- based on

Computer Networks

Lab 2 - Computer Networks

Manuel Ricardo, Filipe B. Teixeira, Eduardo Nuno Almeida

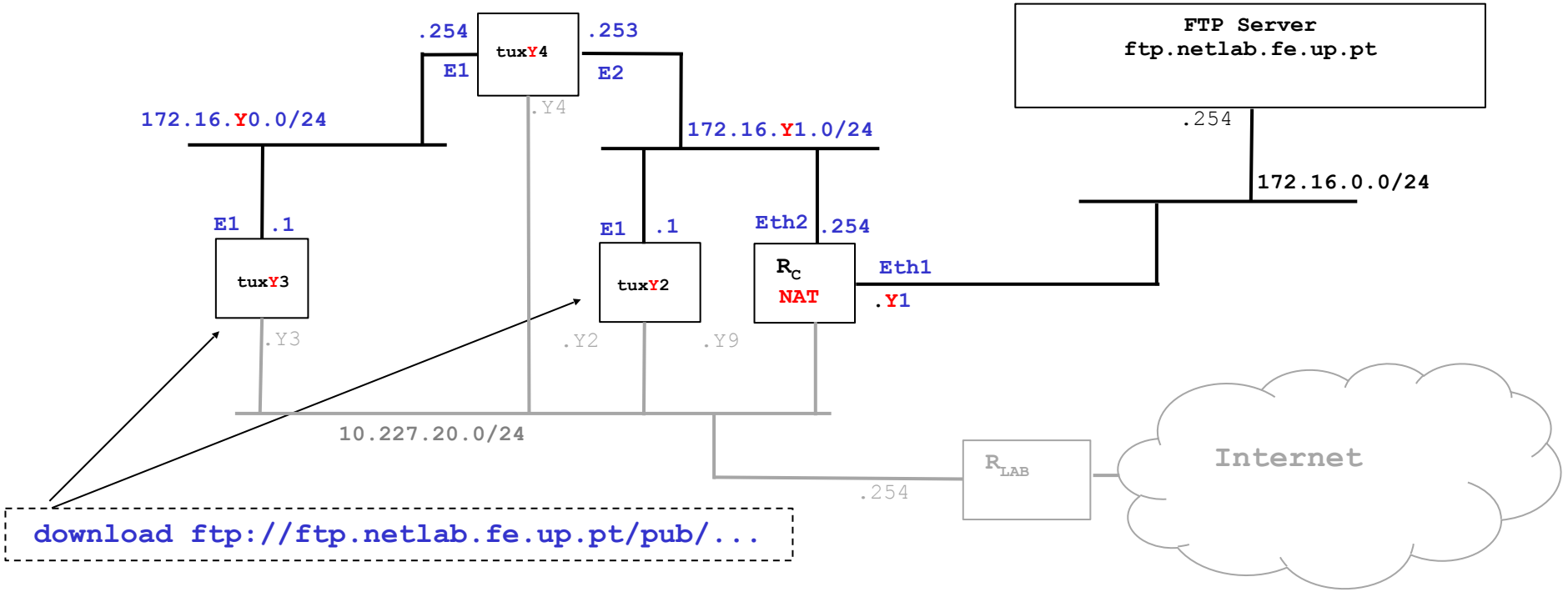
Universidade do Porto

Lab Work - Two parts

- Part A - Development of a download application

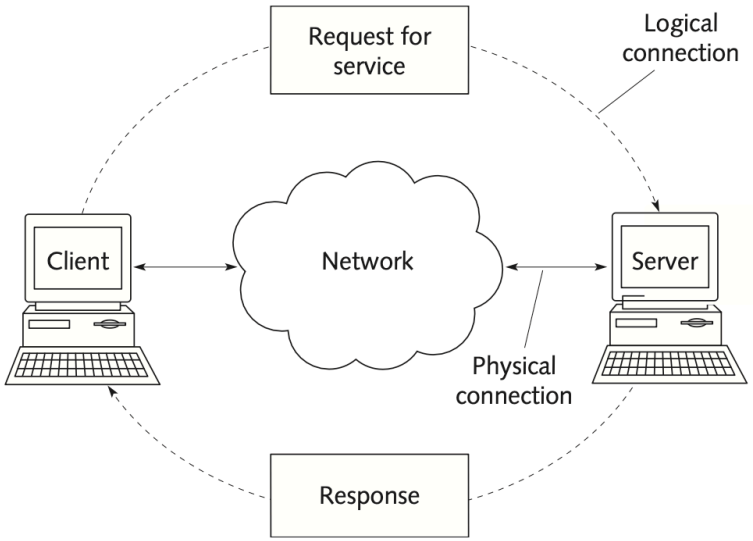
```
download ftp://ftp.netlab.fe.up.pt/pub/...
```

