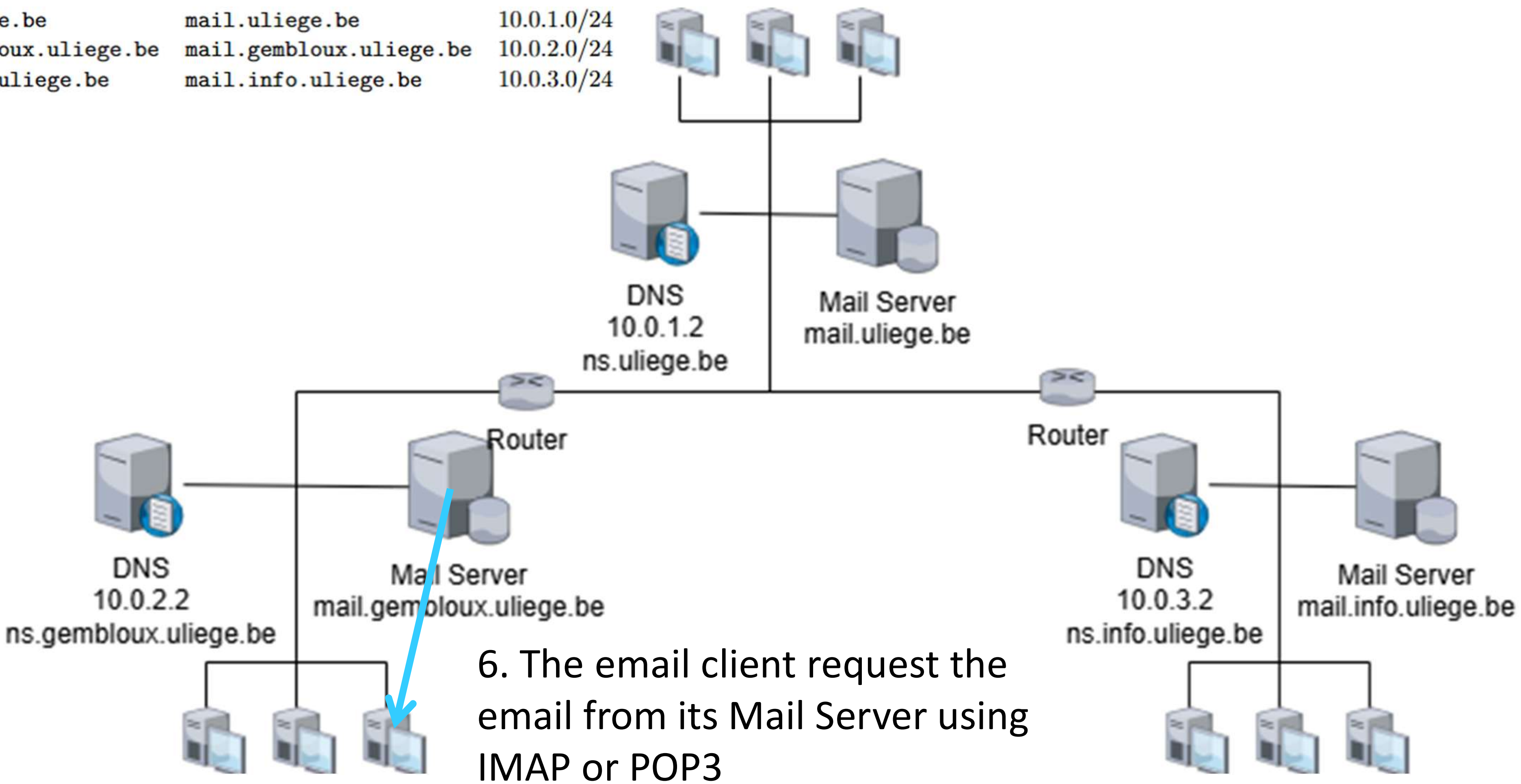


5. User vj@gembloux.uliege.be
connects to his email client

Overview

User dcd@info.uliege.be sends an email to vj@gembloux.uliege.be

uliege.be	mail.uliege.be	10.0.1.0/24
gembloux.uliege.be	mail.gembloux.uliege.be	10.0.2.0/24
info.uliege.be	mail.info.uliege.be	10.0.3.0/24



