Operating Systems and Networks

Lecture 01:

Introducing the course

Behzad Bordbar

Introducing the course

❑Why we created this new module

- Personal experience and trends in technology
- HoS

Important

❑First time

❑Both practical and theory

❑Requires lots of work and time… exam HARD ☹

But

❑Does not require lots of background

Introducing the course (continue)

1. Demonstrate an understanding of the **fundamental** concepts and issues involved in OS and networking of IP-based systems Interact and manage an Operating System
2. Demonstrate an understanding of the challenges involved in the design of Distributed Systems in general and main methods of addressing them
3. Explain Transport Layer protocols and their differences
4. Understand the basics and practical issues and architectures involving in important Application Layer protocols
5. Demonstrate **practical understanding** of the theoretical foundations of Operating Systems and Distributed Systems

Introducing the course (continue)

- Sessional: 1.5 hr examination (80%), continuous assessment (20%).
- Supplementary (where allowed): 1.5 hr examination only (100%).
- You will have lab hours to ask your questions from a teaching assistant (time will be announced later) .

What am I going to learn?

Mixture of theory AND practice of:

- OS Fundamentals and architecture
- Shell programming
- OS networking
- OS support for Distributed Systems
- Distributed object, RMI and RPC
- Virtualisation, Xen, KVM *
- cryptographic algorithms and Security
- P2P (*)
- Wireless protocols
- Web servers: Apache server and nginx
- subversion and git
- maven (*)
- Distributed file systems ()
- HDFS
- Web Services (*)
- NoSQL and OS (*)

But we don't teach kernel programming

Exercise 0: learn Linux

- **This course is not for you if you don't want to learn Linux**
- Mac people are OK, as have access to shell.
- Cygwin and MS powershell wont do ☹
- Is Linux a new OS to you or need to brush up?
  - SoCS machines
  - Dual boots system
  - Virtualised environment **(**install Ubuntu on **Vmware player** or **Virtual box)**

Exercise 0: Learn Basic Linux commands, for example from

http://www.debianhelp.co.uk/commands.htm

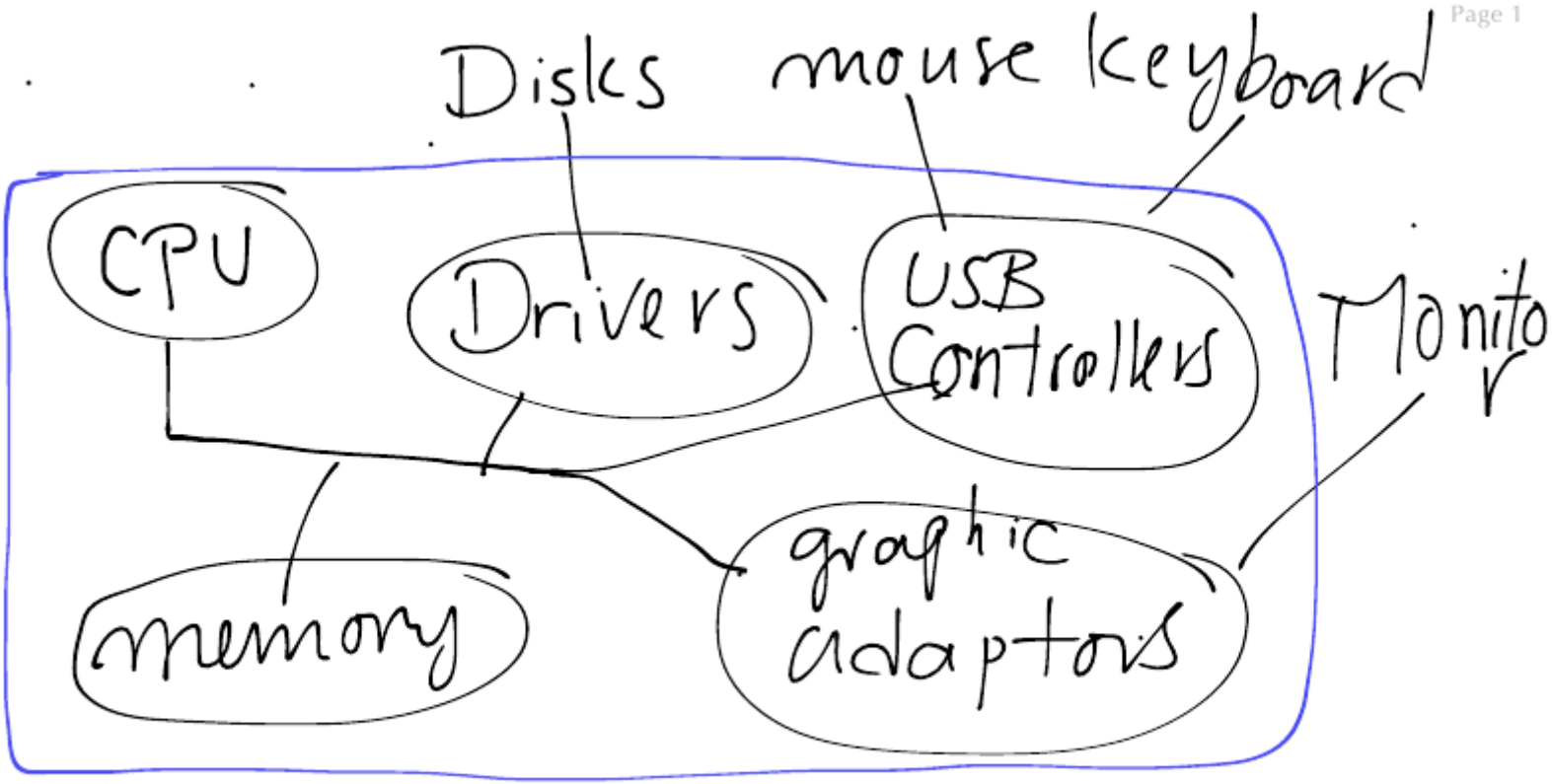
What reading material can help me?

- excellent library with lots of books on OS and Networking and Distributed systems
- Lots of online books

- Modern Operating Systems by Andrew S. Tanenbaum
- Operating System Concepts by Abraham Silberschatz, Peter B. Galvin and Greg Gagne
- Distributed Systems: Concepts and Design by George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair
- Data Communications and Networking by Behrouz Forouzan

Operating system: preliminaries

- What is in your computer?



- We want to learn these? But what is OS?

central processing unit (CPU)

- "brain" of the computer

CPU

Semiconductor companies:

- **Intel**
- PC: core i7, i5,....
- Sever and Workstation processor: Intel Xeon* processor E7, E5, ...
- Mac: Hawell, Ivy Bridge, Sandy Bridge,...

AMD (Advanced Micro Devices Ltd)

- Boldddozer (Vishera, Zambezi,...), K10 series (Athlon, Operon, ...)

All above are based on x86: **backward compatible (?)** set architecture based on Intel 8086 cpu.

AMD also produces CPU based on a blueprint developed by ARM

- Majority of mobile phones use ARM

CPU(continue)

How does it work? *Draw picture!*

- fetch the first instruction from memory,
- decode it to determine its type and operands
- execute it, and
- then fetch, decode, and execute subsequent instructions.

Instructions are CPU specific

- i7* core can not execute ARM instructions!

CPU(continue)

Accessing memory to get an instruction or data takes much longer than executing an instruction:

- CPUs contain registers inside to hold variables and temporary results.

What is instruction set?

- Load/store data from/to memory into a register
- combine two operands from registers and store result, for example adding
- …

Register

- 8-bit or 32 –bit storage. Examples of registers are:
- **General purpose** reg.: temporary data & results
- **program counter**: memory address of the next instruction to be fetched and then program counter is updated to point to its successor.
- **Stack pointer**: points to the top of the current stack in memory

Stack?

- Stack contains one frame for each procedure to be entered
- stack frame holds those input parameters, local variables, and temporary variables that are not kept in registers.

CPU modes

- At least two modes, **kernel mode** and **user mode**
- kernel mode: CPU can execute every instruction in its instruction set and use every feature of the hardware (complete access to hardware).
- When CPU in kernel mode we say OS in kernel mode.
- User programs run in user mode, which permits only a subset of the instructions to be executed and a subset of the features to be accessed. Generally, all instructions involving I/O and memory protection are disallowed in user mode.
- Program Status Word (PSW) is a register that (among other things) keeps the mode of the CPU

System call and Trap

So how do a user program do for example I/O?

- Ask OS: a user program must make **a system call**, which **traps** into the kernel and invokes the operating system.
- The TRAP instruction switches from user mode to kernel mode and starts the operating system.
- When the work has been completed, control is returned to the user program at the instruction following the system call.

We will look at this in details later! Jut one small point

Not every trap is caused by system calls!

- Some traps are caused by the hardware to warn of an exceptional situation such as an attempt to divide by 0 or an arithmetic underflow.
- Trap causes operating system gets control and must decide what to do. Example:
  - OS terminates program when error
  - error can be ignored and underflowed number set to 0
- Do you know about exception handling: that is when control is handled to program.

GPU =/=CPU

- Graphic Processing Unit (GPU) coined by Nvidia
- Called VPU (Visual Processing Unit) by ATI (a competitor of Nvidia)
- GPU: electoronic circuit specialised for graphic processing
- Presented as a video card in a PC for processing graphics
- Originally designed and used for image manipulation. Turned out to be suitable for parallel computing (most notably CUDA)

How to say what CPU you have

$ cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 58
model name : Intel(R) Core(TM) i7-3667U CPU @ 2.00GHz
stepping ….

- You see a number i7-3667U, what does it mean?

See intel page

- or run hardware lister

$ sudo lshw
$ sudo lshw |grep -i cpu

What happens when your computer starts?

Bootstrap program (bootloader, GRUB) runs first :

- It initializes all aspects of the system, from **CPU registers** to **device controllers** to **memory contents.**
- Load operating system and kernel to memory and start it.

Where is it saved if no CPU powered up?