- Part B - Configuration and study of a computer network



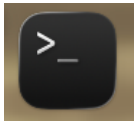
Part A

- development of a client-server application to transfer files between remote computers
  - client and server communicate using specific set of messages
    - client issues commands to the server
    - server send replies
      - format and meaning of messages (syntax and semantics) as well as sequence, are defined in the protocol FTP, File Transfer Protocol (IETF RFC 959)



Part B

- use linux based machines (workstations, switches and routers) to build an interconnection of two local networks and connect the set to the Internet
  - necessary to configure network interfaces of such equipment
    - IP addresses, routes
    - virtual networks or bridges
  - command line configurations
    - for Linux workstations, using a terminal and typing the appropriate linux commands



>> ifconfig eth0 172.16.10.1/24 ≡ >> ip address eth0 172.16.10.1/24

>> route -n ≡ >> ip route

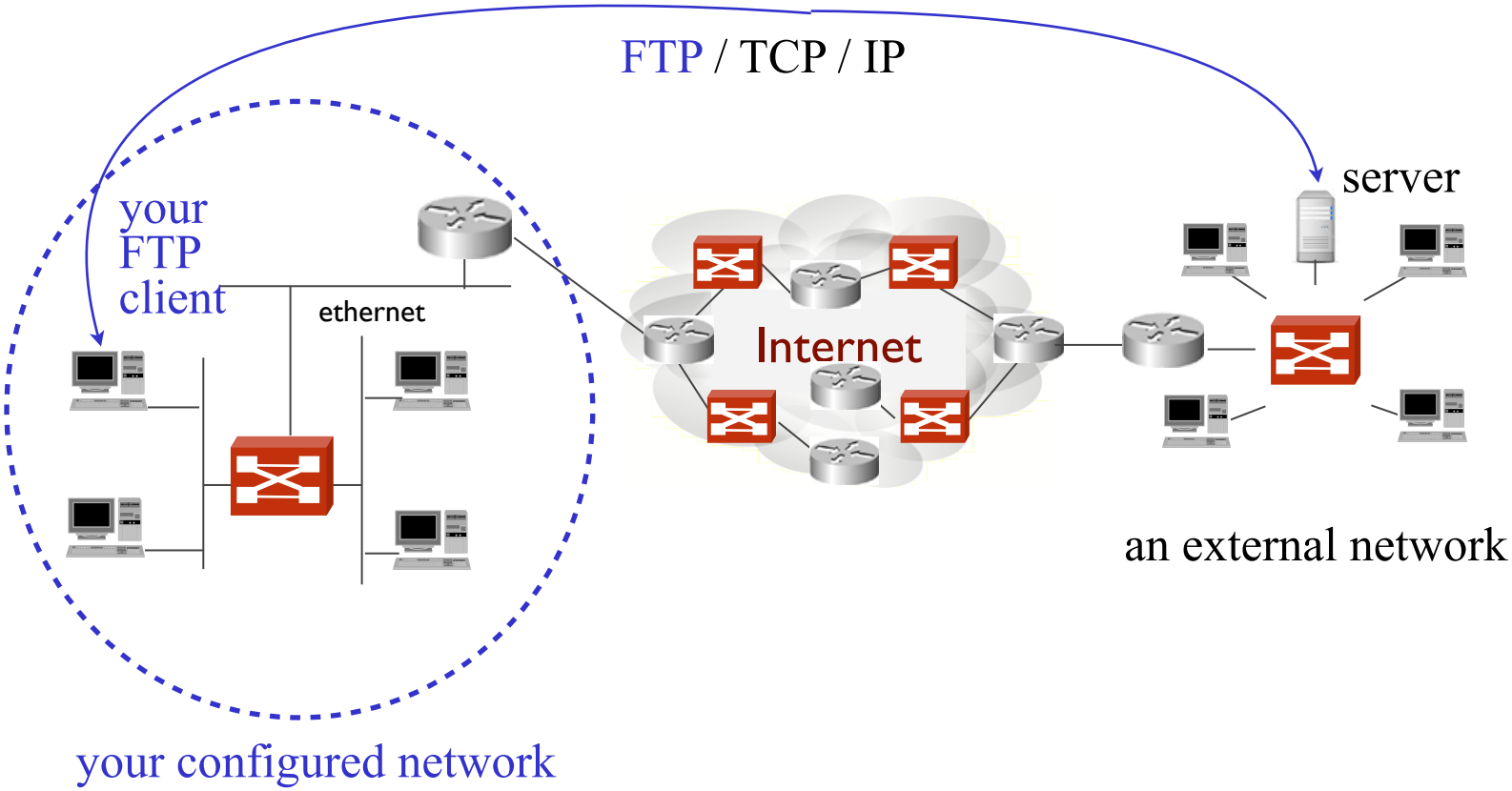
    - for switches and routers, using the console of such machines and typing the appropriate microTik commands (examples in the slides)