6. The email client request the email from its Mail Server using IMAP or POP3

SMTP protocol

- Simple Mail Transfer Protocol
- Applicative protocol over TCP
- Sending and transferring emails between mail client and mail servers
- Text-oriented protocol
- Port 25
- Commands and status code: cf RFC821, RFC5321

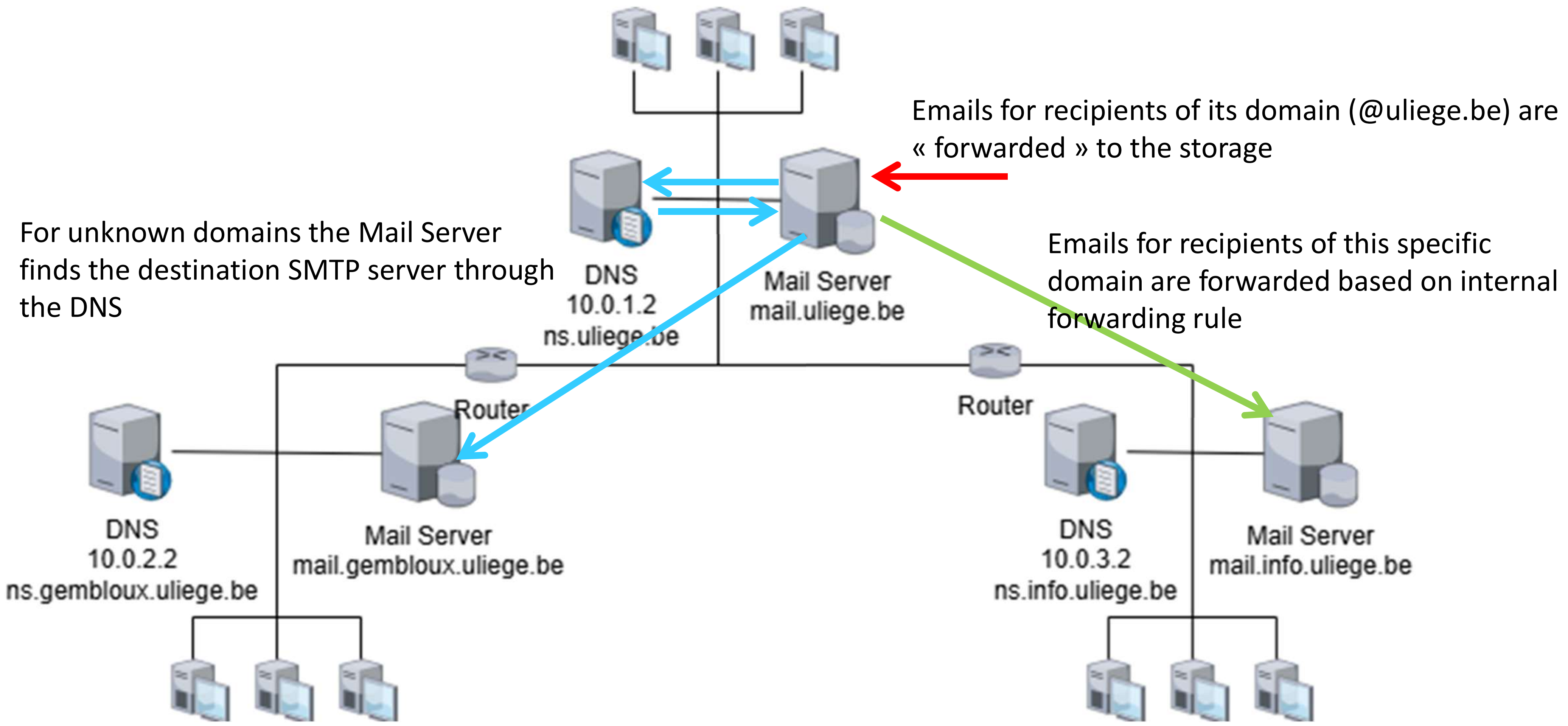
SMTP protocol

- Example of SMTP Session

```
S: 220 mail.uliege.be Service ready
C: HELO uliege.be
S: 250 mail.uliege.be greets uliege.be
C: MAIL FROM:<dcd@uliege.be>
S: 250 OK
C: RCPT TO:<vj@uliege.be>
S: 250 OK
C: DATA
S: 354 End data with <CRLF>.<CRLF>
C: From: dcd@uliege.be
C: To: vj@uliege.be
C: Subject: Test message
C:
C: Hello,
C: This is a test message sent within uliege.be.
C: .
S: 250 OK Message accepted for delivery
C: QUIT
S: 221 Bye
```

SMTP protocol

- Determines where to forward based on
  - email recipient
  - internal forwarding rules
  - DNS Lookup


Emails for recipients of its domain (@uliege.be) are « forwarded » to the storage
For unknown domains the Mail Server finds the destination SMTP server through the DNS
Emails for recipients of this specific domain are forwarded based on internal forwarding rule
DNS 10.0.1.2 ns.uliege.be
Mail Server mail.uliege.be
Router
Router
DNS 10.0.2.2 ns.gembloux.uliege.be
Mail Server mail.gembloux.uliege.be
DNS 10.0.3.2 ns.info.uliege.be
Mail Server mail.info.uliege.be

POP3 protocol

- Post Office Protocol version 3
- Applicative protocol over TCP
- Retrieving and deleting emails from the mail servers
- Text-oriented protocol
- Port 110
- Commands and status code: cf RFC1939

POP3 protocol

- Example of POP3 Session

```
S: +OK POP3 server ready
C: USER dcd@uliege.be
S: +OK
C: PASS password
S: +OK Mailbox locked and ready
C: STAT
S: +OK 2 680