- firmware: hardware in **read-only memory (ROM)** or **electrically erasable programmable read-only memory (EEPROM)**,
- Why in firmware? Can not be infected easily with virus

After your computer started:

- program at boot time become System Daemons/processes (in unix “init”)
- Event occur (mouse click, a program want to access a file…) we refer to these as **interrupts** from either the hardware or the software.
- Hardware may trigger an interrupt at any time by sending a signal to the CPU.
- Software may trigger an interrupt by executing a special operation called a **system call… (**we see what happens with this, how about hardware**)**

When CPU interrupted:

- it stops what it is doing and immediately transfers execution to a fixed location.
- The fixed location usually contains the starting address where the service routine for the interrupt is located.
- The interrupt service routine executes
- on completion, the CPU resumes the interrupted computation.

Memory

An array of bytes.

- read-only memory (ROM) and EEPROM
- ROM can not be modified: suitable for bootstrap program or game cartridges
- EEPROM can be modified but not frequently: smartphones have EEPROM to store their factory-installed programs.
- CPU needs read **and write**: **main memory** (also called random-access memory, or RAM).
- Main memory commonly is implemented in Dynamic Random-Access Memory (DRAM) technology.

Memory (continue)

- Register is another type of memory.

Main memory goes away when machined turned off:

- Secondary storage: magnetic disk, optical disk, tapes

Cache

- The data that has been used a lot is cached in a faster storage system.
- So if CPU looking for info, first check cache then main memoery. …

[cache is a concept more general than operating system, your browser has a cache, your dbms has a cache]

Speed of access to memory

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Picture from Dinosaur book

Summary

- Description of the module

Started by preliminaries of OS

- CPU
- Register
- System call trap and interrup
- What happens when computer starts?
- Different types of Memory and their speed

Will continue with preliminaries of OS

Operating Systems and Networks

Lecture 03:

Introduction to OS-part 2

Behzad Bordbar

Recap

Lecture 1:

- CPU: how it works, user and kernel mode, use of register, cache, …
- System call trap and interrup
- What happens when computer starts?
- Different types of Memory and their speed

Will continue with preliminaries of OS

Demo lecture

Contents

- Service view (provider of services) of the OS
- Shell
- Everything a directory
- mkdir, mv, cp,…
- Access control
- Find, grep
- |, >, >>, ; and their differences
- wget,…

Contents

- How does mouse and keyboard work?
- Device controller
- CPU multitasking
- Time sharing
- a short study of system calls
  - API

How does mouse, keyboard …work?

- We said

Hardware may trigger an interrupt at any time by sending a signal to the CPU.

- But how?
- **Devices** interact via device controller connected through a common bus to CPU. Draw picture!
- small computer-systems interface (SCSI) controller.
- SCSI controller: hardware (card or chip) that allows SCSI storage device to communicate with the operating system using a SCSI bus

Device controller

- Maintains some local buffer storage and a set of special-purpose registers.
- Device controller moves the data between the peripheral devices that it controls and its local buffer storage.
- Operating systems have a device driver for each device controller.
- you download drivers! Manually or automtaically
- Device driver understands the device controller and provides the rest of the OS with a uniform interface to the device: copy the same no matter what device

Back to, How does I/O (mouse..) work?

- Device driver loads the appropriate registers within the device controller.
- Device controller examines the contents of the registers to determine what action to take (read char from k/b)
- controller starts the transfer of data from the device to its local buffer.
- when transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation.
- Device driver then returns control to the operating system, possibly returning the data or a pointer to the data if the operation was a read or status (success…)
- Exercise: draw a sequence diagram for yourself!

How does I/O (mouse..) work? (cont..)

- This form of interrupt-driven I/O is fine for moving **small amounts** of data
- Not suitable for bulk data movement such as disk I/O.

Instead: Direct Memory Access (DMA) is used that takes CPU out of the loop.

- After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU.
- Hence: only one interrupt is generated per block, to tell the device driver that the operation has completed, rather than the one interrupt per byte generated for low-speed devices.
- CPU is made free to do other things

Multitasking in CPU

- OS picks and begins to execute one of the jobs in memory.
- Eventually job may have to wait for some task, such as an I/O operation, to complete.
- OS switches to, and executes, another job.
- When that job needs to wait, the CPU switches to another job, and so on.
- Eventually, the first job finishes waiting and gets the CPU back.

Time-sharing:

- CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it is running.

Time sharing

requires CPU scheduling of user tasks. But how?

- Each user has at least one separate program in memory.
- A program loaded into memory and executing is called a **process**. We will study this in details!
- When a process executes, it typically executes for only a short time before it either finishes or needs to perform I/O.
- I/O takes long long long time compare to execution! (look at the speed of access slides!)

Time sharing

- Time sharing: several jobs be kept simultaneously in memory.
- CPU scheduling: process of deciding which job is brought to memory to be executed, when there are not enough room.

Reasonable response time must be ensured:

1. processes are swapped in and out of main memory to the disk
2. use virtual memory: a technique that allows the execution of a process that is not completely in memory

- virtual-memory scheme enables users to run programs that are larger than actual **physical memory**. Further, it abstracts main memory into a large, uniform array of storage, separating **logical memory** as viewed by the user from physical memory. This arrangement frees programmers from concern over memory-storage limitations.

Dual mode

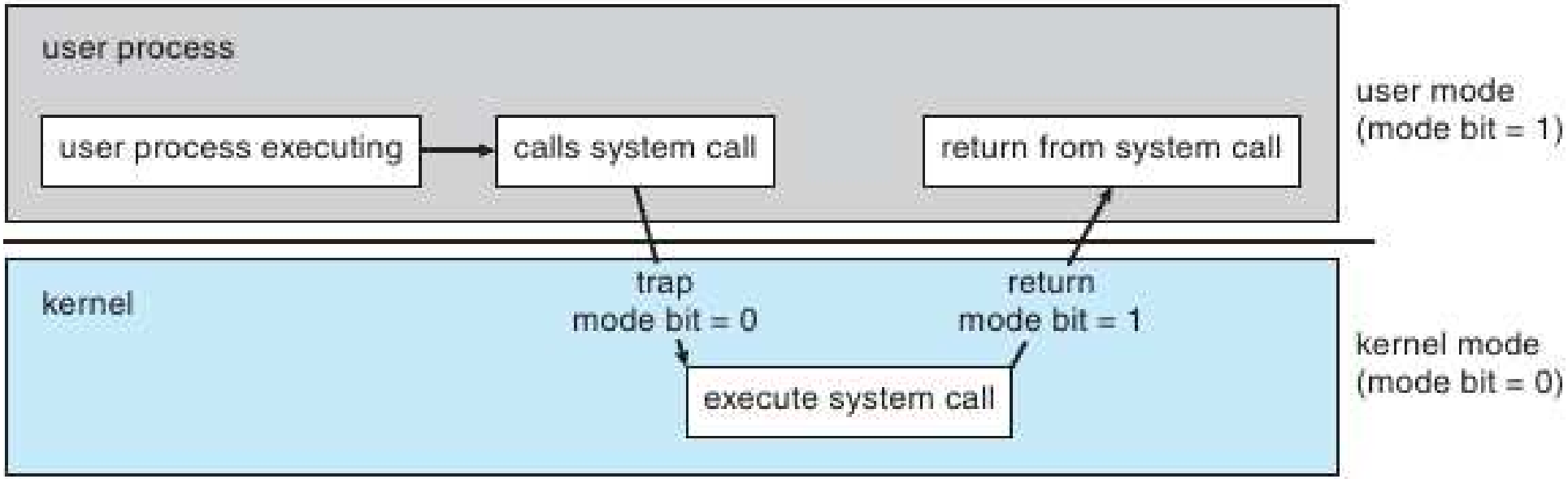


Figure from Dragon book.

- Dual mode OS protect from harm caused by privileged instructions
- Extended to multi mode by domains: Dom0, DomU

Why do we need hardware support?

- MS-DOS: Intel 8088 architecture, which has no mode bit
- user program can wipe out the whole OS
- programs are able to write to a device …

In dual mode:

- hardware detects errors that violate modes and handle them by Os
- stops user program attempts to execute an illegal instruction or to access memory of other users

When error detected

- OS must terminate the program
- OS gives error message
- produces memory dumps by writing to a file (users can check or OS vendors can check (Sun)).

system calls

provide an interface to the services made available by an operating system.

What language: are System-call written in?

- typically C and C++ and sometimes assembly-language involved
- explain the system calls for reading data from one file and writing to another file:

$cp file1 file2

open file1, possible error(print, abort), create file2 (file2 exists, rewrite/rename…), start read and write (errors:disk space, memory stick unplugged…), all read and written, close files, ack

Do I access system call directly?

system calls: API to wrap system calls

Application Programming Interface (API)

- specifies a set of functions that are available to an application programmer, including the parameters that are passed to each function and the return values the programmer can expect.
- programmer accesses an API via a library of code provided by the operating system.

Example of APIs:

1. Windows API for Windows systems

Example: CreateProcess() which invokes the NTCreateProcess() system call in the Windows kernel return value 0 or 1 (error)

system calls: API (continue)

Example of APIs:

2. POSIX API for POSIX-based systems (UNIX, Linux, and Mac OS X)

- programmer accesses an API via a library of code provided by the operating system.

Example: read

input:

- int fd: file descriptor to be read
- void *buf: pointer into buffer to be read into
- size_t count: maximum number of bytes to read

output:

- number of bytes read (if success)
- -1 if fail
- UNIX and Linux for programs written in the C language, the library is called libc.

system calls: API (continue)

Example of APIs:

3. Java API for programs that run on the Java virtual machine.

getParentFile()

invoked on a file object.

output:

Returns the abstract pathname of this abstract pathname's parent, or null if this pathname does not name a parent directory.

JVM uses the OS system calls.

why do we use API?

Why not invoking actual system calls directly?