      - consoles are accessible using the GTKTerm application in a Linux workstation

Assignment 2 - scope and purpose

- Communication in IP shared networks
  - configuration of IP cabled networks
    - use, analyse and understand protocols to exchange information in layers 2, 3 and 4 in a reliable way
    - configure hosts and network equipment
  - development of an application layer protocol to exchange information between two remote hosts
    - FTP, File Transfer Protocol
    - client that exchanges commands and responses with a remote server to request and receive a file

Assignment 2



Assignment 2

- application to transfer a file between a server and a host connected to distinct (private) IP networks using an application layer protocol
  - 2 main components of the work

    1. Configuration of an IP-based network infrastructure
       - physical configuration using Ethernet cables and interfaces
       - logical configuration in terms of IP addresses and routes
       - done in the lab

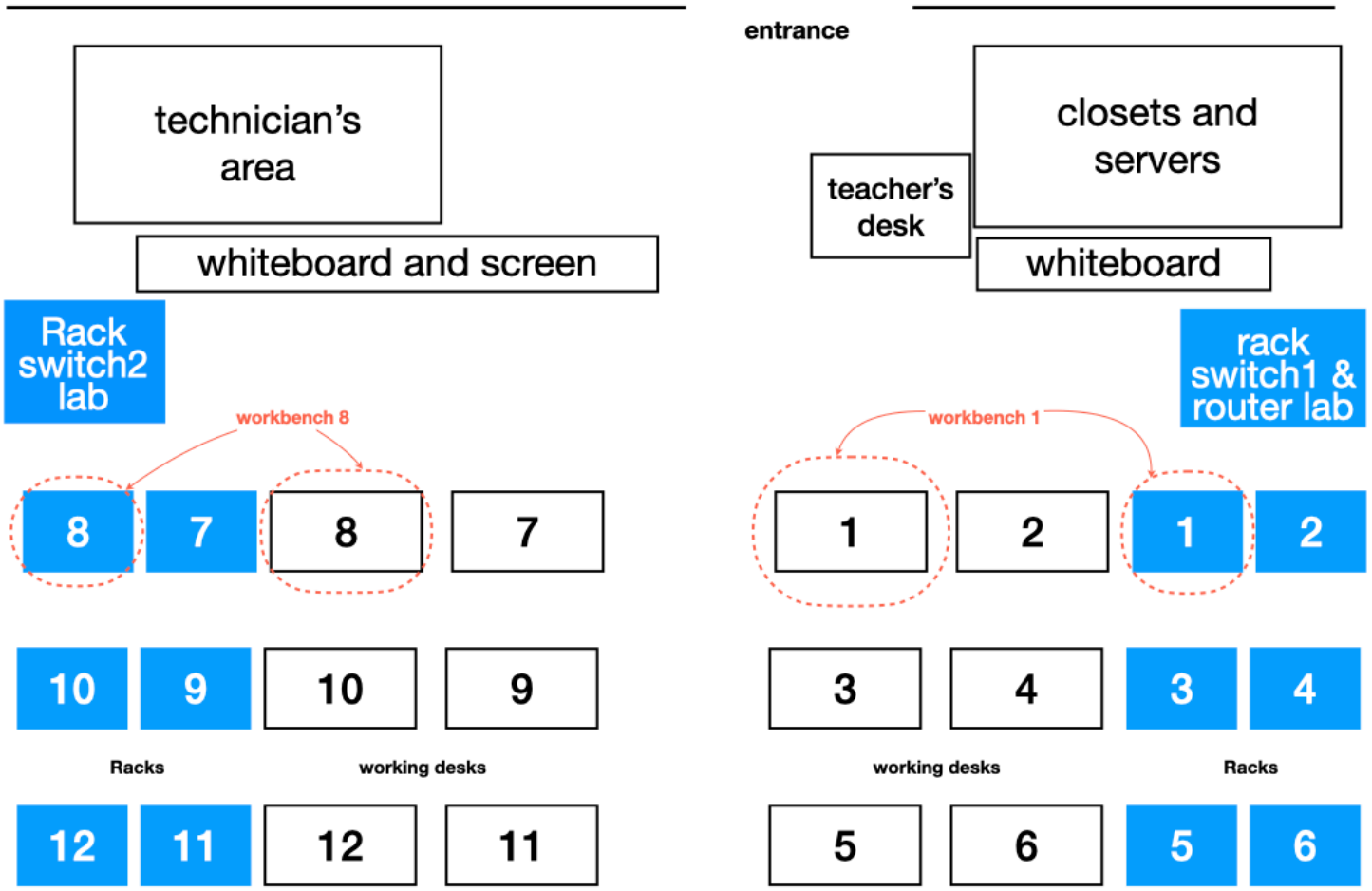
    2. Development of a software FTP client (File Transfer Protocol)
       - client running at the AL that sends commands and understands responses of the FTP protocol, enabling to request the transfer of a data file from a remote server to the local hard disk using IP communication networks
       - outside the classroom

