Laboratory session: Cloud environment and Naming server

Distributed Systems

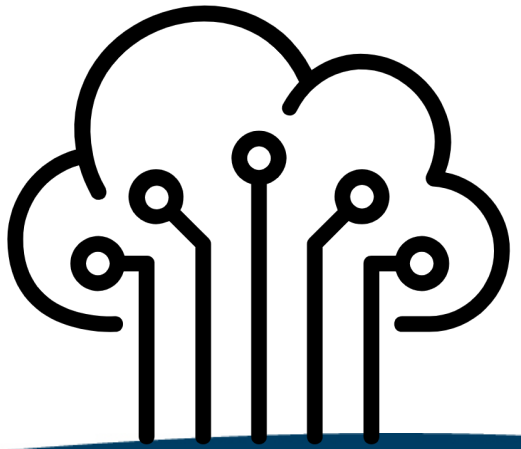
Universiteit Antwerpen

Overview of new working environment (1/3)

- IDLab cloud environment instead of RPi devices
- remote access to nodes that will be used in project implementation
- advantage: all nodes will be accessible 24/7
  - students can work on the labs and the project whenever needed
- disadvantage: persistent storage is not guaranteed

Overview of new working environment (2/3)

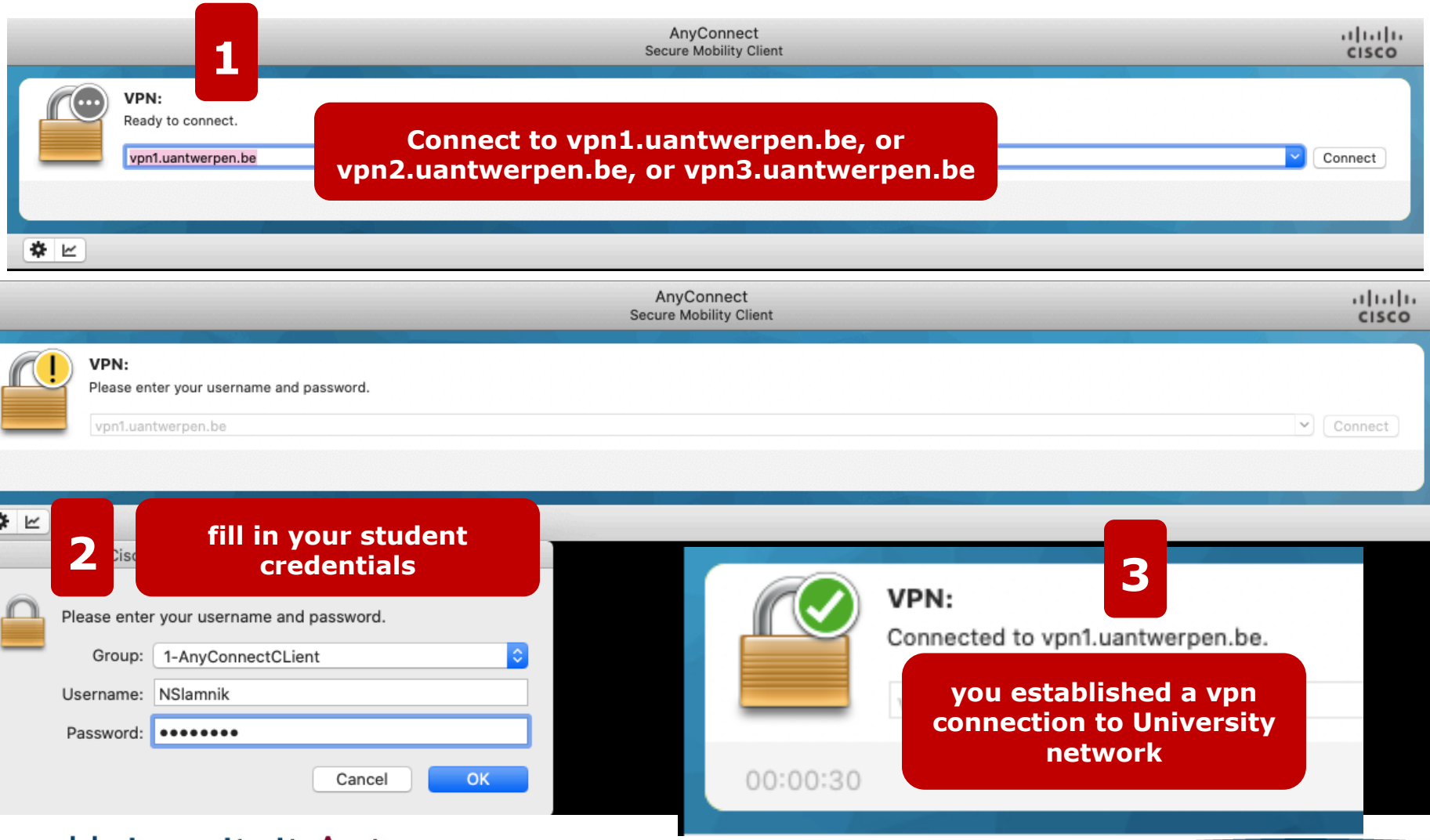
- each group will have 5 containers/machines with Debian
- to access machines students need to be connected to University network
  - no access at Campuses, use VPN!
  - vpn1.uantwerpen.be
  - vpn2.uantwerpen.be
  - vpn3.uantwerpen.be