- Program portability: program can compile and run on any system that supports the API
- system calls can often be more detailed and difficult to work with
- give access to high level objects (java API)

Do you know interfaces in java? using system calls is like implementing an (or many) interfaces

What happens when a user prog. makes a system call

- caller only needs to know the signature!
- method call and parameters are passed into a registers
- values saved in memory for example on table or stacks but addresses in registers
- Stack is preferred because do not put limit on the number of parameters stored.

summary

- Study of I/O devices
- Device Controller
- How OS manages multiple tasks
- System call and their use in user programs
- API and examples of API
- Similarity between API and interface

Operating Systems and Networks

Lecture 05:

Introduction to OS-part 4

Behzad Bordbar

Recap

- Different types of system call
  - process control
  - file manipulation
  - device manipulation
  - information maintenance
  - communications
  - protection.
- process
- process =/= program (and =/= thread)

Contents

- Heap vs stack (what size is a proc stack?)
- proc. states and control block
- context switching
- process and thread
- process in linux

From C to Assembly

- http://gcc.godbolt.org/
- What does this piece of C code do?

```
#include <stdio.h>
 #define LAST 10
   int main()
   {
      int i, sum = 0;
      for ( i = 1; i <= LAST; i++ ) {
        sum += i;
      }
      printf("sum = %d\n", sum);
      return 0;
   }
```

From C to Assembly (continue)

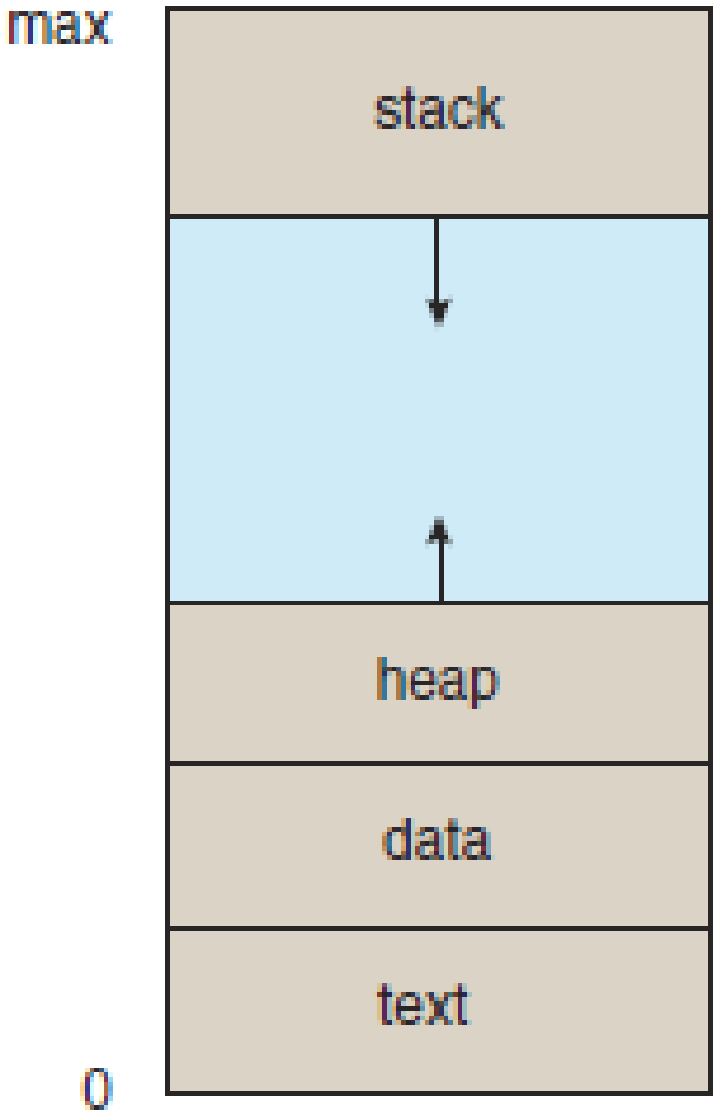
- Explore assembly output with different compilers (remove – O2 from Compiler options and activate Colourise)
- process a program which is running.
- What do we need for that?
- observe: push, pop, mov…

stack and heap

- Stack is a contiguous block of memory that is allocated to each process.
- Stack is LIFO (last in first out).
- Stack has stack frame: contains parameter to functions, local variables and data for recovering previous stack frame.
- When a function is called frame for that function is pushed into stack, when done executing we pop. When we call stack grow, when we execute stack shrinks and we end up with the caller and then frame pops.

heap and stack

- stack frames



picture from dragonbook

What is difference between heap and stack?

stack

- in java we have a data type jave.util.stack (not here!)
- part of computer memory that stores temporary variables created by each function (including the main() function). T
- FILO data structure managed and optimized by the CPU
- push and pop of stack frames
- Once a stack variable is freed, that region of memory becomes available for other stack variables.

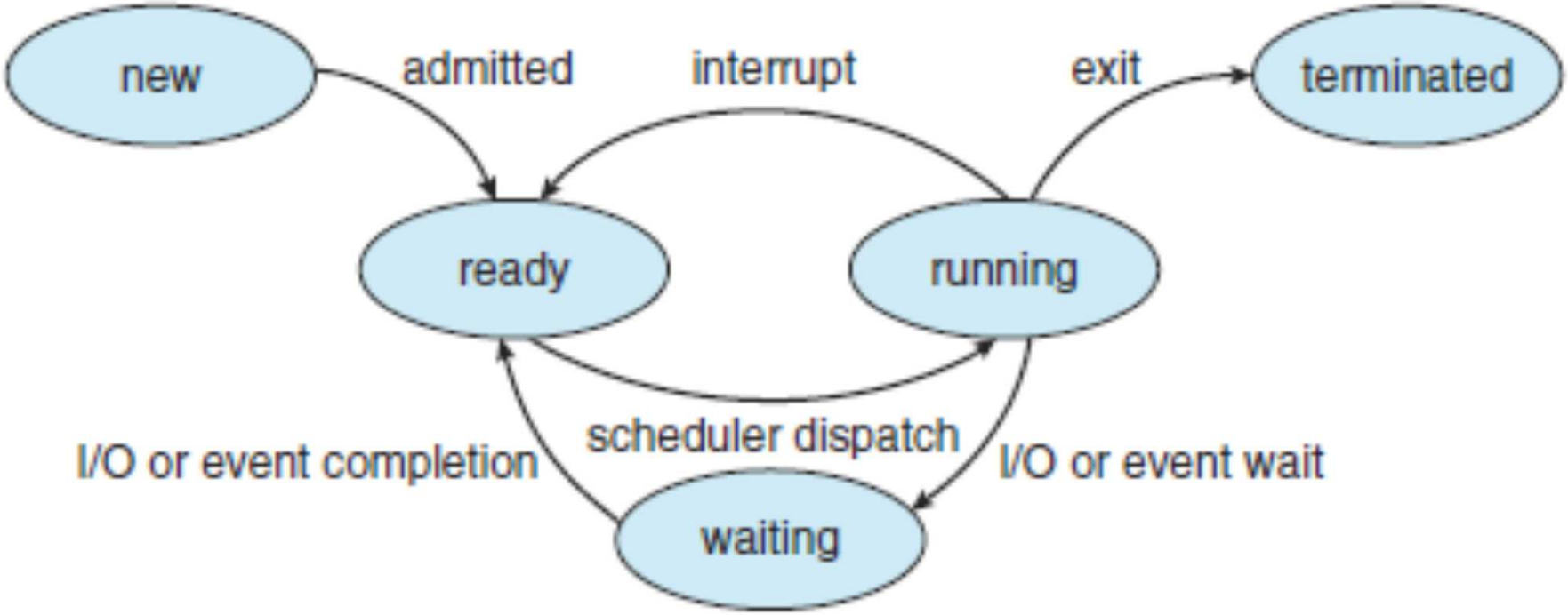
stack

- advantage: that memory is managed for you [no need to allocate memory yourself or free it once you don't need it any more]
- CPU organizes stack memory so efficiently (very fast).
- stack is limited
- $pgrep chrome
- $ cat /proc/<PID>/limits

heap

- part of computer memory that is not managed automatically.
- in C you must use malloc() or calloc() to allocate
- you are responsible for using free() to de-allocate that memory
- if you don’t: memory leak (memory on the heap will still be set aside (and won't be available to other processes)
- Unlike the stack, the heap does not have size restrictions on variable size (apart from the physical limitations of your machine).
- Heap memory is slightly slower to be read from and written to, because one has to use pointers to access memory on the heap.
- Heap variables are essentially global in scope