Infrastructures & Tools

- Computer Networks laboratory
  - I.322
- 12 workbenches
  - Workbench Y numbered from 1 to 12
  - Each composed of a rack with
    - 3 Linux machines called Tux's numbered from Y1 to Y3
      - e.g., workbench 6,: Tux 61, Tux 62, Tux 63
      - each Tux has one serial port and 3 Ethernet interfaces
    - one 24-port Switch (Microtik)
    - one 10-port Router (Microtik)
    - one Linux server (Nexus, not used in this UC)

Infrastructure

- Lab I322



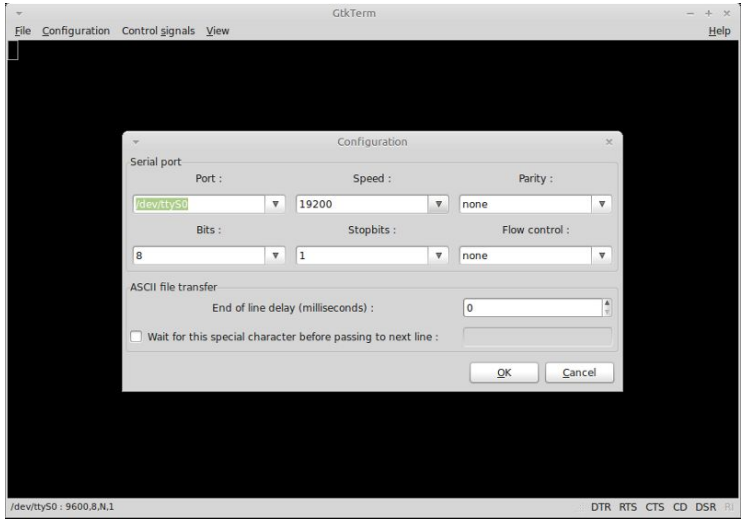
Infrastructure

Workbench rack - back

Workbench rack - front

Part B

- to configure the switch and the router
  - use the application GTKTerm
    - establishes a serial connection to the switch or to the router
    - commands typed in the GTKTerm console will be executed in the switch or in the router



Lab 2 Planning

- the two parts can be developed in parallel
  - part A outside the laboratory
  - part B needs to be done in the laboratory

week	13 Nov	20 Nov	27 Nov	4 Dec	11 Dec	12 Dec	18 Dec
part A	Experiences with telnet and app layer protocols; emphasis on the protocol FTP	✓	✓	✓	✓	submit code in Moodle	✓
part B		Experiments 1 and 2	Experiments 2 and 3	Experiments 3 and 4	Experiments 5 & 6 individual quiz in Moodle		Demonstration of work done

- submission of report
  - 23 December
  - contains direct answers to questions formulated in each experiment