C: LIST
S: +OK 2 messages (680 octets)
S: 1 320
S: 2 360
S: .
C: RETR 1
S: +OK 320 octets
S: From: vj@uliege.be
S: To: dcd@uliege.be
S: Subject: Test message
S:
S: Hello,
S: This is a message retrieved through POP3.
S: .
C: DELE 1
S: +OK Message marked for deletion
C: QUIT
S: +OK Goodbye
```

IMAP protocol

- Internet Message Access Protocol
- Applicative protocol over TCP
- Allows multiple clients to access the same mailbox
- Text-oriented protocol
- Port 143
- For this project: IMAP4rev1
- Commands and status code: cf RFC3501

IMAP protocol

- Example of IMAP Session

```
S: * OK IMAP server ready
C: A1 CAPABILITY
S: * CAPABILITY IMAP4rev1 UID
S: A1 OK CAPABILITY completed
C: A2 LOGIN dcd@uliege.be password
S: A2 OK LOGIN completed
C: A3 LIST "" "*"
S: * LIST (\HasNoChildren) "/" INBOX
S: A3 OK LIST completed
C: A4 SELECT INBOX
S: * 2 EXISTS
S: A4 OK [UIDVALIDITY 12345] SELECT completed
C: A5 UID FETCH 1:* (UID FLAGS BODY[])
S: * 1 FETCH (UID 1001 FLAGS (\Seen) BODY[] {120}
S: From: vj@uliege.be
S: To: dcd@uliege.be
S: Subject: Meeting reminder
S:
S: Reminder: Meeting at 10:00.
S: )
S: A5 OK FETCH completed
C: A6 LOGOUT
S: * BYE IMAP server logging out
S: A6 OK LOGOUT completed
```

IMAP protocol

- Mailbox
  - In IMAP, a mailbox corresponds to a folder (e.g. INBOX)
  - Each mailbox stores its own independent list of messages/emails
- UID (Unique Identifier)
  - A unique, increasing number assigned to each message in a mailbox
  - Each new message gets a larger UID than any existing one
  - Persistent, they remain valid until they are permanently removed
- UIDVALIDITY
  - A unique number identifying a mailbox instance
  - Changes only when the mailbox is recreated (e.g. deleted and rebuilt)
  - Signals to clients that all previous UIDs are invalid