process state



picture from dragonbook

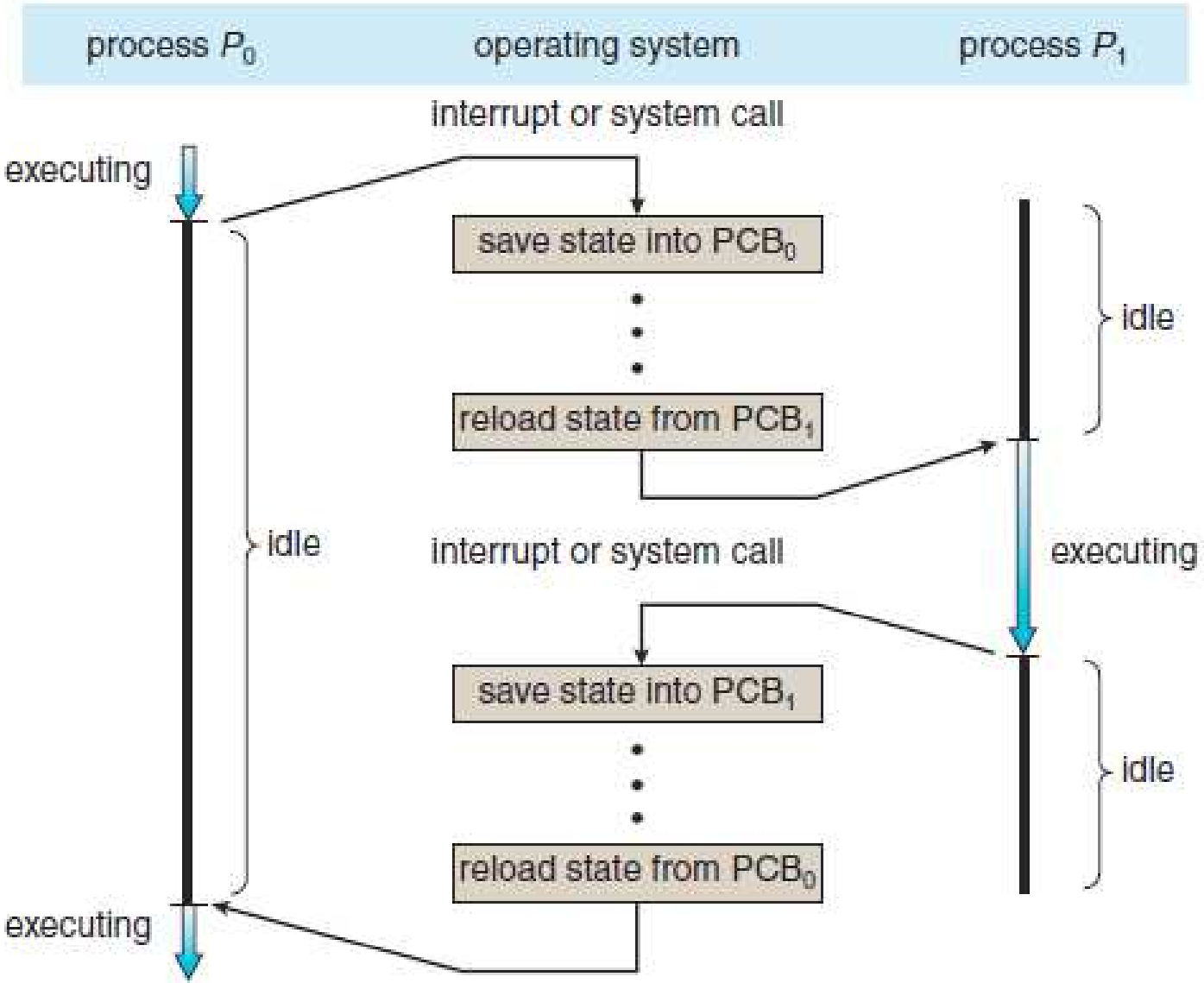
- only one process can be running on any processor at any instant.
- Many processes may be ready and waiting,

process control block (PCB)

information required for context switching

- Process state
- Program counter.
- CPU registers
- CPU-scheduling information
- Memory-management information
- Accounting information
- I/O status information

How does CPU execute multiple processes?



dragonbook. so… how do process and program compare?

Multiprogramming (revisited)

- Maximising CPU utilisation: some process running at all time
- time sharing: is to switch the CPU among processes so frequently that users can interact with each program while it is running
- Only one process can run at a time, the rest of processes must wait

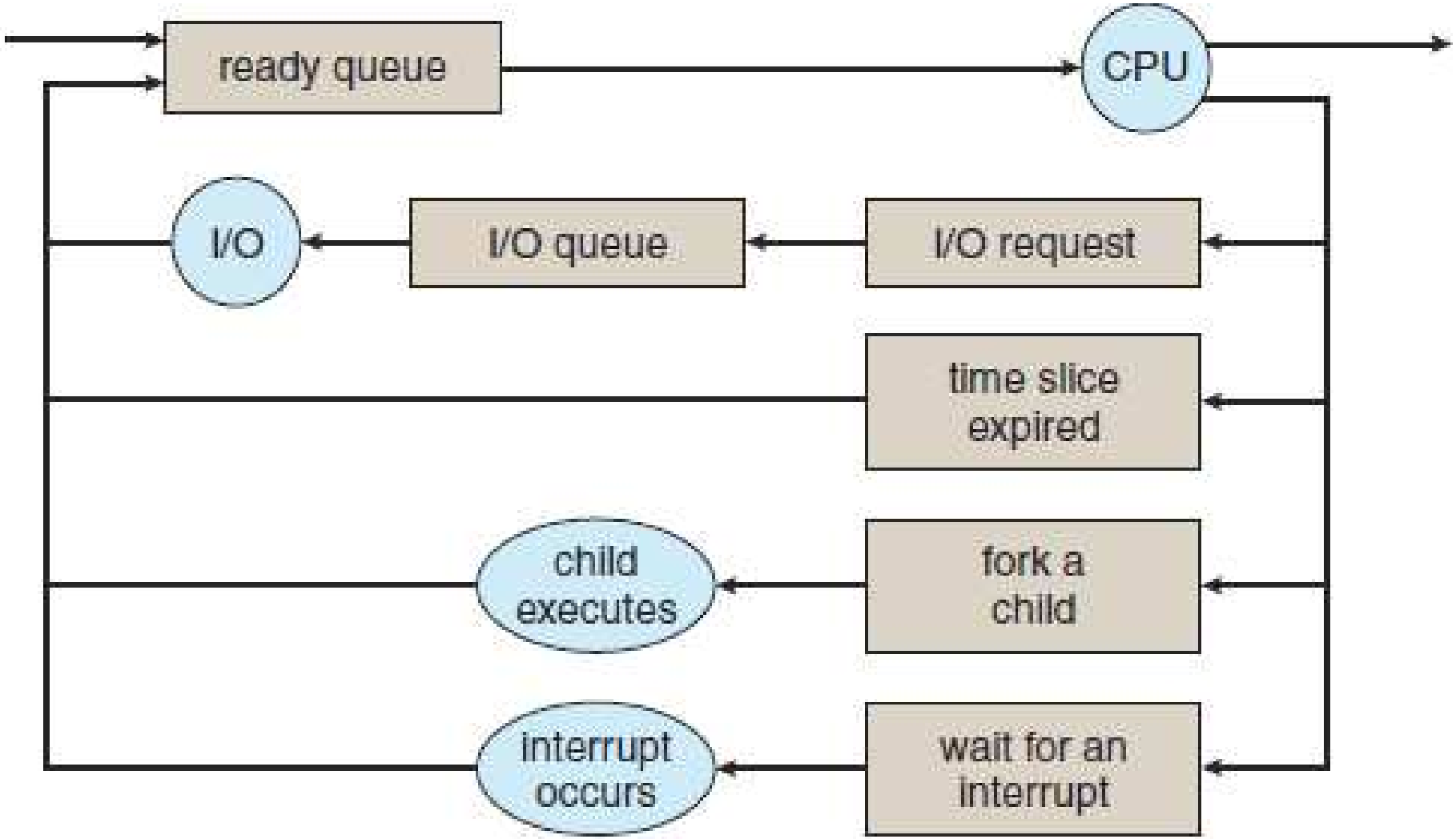
Process Scheduler:

- selects an available process for program execution on the CPU.

Queues

- job queue: all processes in the system
- ready queue: processes that are residing in main memory and are ready and waiting to execute are kept on a list (linked list)
- PCBs that have a link to the next PCB
- Device queue: list of processes waiting for a particular I/O device is called a

Queuing Diagram



source Dragon book

Context switching

- interrupt >> a state save of the current state of the CPU and then a state restore to resume operations
- kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run.
- Context-switch time is pure overhead.
- Switching speed varies from machine to machine (few milliseconds)

Process and Thread

- McDonald on high street: thread and process
- Example: a document with 1) editor 2) spell checker
- fork in unix (two process)
- can we share the context… thread
- Suitable particularly for multicore

Process in Linux: called task

- processes form a tree $ps -el
- unique id number (pid integer)
- The init process (pid of 1) is the root parent and spwans other proceses
- kthreadd: creating additional processes that perform tasks on behalf of the kernel

- ps -ef |more
- ps -efH |more
- pstree

Process in Linux

- fork() system call: makes new process
- New process has a copy of the address of the original process. Why?
- better communication between father & child
- father continues executing and child calls exec() sys. call and replaces memory space with new prog.
- Draw picture + example of shell and editor
- both proc. can create more children or
- father invokes wait() sys. call to move to ready queue until the termination of the child (background proc. &)
- termination, cascading termination, zombie process

Process in Linux

- termination: A process terminates when it finishes executing its final statement and asks the operating system to delete it by using the exit() system call.

Then what happens:

- return status value (typically an integer) to its parent process (via the wait() system call)
- All the resources of the process—including physical and virtual memory, open files, and I/O buffers—are deallocated by the operating system.

A process can be terminated directly by other process

- cascading termination: if parents terminate all children terminated (show via shell and background process)

Process in Linux (continue)

- What is exit status?
- wait() system call by parents
- a pointer is passed to return exit status of child
- zombie process
- orphan process
- terminating of orphan process by init()