Evaluation

- Organization
  - Groups of 2 students
- Evaluation criteria
  - Participation during class (continuous evaluation)
  - Individual Quiz
  - Presentation and demonstration of the work
  - Final report
- Demonstration of the work
  - Replicate the network topology described in Part 2 / Exp 6
  - Using different IP addresses for the VLANs (announced before the demonstration starts)

Part A - Development of a download application

Development of an Application

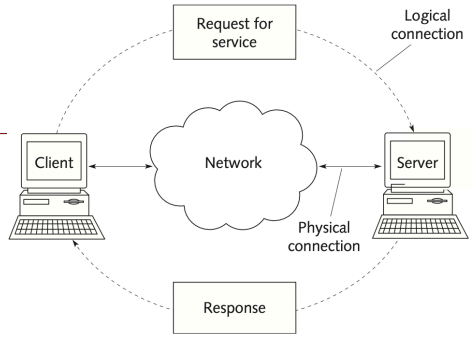
- Develop application `download ftp://ftp.netlab.fe.up.pt/pub/...`
  - Application downloads a single file
  - Implements FTP application protocol, as described in RFC959
  - Adopts URL syntax, as described in RFC1738

    `ftp://[<user>:<password>@]<host>/<url-path>`
- Learning objectives
  - Describe client - server concept and its peculiarities in TCP/IP
  - Characterise application protocols in general, characterise URL, describe in detail the behaviour of FTP
  - Locate and read RFCs
  - Implement a simple FTP client in C language
  - Use sockets and TCP in C language
  - Understand service provided DNS and use it within a client program

Development of an Application

- Steps
  - This week: Experiments using Telnet application (Telnet, SMTP, POP, HTTP and FTP); focus on FTP
  - Specification/design of a download application
    - unique use case: connect, login host, passive, get path, success (file saved in CWD) or un-success (indicating failing phase)
    - challenging programming aspects: gethostbyname, sockets, control connection, passive, data connection
  - Implement a very simple FTP client at home
    - reuse existing programs: **`clientTCP.c, getIP.c`**

Client-server model



- concept
  - the client sends messages to the server requesting service of some kind
  - the server responds with messages containing the desired information or takes other appropriate action
    - e.g., to indicate error situation or requesting more data from the client
  - the server must already be running for the client to communicate with it
  - the client must know the IP address or domain name of the server to initiate communication, as well as the port number the server is listening on

Client-server model (2)

- Clients and servers exchange messages in two ways
  - through connection-oriented or connectionless communication
- Connection-oriented communication requires a session to be established between the client and server
  - guarantees reliable, error-free delivery of messages in both directions
  - accomplished through the use of TCP
- Connectionless communication
  - messages are sent hoping that they arrive correctly
  - accomplished through the use of UDP datagrams with less overhead

Client-server model (3)

- Every network application uses sockets to communicate
  - a socket is an input/output mechanism for network messages
    - it provides a descriptor for a network connection
    - socket binding is the process to associate the protocol, the IP address, and the specific port number with a socket

Using sockets

- BSD sockets, are a standardised API for network communication
  - provide an interface for processes to communicate over a network using file descriptor-like objects
- different types of communication are supported, among which
  - Stream sockets (SOCK_STREAM)
    - Reliable, connection-oriented (e.g., TCP)
  - Datagram sockets (SOCK_DGRAM)
    - Connectionless, packet-based (e.g., UDP)

Berkeley sockets API

- socket()
  - create a socket
- bind()
  - assign an address/port to a socket
- listen()
  - set a socket to listen for connections
- accept()
  - accept incoming connections
- connect()
  - connect to a remote socket
- send()/recv()
  - end and receive data
- close()
  - close a socket

Client Workflow (TCP)

- Resolve the server address
  - getaddrinfo()
    - or gethostbyname()
- Create a socket
  - socket()
- Connect to the server
  - connect()
- Send and receive data
  - send() / recv()
- Close the connection
  - close()

General Procedure for developing the TCP client - prior work

- Consult the document IETF RFC 959
  - verify the general procedure of the protocol
  - identify the specific commands that you will need to use/ implement in your client to download a file
    - verify syntax of commands and the alternative responses (numeric codes) of the server
    - determine the actions of the client for each possible alternative