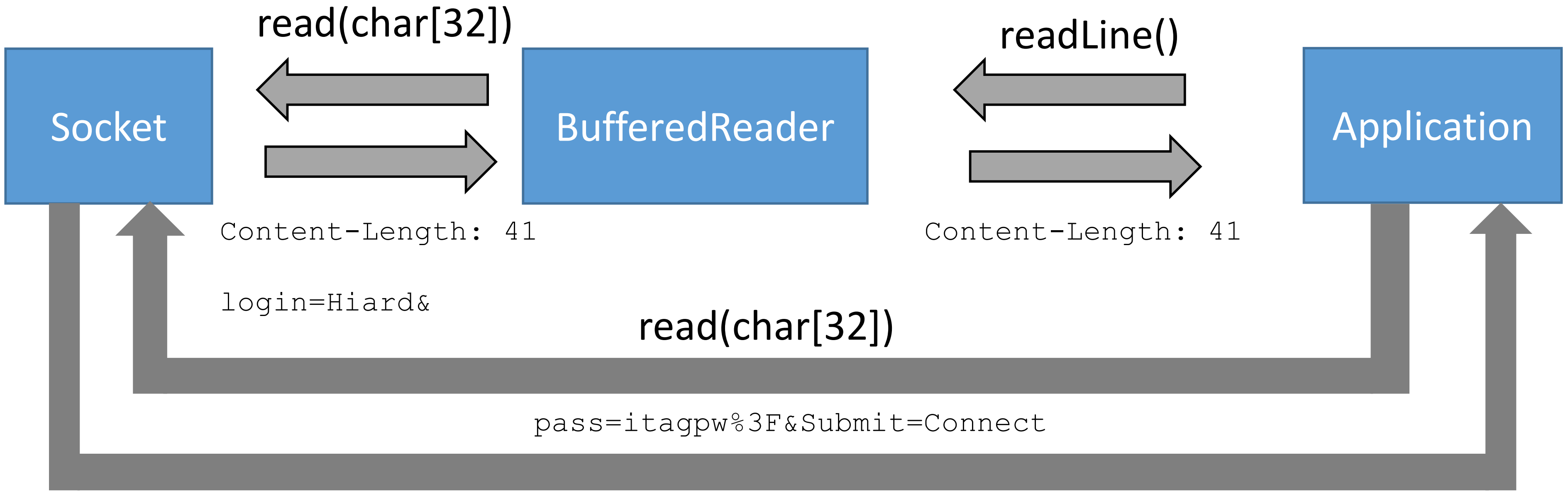
IMAP clients combine UIDVALIDITY + UID to uniquely identify messages, ensuring reliable synchronization across multiple devices

Beware of BufferedReader

- BufferedReader makes buffered reads!
  - i.e. maybe it already read more on the Socket than what you requested