summary

- Heap vs stack (what size is a proc stack?)
- proc states and control block
- context switching
- process and thread
- process in linux

```
----Lecture 06

Importance of playing

--command line color
bash Bourne again shell
bashrc (resource control)
Talk about .bashrc and how you can make it color of the prompt change
.bash_aliases

--echo command print to terminal and move to next line
echo “Hello World”
--variables
test=”Hello World”
echo $test
what happens if I type
echo test

read puts into a variable
read yourname  # no space
# for comments

-- Exercise 6.1: read a name and echo it
read yourname
echo $yourname

-- How to do I prompt for  my name and read it and type it?
echo "what is your name?"; read yourname ; echo "you typed:"$yourname

This is lame!?%@@E
-- make sh file with the following contents
echo "what is your name?"
read yourname
echo "you typed:"$yourname

execute it
bash sth.sh
bash -xu sth.sh  print traces of execution.

shebang
#! /bin/bash
# print your name

--- Exercise6.2:
Write a script to ask your name and your family name and prints both.

-- further use of echo
We use pgrep to find process id number and then look for limits can you do both together
 $ pgrep chrome
 $cat /proc/3158/limits
Make a guess
$cat /proc/$(pgrep chrome)/limits
Would {} or [] work instead of ()

--  use of grave accent (push twice)
another way is to use ` ` as it expands comi
mands
$ cat /proc/`pgrep chrome`/limits
similarly
 2046  echo `pwd`
 2047  echo $(pwd)


--  difference between “” '' without
obviously cant use ! Or ;  They have specail meanings
echo Another Brick In The Wall
echo "Another Brick In The Wall"
echo 'Another Brick In The Wall'
What is I add ! Or ;
it can think it s a command use “” and ''

-- Double quote allows expansion of variables and commands and single quote does not
```

```
Try
 test="Hello World"
 echo test
 echo $test
--but ' ' does not open
 echo '$test'

Double Quotes
Use when you want to enclose variables or use shell expansion inside a string.
All characters within are interpreted as regular characters except for $ or ` which will be expanded
on the shell.
Single Quotes
All characters within single quotes are interpreted as a string character.


--  ` is something different does not
 echo `hello world I am here`



-- Integer arithmetic via expr
 icy=53
 others=23
 echo `expr $icy + $others `
 echo `expr $icy - $others `
 echo `expr $icy \* $others `

echo `expr $icy / $others `
echo `expr $icy % $others `


--Excersie 6.3:
Modify the program to read the icy and others from the command line and print the values properly

-- Exercise 6.4
Write a program to ask for date of birth of people and calculate how many days they lived. Year=365
days Months are 30 days and todays date is hardcoded.

– Shell awareness
Change the spaces and see what happens. Be careful with spaces.
Paranthesis are with \( \)

----- you can also do $((
icy=23
others=24
echo $((icy + others))
echo $((icy+others))
echo $((icy*others))

-- Exercise 6.5: write the above arithmetic with $((

---For floating point arithmetic
you need to use other methods  bc, we will see that later
priceofapple=1.2
priceoforange=2.5
c=`echo $priceofapple + $priceoforange |bc`
echo $c

-- tables
Speaking of the price I like to make s shopping list
$ echo -e "no \t item \t price \n 1 \t orange \t 1.1 \n 2 \t apple \t 1.50 \n 3 \t banana \t 2.1 "

no       item    price
 1       orange          1.1
 2       apple   1.50
 3       banana          2.1

Not really good spaces are not OK let us print it properly

-- use printf
left alignment of text
f for floating point
```

```
printf "my weakly shopping list:\n"
printf "%-5s %-15s %-6s\n" no  item  price
printf "%-5s %-15s %-3.2f\n" 1 apple 1.31
printf "%-5s %-15s %-3.2f\n" 2 orange  1.61
printf "%-5s %-15s %-3.2f\n" 3 banana 2.35
printf "%-5s %-15s %-3.2f\n" 4 sthstrange 43.5446

-- Exercise 6.6:  print your next week diary

-- Exercise 6.7:
1) remind yourself what commands touch and >> mean
2) Write a shel script which
i)  asks for a file name, say for example user types foo
ii) creates a txt file name with the name foo (this would be making a file called foo.txt)
ii) adds a first line of the following form
This file is: foo.txt
```

Operating Systems and Networks

Lecture 07:

Introduction to OS-part 5

Behzad Bordbar

Recap

- ❑General recap of all we have learnt!
- ❑CPU, how computer starts? kernel/user mod, system calls, mutlitasking

Last week

- ❑Heap vs stack (what size is a proc stack?)
- ❑proc states and control block
- ❑context switching (just overhead ☹)
- ❑process and thread
- ❑process in linux

Contents

- ❑why do we need threads?
- ❑What is a thread?
- ❑What is multicore(multiprocess)?
- ❑How does it fit into the story.
- ❑Is more core ALWAYS better?
- ❑How are threads implemented?
- ❑End... move to networking

What is a thread?

- program=/=process =/= thread
- consider client's accessing a server. Design a model for interaction?

Modern OS are multi-threaded, multiple threads operate in the kernel, and each thread performs a specific task:

- managing devices
- managing memory
- interrupt handling.

What is a thread? (continue)

- program=/=process =/= thread
- Have you written a multi-threaded program?

main() + gc …

- Garbage collection!
- if GUI many more

thread is a basic unit of CPU utilization

Single threaded process vs. multi-threaded process.

- Why not multiple processes?

Data can be shared, but execution separated!

What is a thread? (continue)

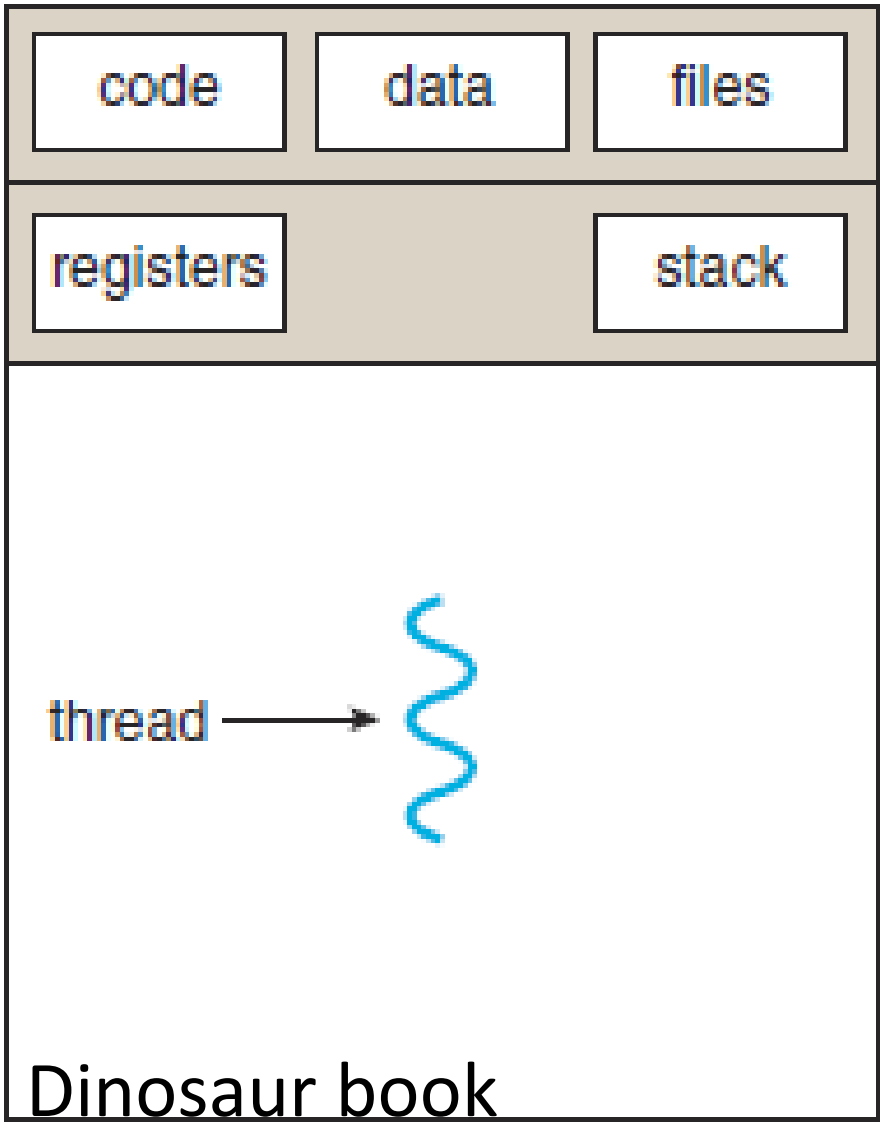
Similar to process, a thread has

- thread ID,
- program counter
- register set
- stack

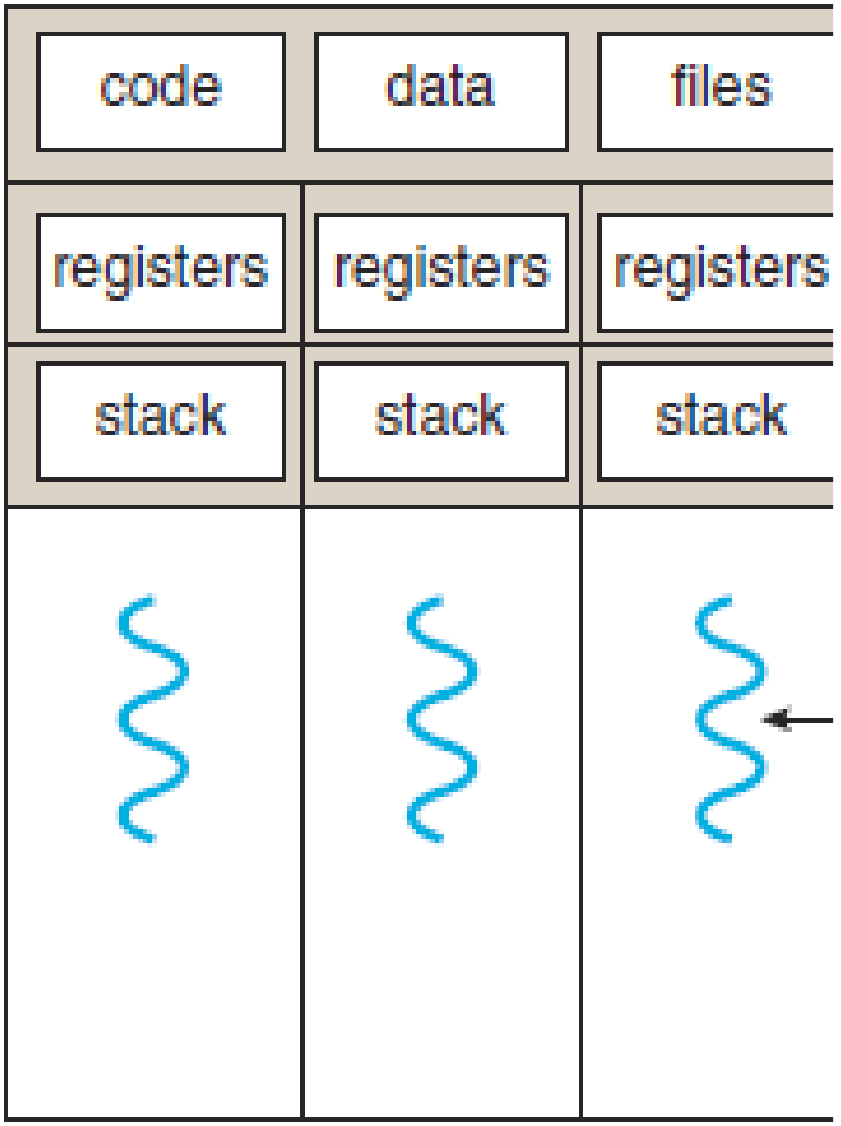
threads belonging to same process share

- code section
- data section
- OS resources, such as open files

single threaded vs multithreaded



Dinosaur book



How to see threads in Linux

- ps -e -T | grep firefox
- -e all processes
- -T all threads
- Exercise: find out threads for a number of well know processes
- *How many threads are running on your machine?*

Why threads?