Overview of new working environment (3/3)

- SSH connection to remote nodes is possible via following command:
  - ssh root@ip_address -p ssh_port
    - ip_address is the same for all nodes (i.e., dist-computing.idlab.uantwerpen.be), as nodes are developed as containers, but they are differentiated via ports
    - ssh_port is known for each node (check document Cloud access)
- brief recap of steps:
  1. connect to University network via VPN
  2. ssh to remote nodes via terminal on your local machine (laptop, PC) by typing the command stated above (ssh_port is different for each node, make sure to use the right ones defined for your group)

VPN connection to University network via Cisco AnyConnect Secure Mobility Client



Tasks for students (Work individually)

1. Develop REST-based server-client application in java.
   - the server provides a set of methods that are accessible for the client
   - server is a designed to work as a bank, and client as a customer
   - client should be able to:
     - get balance from account
     - add money on the account
     - get money from the account

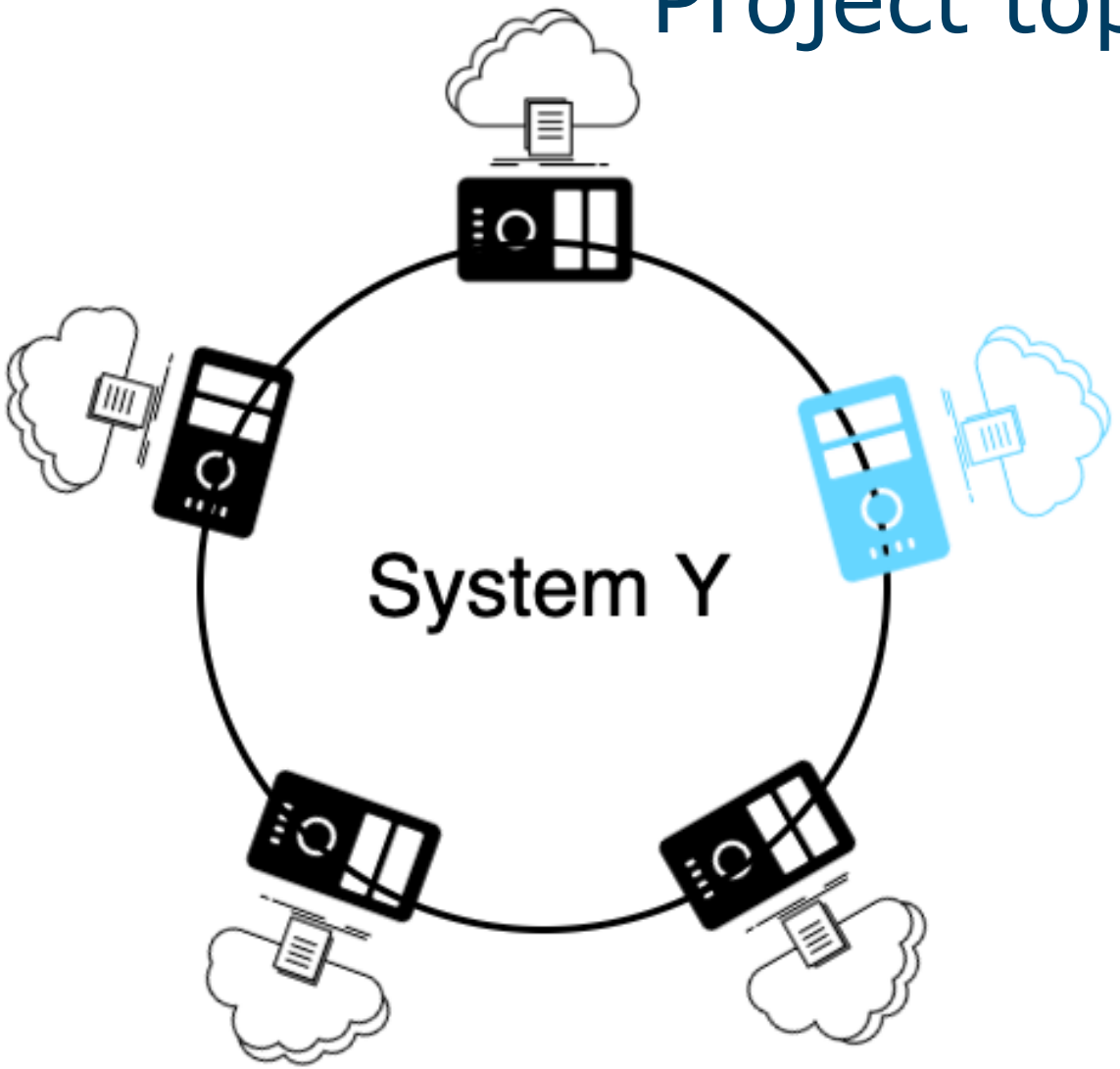
2. If bank account is joint, and two family members share it, extend exercise 1 with 2 clients who can do the same operations on the same account at the same time.