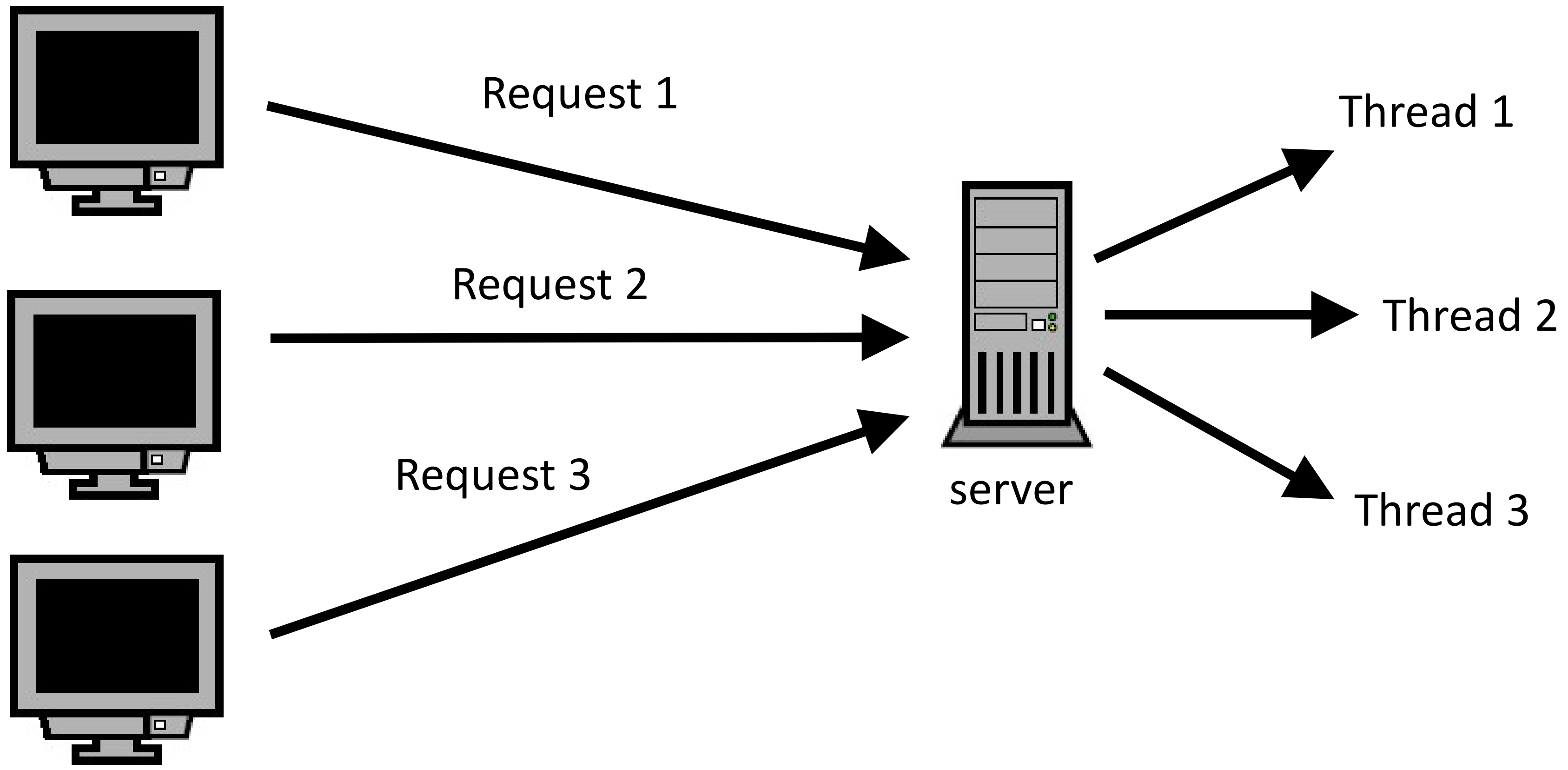
```
...
Content-Length: 41

login=Hiard&pass=itagpw%3F&Submit=Connect
```