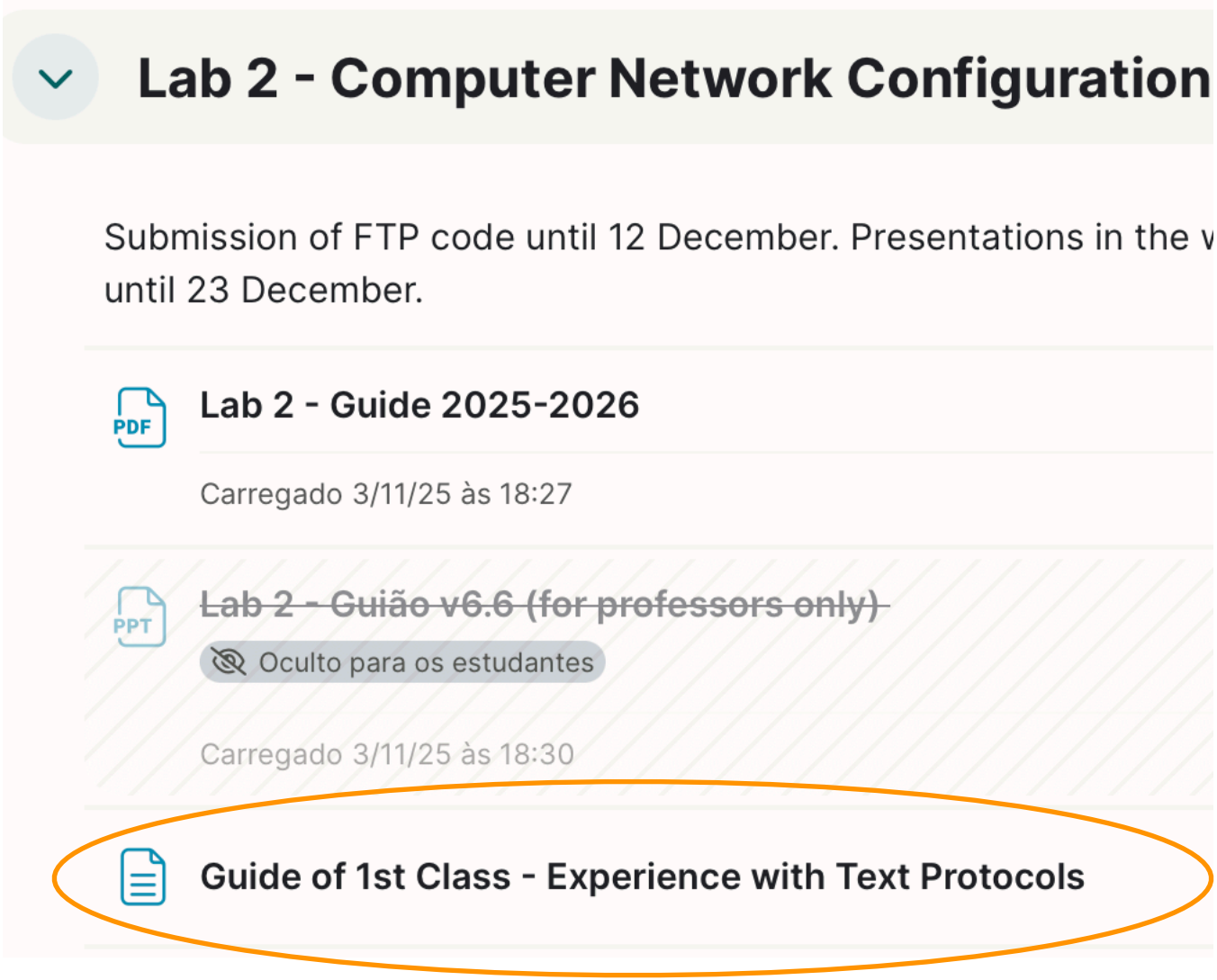
General Procedure for developing the TCP client code

- interpret the command line
  - extract: name of the server; the path; user & pass
- Resolve the server address
  - getaddrinfo() or gethostbyname()
- create socket using the resolved server address and port 21
  - socketTCP.c
- send FTP command and interpret response (3-digit code)
  - when user data needs to be transferred
    - send previously the PASV command, and parse the response to compute the new port number
    - create a new socket for data transfer