3. **For this** lab: Work in a group and try to run applications on top of the designated cloud resources, with one node acting as a server, while another one or two acting as clients.

Project

- Goal of the session:
  - to start working on the project
  - to develop Naming server functionality
- Team work:
  - work in your groups (available on Blackboard)
- Project:
  - to develop a distributed file system called system Y, consisted of nodes that are organized in a ring topology

Project topology



Naming server: Specification 1

- Naming server has a Map to store couples of (Integer, IP address)
  - Integer is a positive value limited 32768, as a result of a hashing function
  - IP address belongs to the unique node in your ring topology
- Hashing function:
  - the information about each node in topology is recorded by Naming server, i.e., it keeps track of all nodes
  - Naming server performs hashing functionality
  - Hashing function gets the name of the node as input, and returns its hash value as output
- The Map should be saved on the local disk (e.g., XML file)
- The Naming server must be able to add and remove nodes from the Map

Naming server: Specification 2

- each node stores some files locally (e.g., .png, and .jpg images, .pdf files, .txt, etc.) – owner of the file
- each node should be able to find out where is a specific file located (at which node)
  - this is possible via Naming server, as it is the only node that has a global information

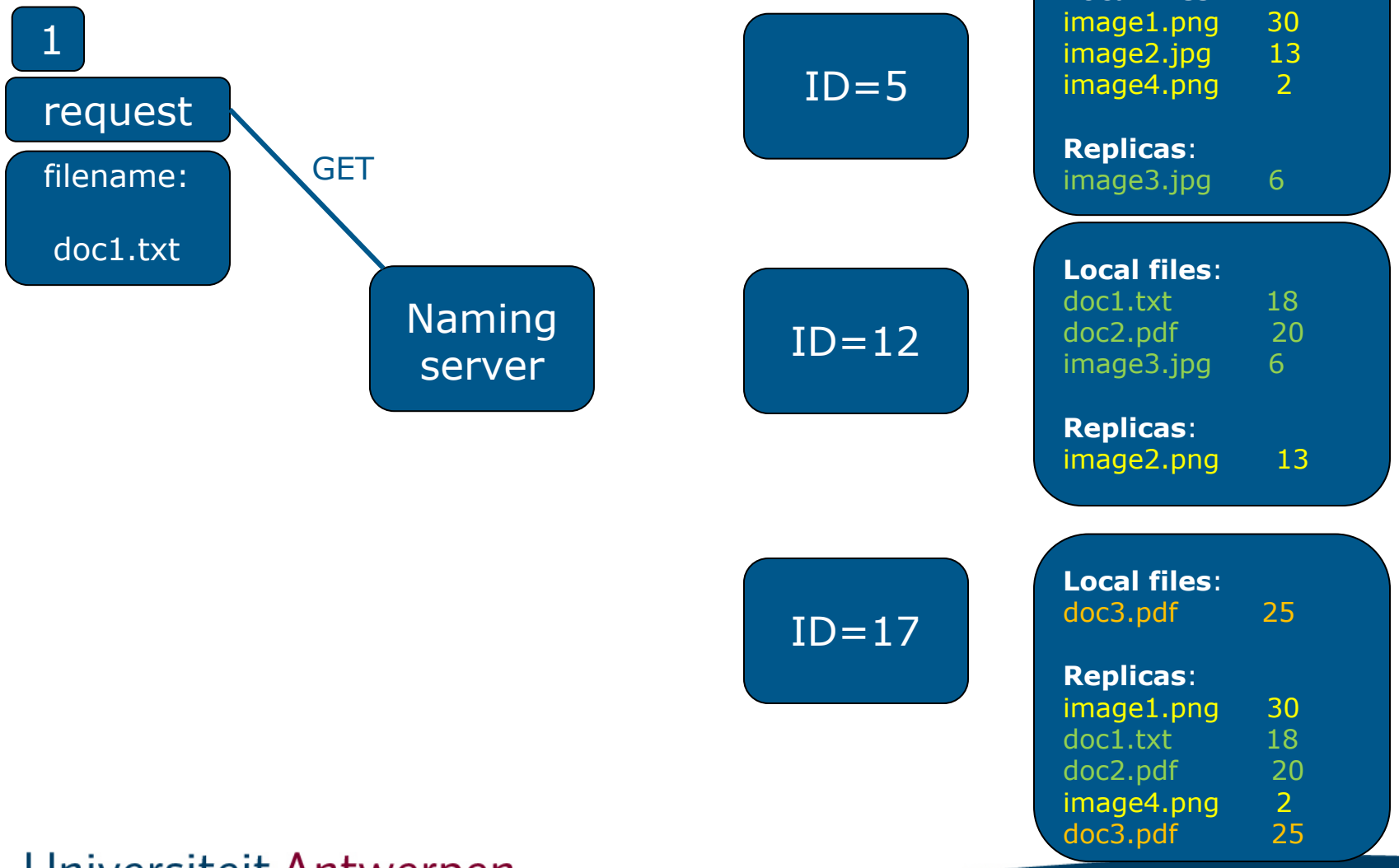
- node talks to Naming server using REST
  - it sends the name of the file that it is looking for
  - Naming server calculates a hash value of the filename
  - Naming server determines the node ID at which a file with filename is stored
  - Naming server extracts the IP address of this node from Map and sends it back to the node that sent request