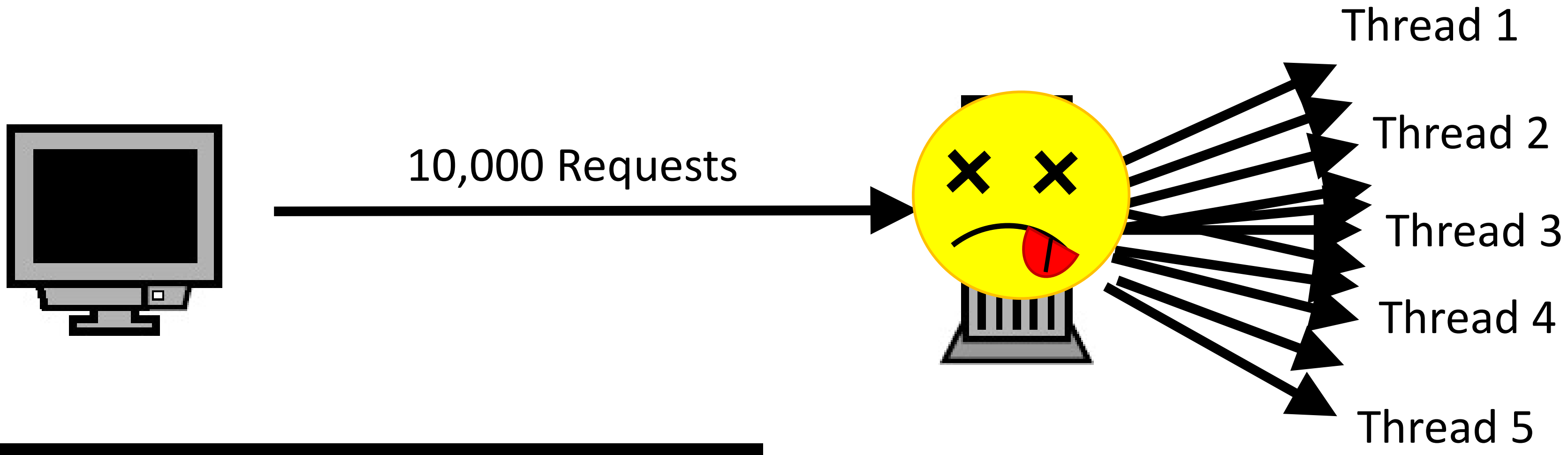
Thread pools

- Previously:



Thread pools

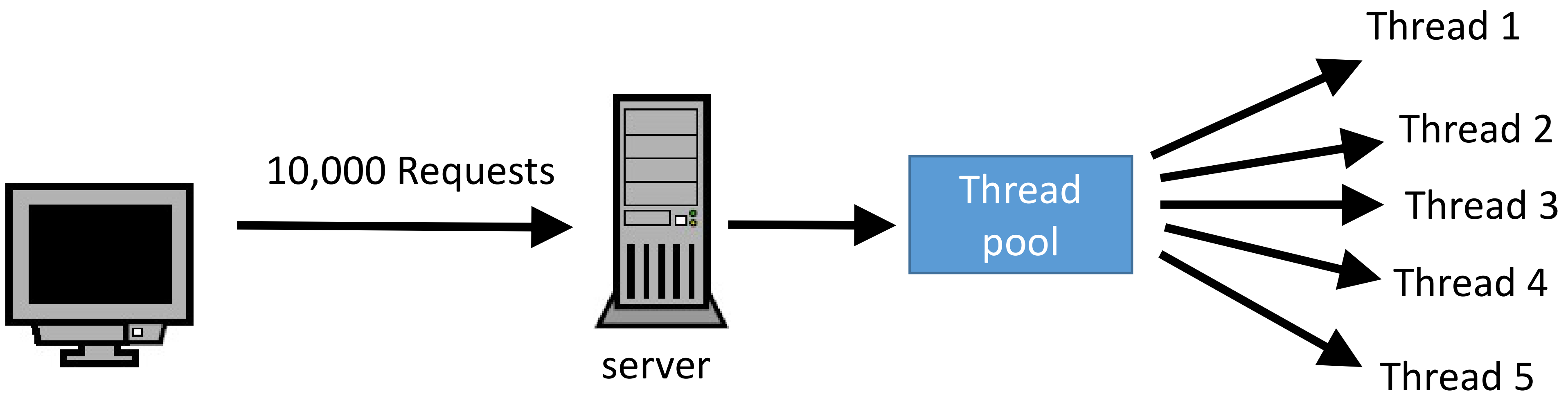
- Imagine the following attack:



```
Socket[] s = new Socket[10000];
for(int i = 0; i < 10000; i++)
{
     s[i] = new Socket("100.100.100.100",80);
}
```

Thread pools

- With a thread pool:



1. Handle the first 5 requests (9,995 remaining)
2. As soon as a thread finishes, it returns to the pool and receives a new task
3. When all tasks are done, each thread is back in the pool, ready for new tasks

Possible loss of speed performance, but increased robustness

Guidelines

- Deadline : 14th of December.
- Work by groups of two students.
- Guidelines of the first part still apply (Java 1.17, no package instructions, don't intercept CTRL-C, …).