Today's class

Experiments with application layer protocols

Experiments with application layer protocols

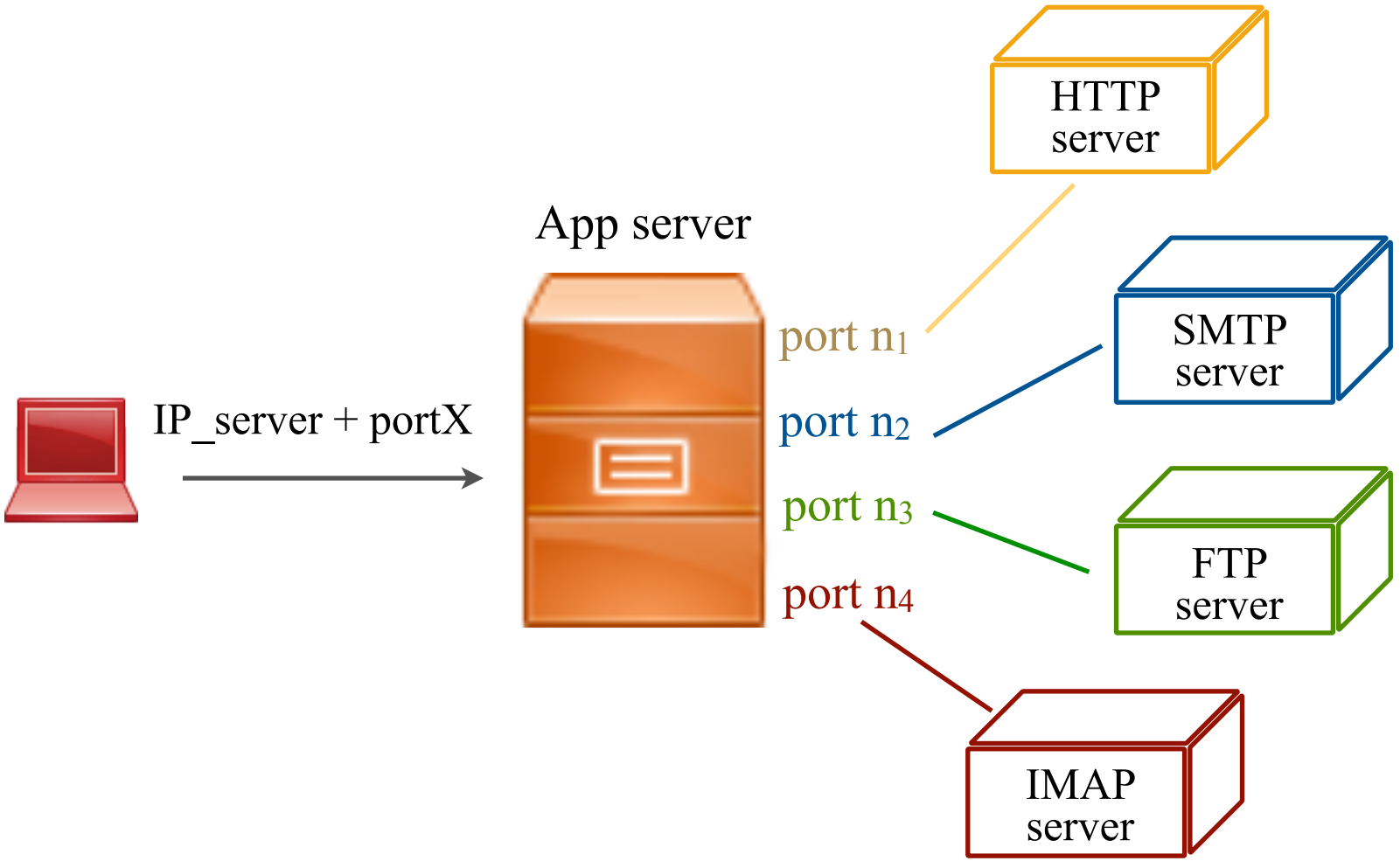


Application Layer Protocols

- client-server model
- communicate by exchanging (text) messages with well-defined syntax and semantics
  - client sends commands
  - server sends responses and data (if requested)
- client sends the messages using sockets to a given IP address and port number
- in that IP address a server is running and will check incoming messages in a given port number
- different port numbers are assigned to different types of application servers
  - in the same IP address, multiple servers may be running
    - each will check for incoming messages in a different port number