Naming server: Specification 2

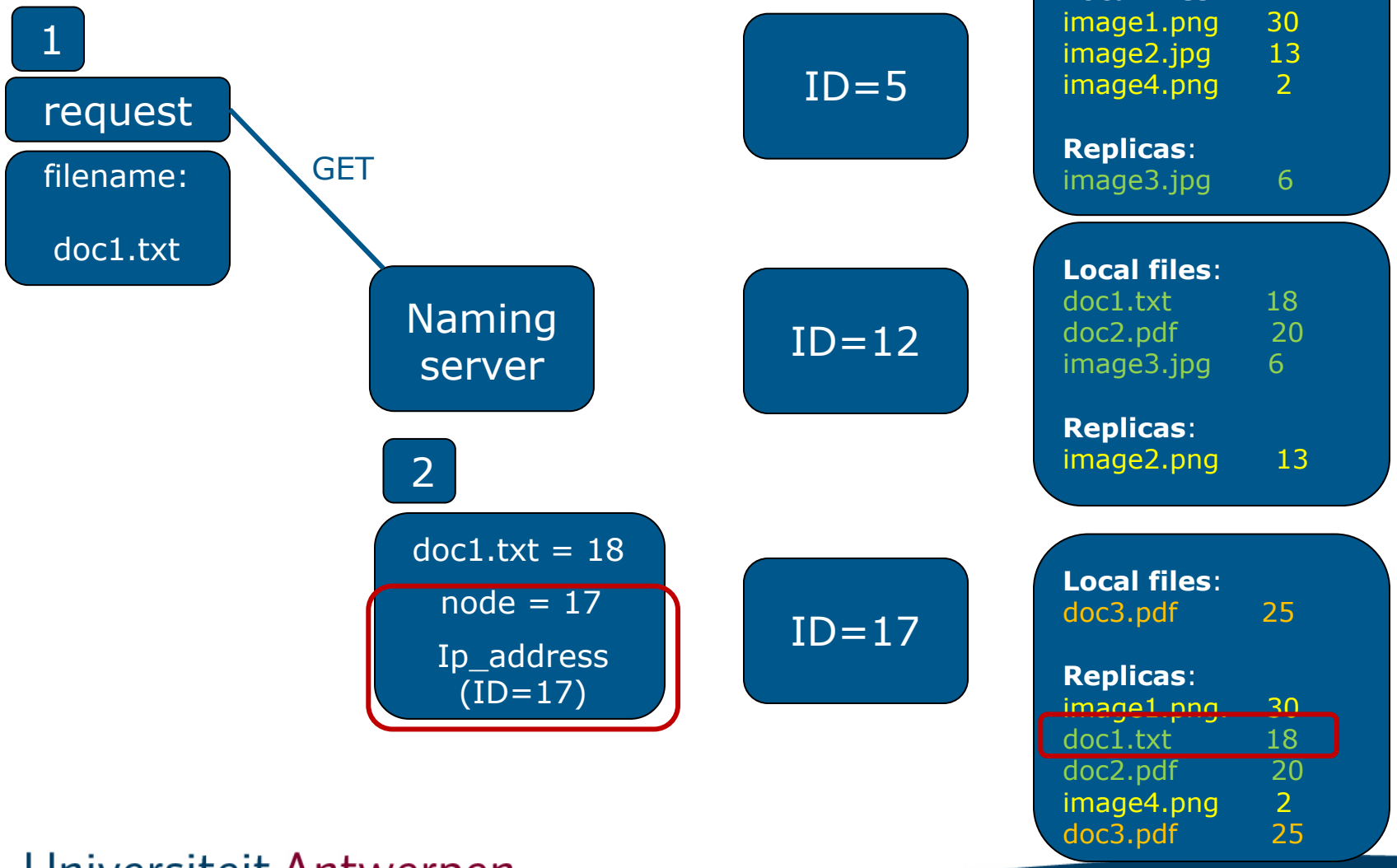
- Algorithm that Naming server uses to determine at which node is file stored:

Suppose N is the collection of nodes with a hash smaller than the hash of the filename. Then the node with the smallest difference between its hash and the file hash is the owner of the file. If N is empty, the node with the biggest hash stores the requested file.