- Responsiveness

a time consuming operation or lengthy process not blocking the whole process

Single threaded GUI may block the usage

- Resource sharing

Processes: shared memory and message passing (program writes code for them)

threads share the memory and the resources of the process to which they belong by default.

Why threads? (continue)

- Economy.

Allocating memory and resources for process and context switching is computationally costly

threads share the resources of the process; more economical to create and context-switch threads.

[in some cases creating a process is about thirty times costlier and switching context is five times slower]

- Scalability

multicore allows shared processing, so multi-threading is much faster than muli-processing

multi process, multicore and all that

- multiprocess: many CPU chip

Symmetric and asymmetric design

- multicore: single CPU chip has multiple computing core (register, cache…)
- faster communication (as no inter process communication)
- significantly less energy consumed

Be aware: people use two phrases interchangeably!

- blade server (data centre and Cloud):

processor process, I/O boards, and networking cards are placed in the same chassis

multicore

- single core one thread executing at at time, two cores two threads…
- Single core illusion of parallelism (by fast switching) , multiple core true parallelism
- in single core task run concurrently not parallel

Amdahl's law

- if I add core will I always make execution faster?

speed up $\leq \frac{1}{S+\frac{1-s}{N}}$

- s percentage of portion that are serial
- N number of processing cores

what happens if N becomes large (N $\rightarrow \infty$)….

throwing in more core is not going to solve the problem always!

You may need to change the program!

Threads an operating system

- user level threads
- kernel level threads (managed by kernel support)

What is the relationship between the two groups:

- many-to-one model
- one-to-one model
- many-to-manymodel.

many to one

- multiple user-level threads to one kernel thread
- Thread management is done in user space so it is efficient

Not used widely any more:

Only one thread can access kernel at a time

- All involving process block if a thread makes a blocking system call
- multiple threads are unable to run in parallel on multicore systems.

one-to-one

- linux and window use this
- Each user thread is mapped to a kernel thread
- When a thread makes blocking system call, another thread can run.
- multiple threads can run in multiprocessors.
- But resource hungry and burden on performance
- Upper bound on the number of threads
- What is the maximum number of threads allowed on my machine?

cat /proc/sys/kernel/threads-max

many-to-many model

- many user-level threads are handled by multiple kernel threads.
- Developer can create as many user thread
- kernel can schedule one thread create maximum concurrent user threads is bounded by number of kernel threads
- when threads run in parallel on a multiprocessor there is advantage
- when one thread performs a blocking system call, the kernel can schedule another thread for execution.

How to program threads?

thread library: API for creating and managing threads.

1. A library entirely in user space with no kernel support i.e. a local function call in user space and not a system call.
2. a kernel-level library supported directly by the operating system, i.e. code and data structures for in kernel space.

❑Invoking a function in the API for the library typically results in a system call to the kernel.

How to program threads? (continue)

main thread libraries:

1. Windows (uses kernel level library on Windows)
2. POSIX Pthreads (both user and kernel level)

posix? [

(Portable Operating System Interface)

family of standards by IEEE, ensure compatibility

Unix like, but microsoft supports some parts, why?]

cygwin posix compliant

3. Java threads: implemented using a thread library available on the host system[windows or pthread]

- Java threads are object: implement runnable or extend thread…

communication and networking

- End of preliminaries of OS

Be aware:

- lots left to learn
- similarities/differences between OS
- some topics important and we did not study: (memory access, registry/hive,…)
- Communicating processes (on a machine and across)
- You know one method for processes to communicate???

pipe |

- command1 |command2 (both in window and linux | dir |more)
- pipes allow to process to communicate
- but how?
- a temporary file is generated on disk
- command1 writes into it and command2 reads?
- but how?
- standard input, standard output and standard error. (next lecture)
- ordinary pipe (anonymous pipe in Windows)
- named pipe (mkfifo) we dont study this.
-

Summary

- Motivated and studied the reason for threads
- threads are units of computation
- multicore is better for threads
- sequential part of core dictates how many core can be useful… need to change code to benefit from manycore
- everything in linux is treated as files even pipe
- further mystery!

----Lecture 08

....
-- ` is something different does not
echo `hello world I am here`

-- Integer arithmetic via expr
icy=53
others=23
echo `expr $icy + $others `
echo `expr $icy - $others `
echo `expr $icy \* $others `

echo `expr $icy / $others `
echo `expr $icy % $others `

--Excersie 6.3:
Modify the program to read the icy and others from the command line and print the values properly

-- Exercise 6.4
Write a program to ask for date of birth of people and calculate how many days they lived.
Year=365 days Months are 30 days and todays date is hardcoded.
---- start of lecture 8

– Shell awareness
Change the spaces and see what happens. Be careful with spaces.
Paranthesis are with \(\)

----- you can also do $((
icy=23
others=24
echo $((icy + others))
echo $((icy+others))
echo $((icy*others))

-- Exercise 8.1 : write the above arithmetic with $((

------- $(( )) is like let
no1=3
no2=7
result=$((no1+no2))
--what will i get if
echo result
-- what about?
echo $result
so $(()) is like let

you can use let

```
 2016  let result1=no1+no2
 2017  echo $result
```
- you can do more increment
```
 2024  let result1+=7
 2025  echo $result1
```

---For floating point arithmetic
you need to use other methods bc, we will see that later
```
priceofapple=1.2
priceoforange=2.5
c=`echo $priceofapple + $priceoforange |bc`
echo $c
```

---- let us make some files and learn about parameters
- How to pass parameters
- see
l08_01_input_oneparameters.sh
l08_02_input_twoparameters.sh

-what does l08-03... does?

Exercise 8.2: modify that so that when you make an sh file it add shebang at the top.

-- tables
Speaking of the price I like to make s shopping list
```
$ echo -e "no \t item \t price \n 1 \t orange \t 1.1 \n 2 \t apple \t 1.50 \n 3 \t banana \t 2.1 "
```

```
no      item    price
 1      orange          1.1
 2      apple   1.50
 3      banana          2.1
```

Not really good spaces are not OK let us print it properly

-- use printf
left alignment of text
f for floating point
```
printf "my weakly shopping list:\n"
printf "%-5s %-15s %-6s\n" no  item  price
printf "%-5s %-15s %-3.2f\n" 1 apple 1.31
printf "%-5s %-15s %-3.2f\n" 2 orange  1.61
printf "%-5s %-15s %-3.2f\n" 3 banana 2.35
printf "%-5s %-15s %-3.2f\n" 4 sthstrange 43.5446
```

-- Exercise 8.3: print your next week diary

--- gently into for loop
- what does this do ?
```
for file in *; do echo ${file}; done
```

-- create a directory with two files
cd test
touch a
touch b
ls
- what does this do
for file in *; do `mv ${file} ${file}.txt`; done
ls

Exercise 8.4: Write a small script that reads all the files in a directory and copy them to a file called MyFiles in a directory one level above. (Hint: one level above is .. and remember >>)

Network and protocols

Lecture 09:

Operating Systems and Networks

Behzad Bordbar

recap

- Finished preliminaries of OS
- CPU, Registers, System calls, traps and interrupts
- What happens when computer starts?
- Device controllers
- CPU mutlitasking and Time sharing
- program, process, stack, heap
- process Control, context switching
- threads
- process and thread in linux
- End part one... move to networking

Contents

- shared object and pipe
- standard input, output and error
- processes communicating
- need network to communicate
- Different type of network
- Modes of transmission
- protocols
- OSI view

pipe |

- How do processes communicate?
- command1 |command2 (both in window and linux | dir |more)
- pipes allow to process to communicate
- but how?
- a temporary file is generated on disk
- command1 writes into it and command2 reads?
- but how?
- standard input, standard output and standard error. (next lecture)
- ordinary pipe (anonymous pipe in Windows)
- named pipe (mkfifo) we dont study this.

standard input, output and error

starting a shell result in creation of three files

- stdin: file that the process gets its input (e.g. from keyboard)
- stdout: file that the process puts its output (e.g. to monitor)
- stderr: file that the error goes to (e.g. to monitor)

standard input, output and error

- make a directory and change to it

$cd a

$ mkdir mydir

- The following gives one stdout and stderr to terminal

$(ls -ld mydir ; ls -ld mydir1)

- put stdout to temp1

$(ls -ld mydir ; ls -ld mydir1) > temp1

- see it

$ cat temp1

put second output to another file

- (ls -ld mydir ; ls -ld mydir1) 2> temp2

see it

- $ cat temp2

standard input, output and error

put steout to temp3 and the second output (stderr) to the same place as 1st output (i e temp3)

- $(ls -ld mydir ; ls -ld mydir1) > temp3 2>&1

see it

- cat temp3
- what does this one do?