Application Layer Protocols - port numbers

- widely used services have been assigned well-known port numbers

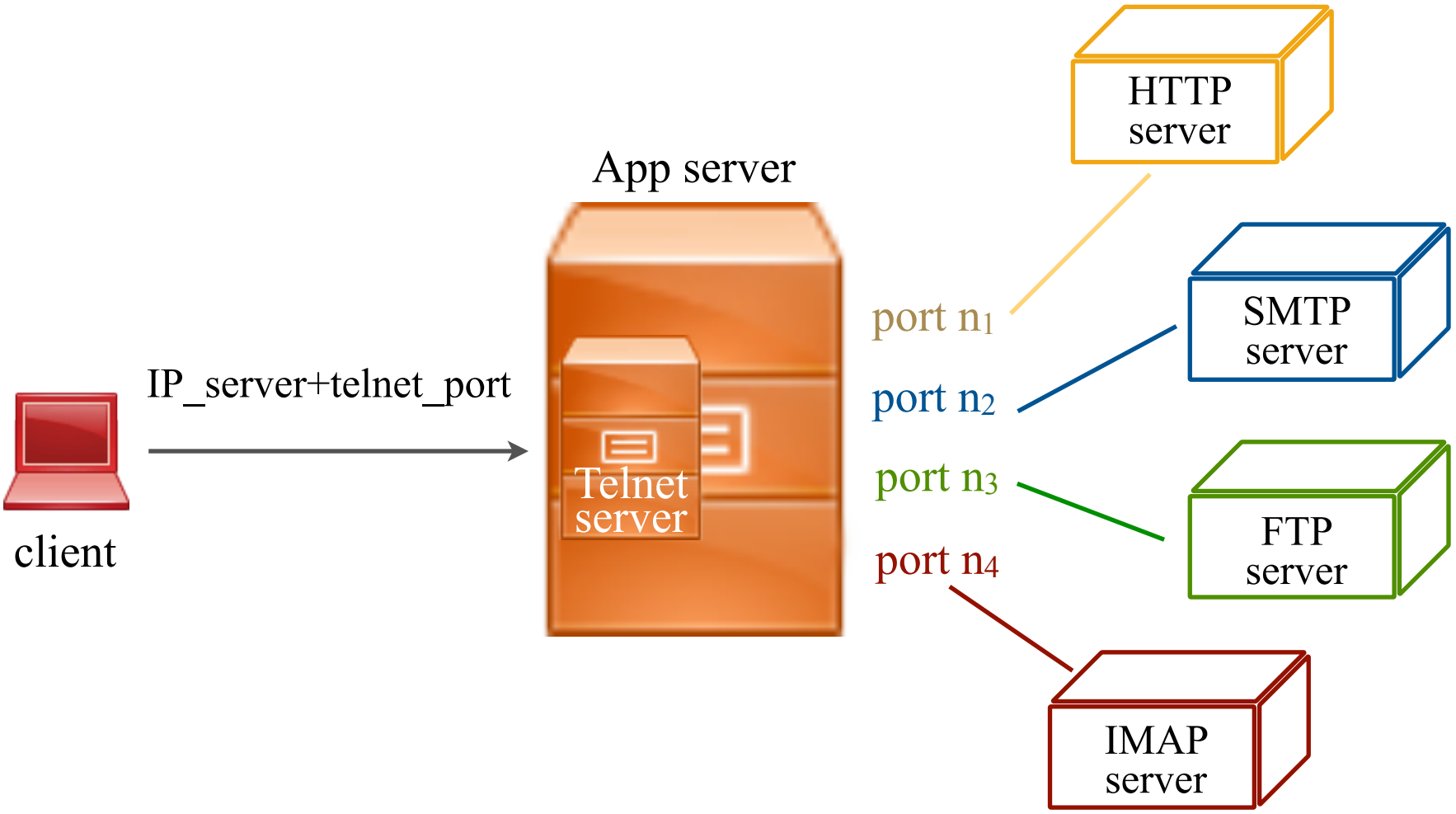


Well-known port numbers (0-1023)

port number	protocol	RFC	purpose
21	FTP	959	upload and download
22	SSH	4251-4254	secure, encrypted remote login and command execution over a network
23	TELNET	854	remote login and command execution over a network but without encryption
25	SMTP	5321	transfer and routing outgoing email between mail servers
53	DNS	1034, 1035	translating human-readable domain names into IP addresses.
80	HTTP	2616	transfer of web pages and resources between clients and web servers
143	IMAP	9051	users acces and management to email stored on a mail server from multiple devices

Application Layer Protocols - our experiments

- Telnet is a an app layer client-server protocol enabling remote login



Application Layer Protocols - our experiments (2)

- using Telnet it is possible to send commands directly to different port numbers
  - and thus, reach different app servers