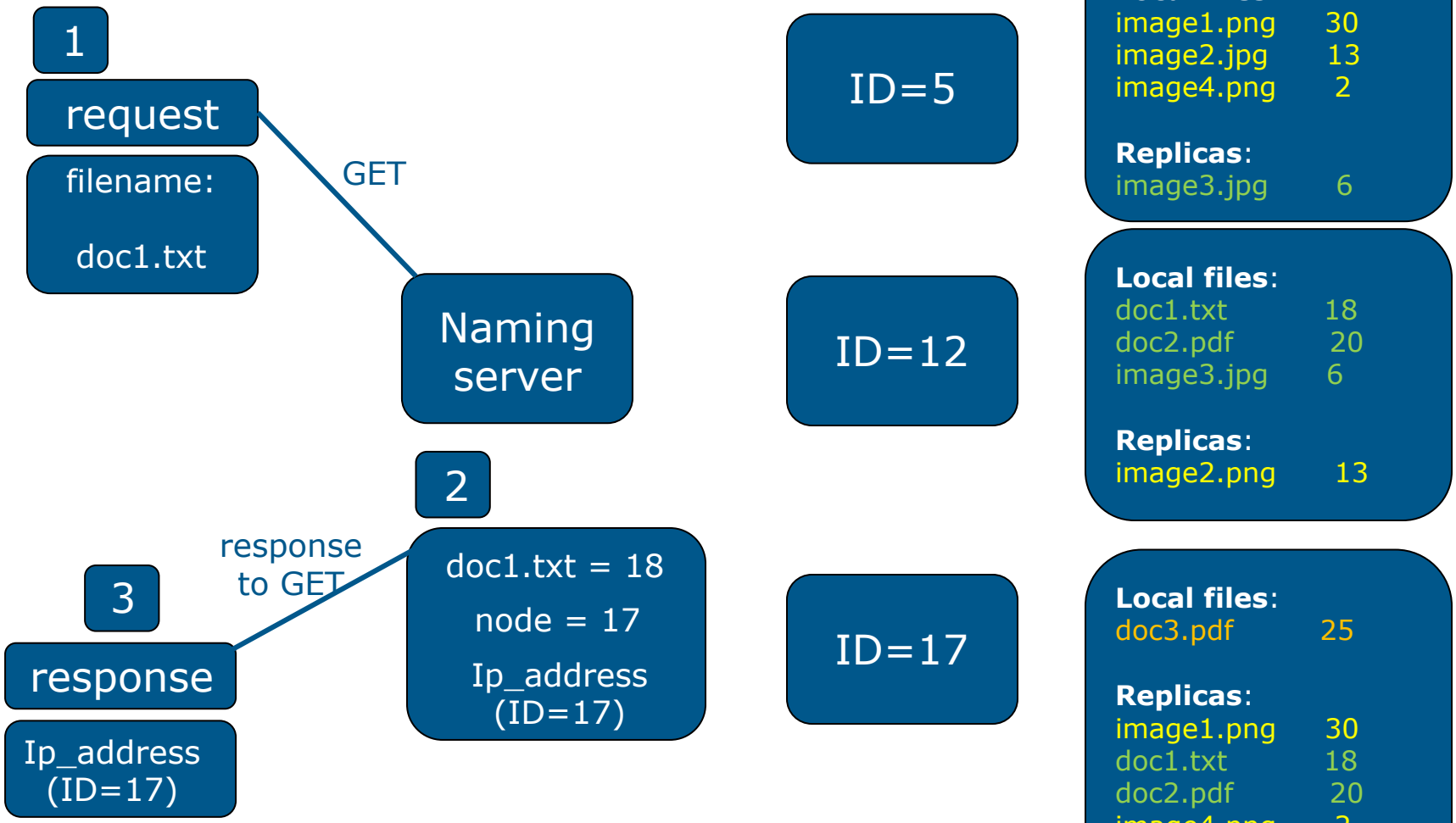
Naming server: Specification 2, Example



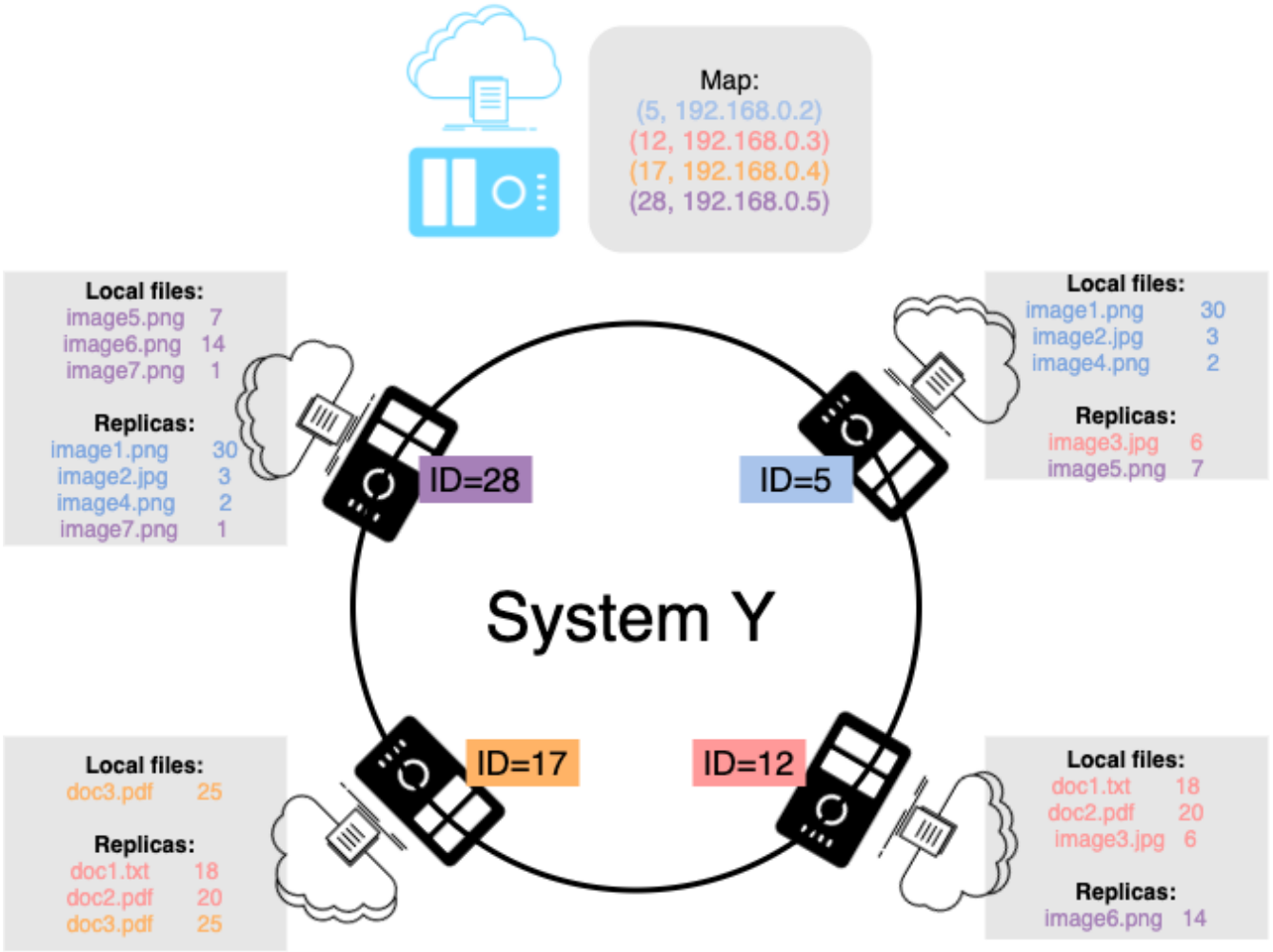
Naming server: Specification 2, Example



Naming server: Specification 2, Example



System Y - example



Tasks

- Tests that should be enabled by Naming server:

1. Add a node with a unique node name
2. Add a node with an existing node name
3. Send a filename and the IP address
4. Send a filename with a hash smaller than the smallest hash of the nodes
5. Send a filename and at the same time remove the node
6. Ask from two PCs for an IP address of a filename

Additional information

- Hashing
  - The standard hashing function in Java returns a value between – 2147483648 and + 2147483648. Since these values are too big for our project, we need to set new bounds for values between 0 and 327680.
  - mapping the default interval (-2147483648, 2147483648) to the required interval (0, 32768) is given below:
    - if max=2147483648, and min=-2147483648, then hashCode() value can be calculated as A=(hostname.hashCode()+max)*(32768/(max+abs(min)))
    - this way, if hostname.hashCode() or filename.hashCode() returns 2147483648, the result is A=32768, which is indeed a new maximum, and if hostname/filename.hashCode() returns -2147483648, A will be 0, which is a new minimum
    - any other hashcode value from the range (-2147483648, 2147483648), after this mapping denoted by A, will be in the range (0, 32768)

Additional information

- Map
  - A Map is an interface in Java. A Map consists of (key,value) couples where the key is unique in the Map. To get a value from the Map, you must use the key. The Map interface gives a set of functions with WHAT can be done with the data, not HOW it’s done. This is implemented in the classes which use the interface (HashMAP, TreeMAp, EnumMap)