$(ls -ld mydir ; ls -ld mydir1) 2>&1 > temp4

$cat temp4

standard input, output and error

- what does this one do? [second output (stderr) to first output (terminal) … well second output of the command goes to temp4]

$(ls -ld mydir ; ls -ld mydir1) 2>&1 > temp4

$cat temp4

temp is only for teaching pruposes! use bit bucket (black hole of null device)

/dev/null

- a special file that discards all data written to it but reports that the write operation succeeded.

you often see

$mycommand > /dev/null

- redirect channel stdout of mycommand to /dev/null

$ mycommand 2> /dev/null

- redirect channel stderr to /dev/null

$ mycommand > /dev/null 2>&1

redirect stdout to /dev/null and then bind channel 2 (stderr) to channel 1 (stdout). Both will go into /dev/null

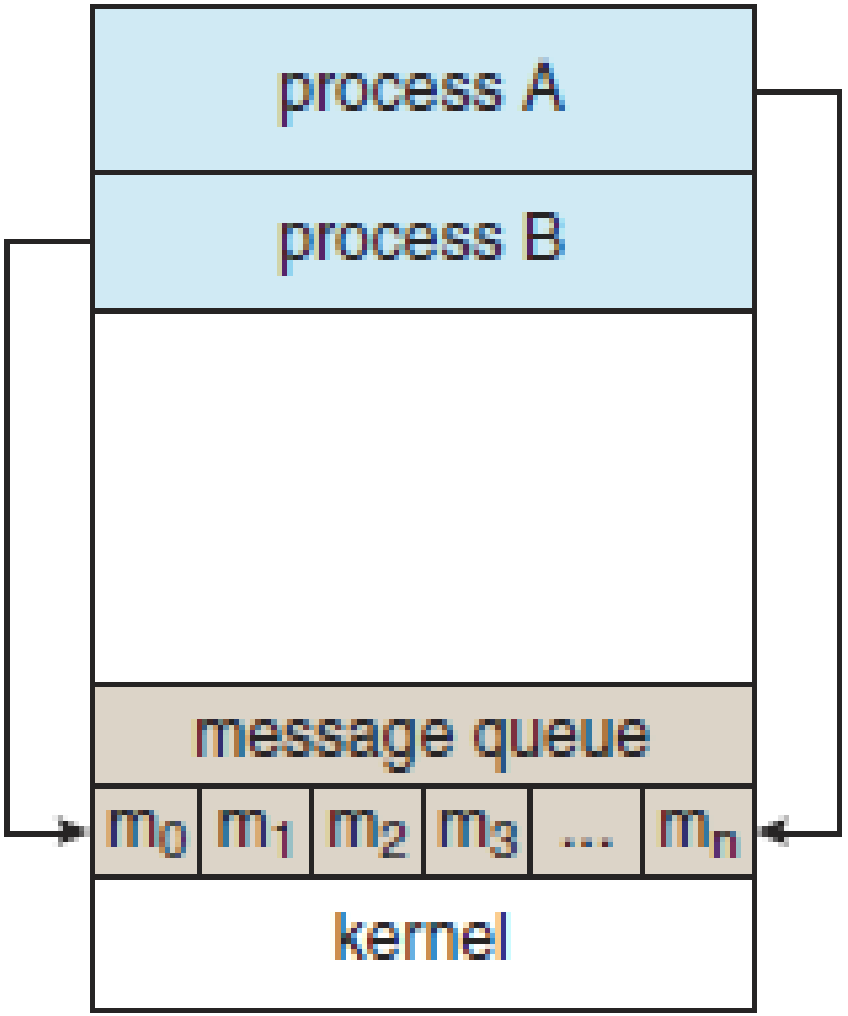
back to pipe

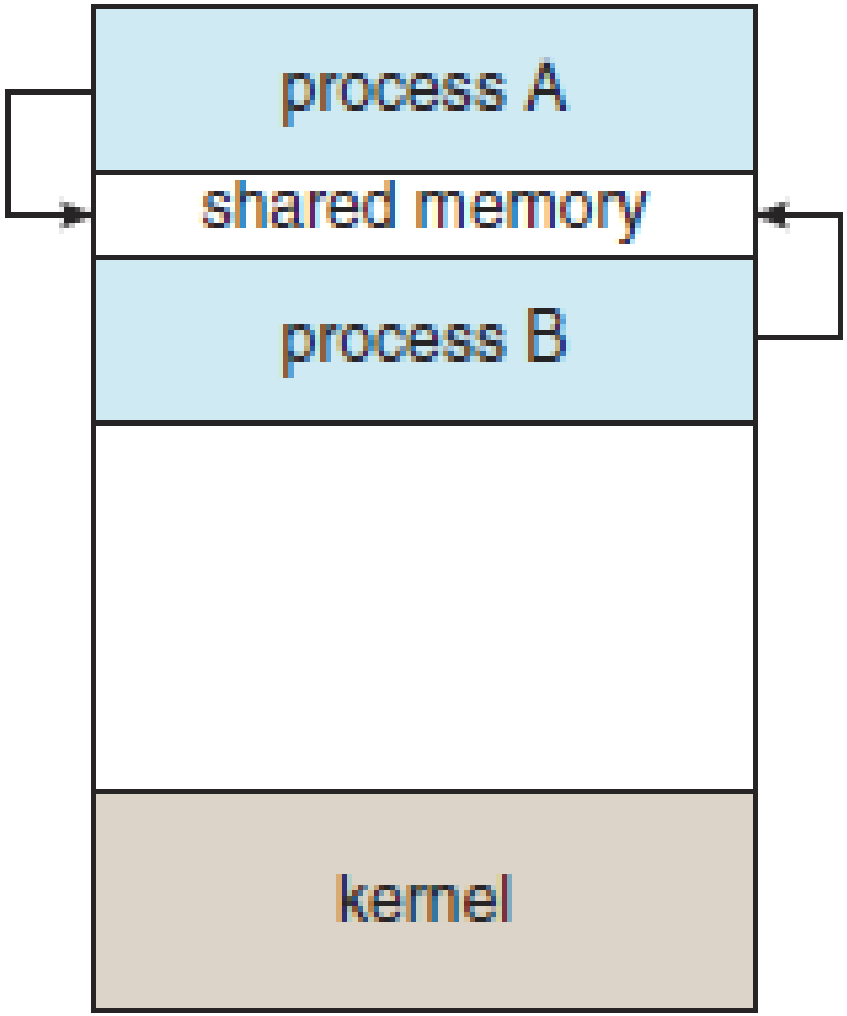
command1 | command 2

- a temporary file is created
- stdout of command1 is redirected to the temp file
- stdin of command2 comes from the file

- pipe is an example of communication via shared file.
- Any other way of processes communicating?

processes communicate in two ways





Between machines we need a network or communication medium!

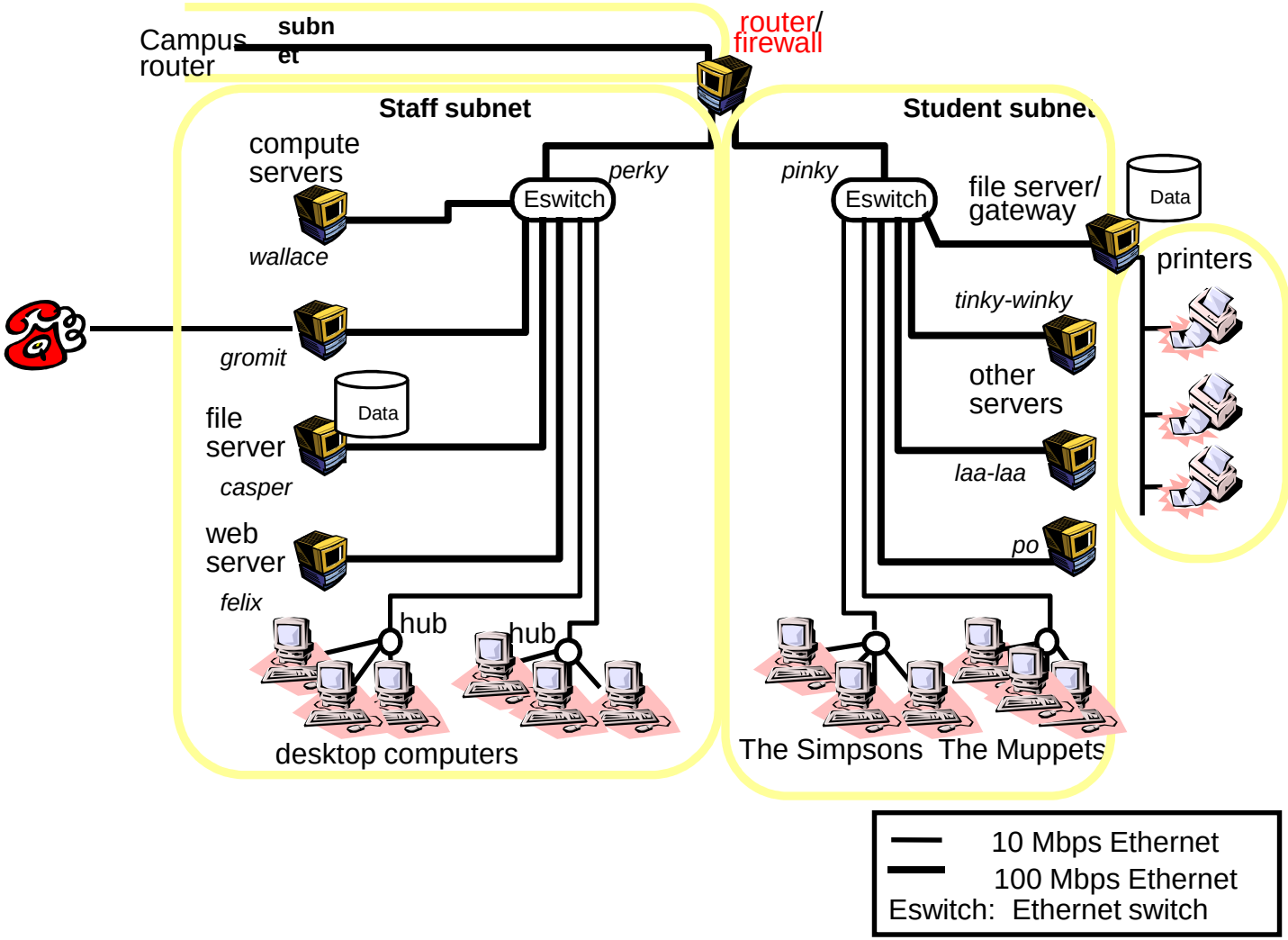
Types of Networks

- PAN (Personal Area Network)
- LANs (Local Area Networks)
- WANs (Wide Area Networks)
- MANs (Metropolitan Area Networks)
- WPAN (Wireless PAN)
- WLAN (Wireless LAN)
- WMAN (Wireless MAN)
- WWAN (Wireless WAN)

LAN

- messages are carried in high speed between connected nodes by a single communication medium
- Suitable for home office ,… radius of 1-2 km
- High bandwidth 10-1000Mbps (total amount of data per unit of time)
- Low latency 1-10 ms (time taken for a bit to reach destination)
- Technology: predominantly Ethernet

LAN example: the old SoCS



WAN

- Covers Worldwide,
- Low bandwidth 0.01-600 Mbps,
- high latency (100-500 ms)
- Satellite/wire/cable, use of routers which also introduce delays

MAN

Wire/cable, uses Digital Subscriber Line (DSL) and cable modem

Range of technologies (ATM, Ethernet)

Wireless networks

- WLANs (Wireless Local Area Networks)
  - to replace wired LANs
  - WaveLAN technology (IEEE 802.11)
- WPANs (Wireless Personal Area Networks)
  - variety of technologies
  - GSM, infra-red, BlueTooth low-power radio
  - WAP (Wireless Applications Protocol)

Network comparison

	Range	*Bandwidth (Mbps)*	*Latency (ms)*
LAN	1-2 kms	10-1000	1-10
WAN	worldwide	0.010-600	100-500
MAN	2-50 kms	1-150	10
Wireless LAN	0.15-1.5 km	2-11	5-20
Wireless WAN	worldwide	0.010-2	100-500
Internet	worldwide	0.5-600	100-500

Fastest ever internet transfer is 1.4 terabits per sec (BT, 2014)

Guinness world record (Cisco):

South Korea has average download 33.5 megabits per second

second-place Hong Kong – 17 megabits per second

Network principles

- Mode of transmission
- Switching schemes
- Protocol suites
- Routing
- Congestion control

Mode of transmission

- Packet Transmission
  - messages divided into packets. Example: 01101110
  - packets queued in buffers before sent onto link
  - QoS not guaranteed
- Data streaming
  - links guarantee QoS (rate of delivery)
  - for multimedia traffic
  - higher bandwidth

Switching schemes

- Broadcasts (Ethernet, wireless)
  - send messages to all nodes
  - nodes listen for own messages (carrier sensing)
- Circuit switching (phone networks)
- Packet switching (TCP/IP)
  - store-and-forward
  - unpredictable delays
- Frame/cell relay (ATM)
  - bandwith & latency guaranteed (virtual path)
  - small, fixed size packets (padded if necessary)
  - avoids error checking at nodes (use reliable links)

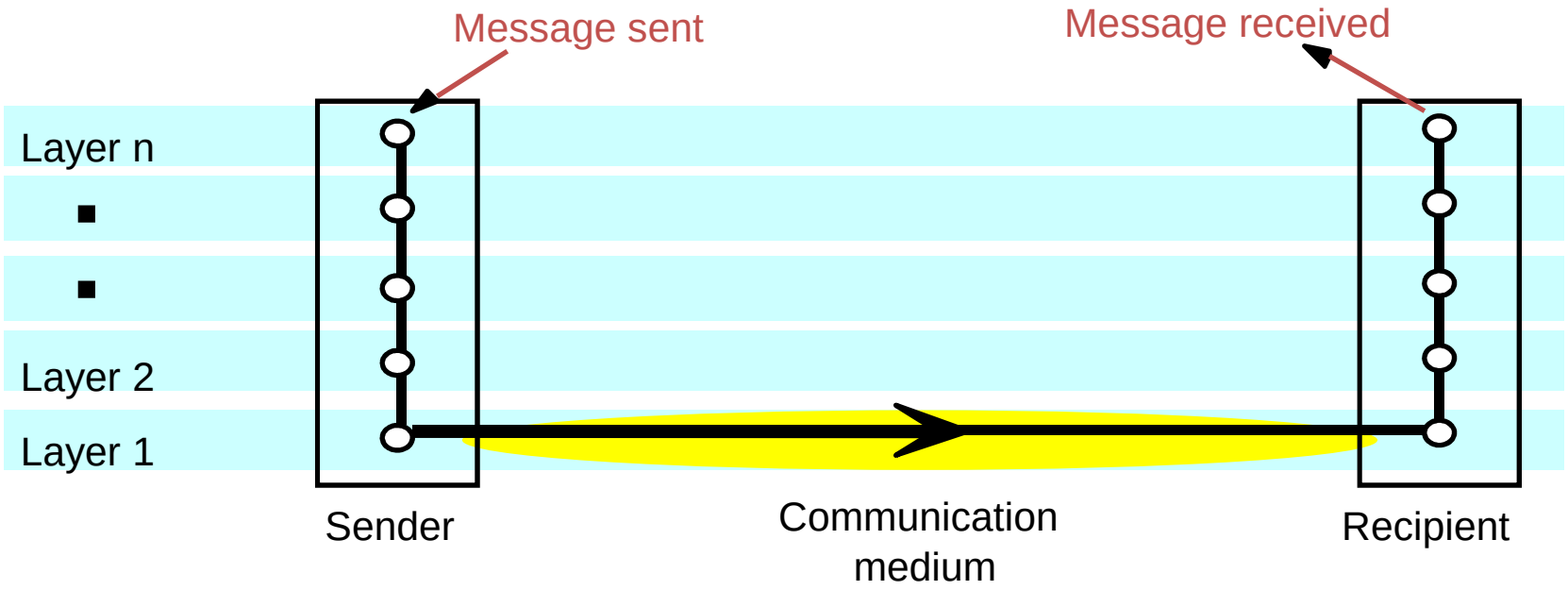
Protocol

- well-known set of rules and formats to be used for communication between processes to perform a given task
- Two parts:

❑specification of sequence of messagegs that must be exchanged

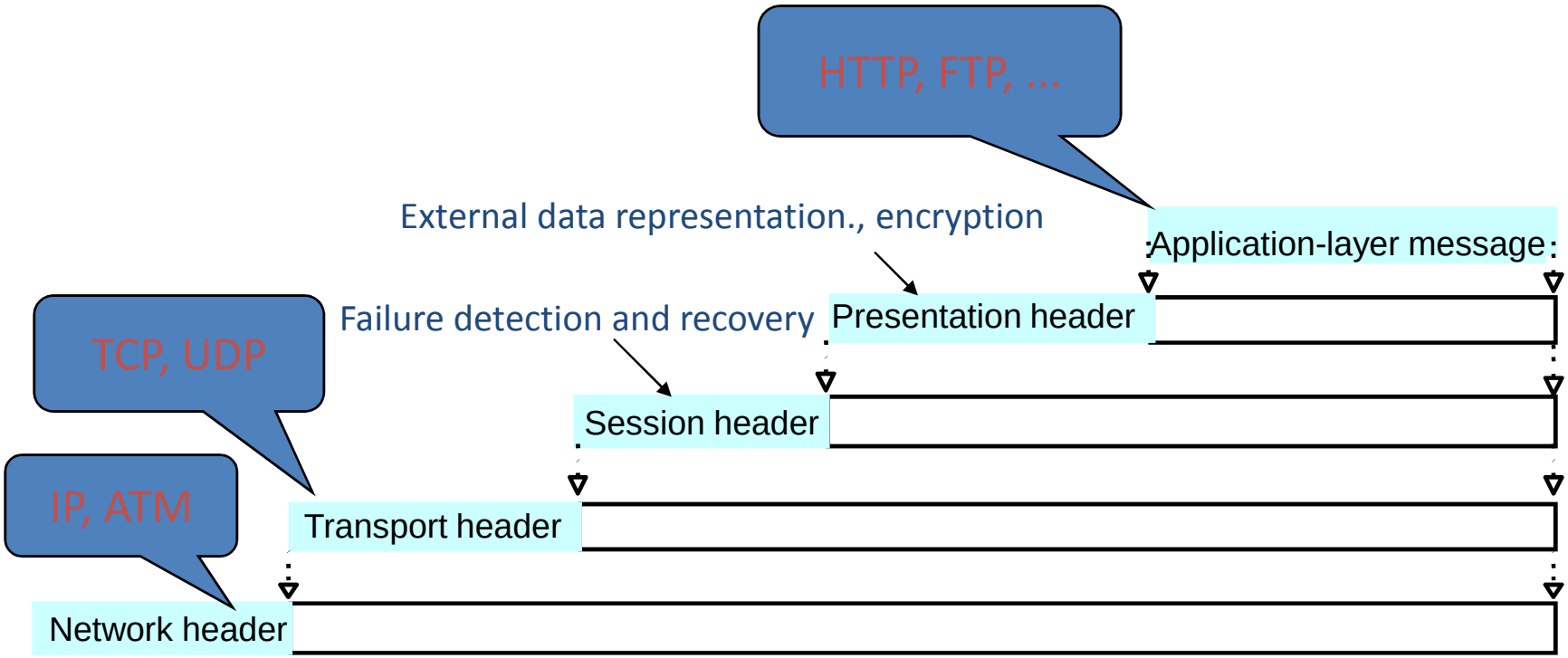
❑specification of the format of the data in each message

Protocols (OSI view)



Definition: set of rules and formats for exchanging data, arranged into layers called protocol suite/stack.

Message encapsulation



Headers appended/unpacked by each layer.

OSI protocol summary

Layer	*Description*	*Example*
Application	Protocols for specific applications.	HTTP, FTP, SMTP
Presentation	Protocols for independent data representation and encryption if required.	Secure Sockets, CORBA CDR
Session	Protocols for failure detection and recovery.	
Transport	Message-level communication between ports attached to processes. Connection-oriented or connectionless.	TCP, UDP
Network	Packet-level transmission on a given network. Requires routing in WANs and Internet.	IP, ATM
Data link	Packet-level transmission between nodes connected by a physical link.	Ethernet MAC, ATM cell transfer
Physical	transmit sequence of binary data using various mediums	Signalling, ISDN

----- lecture 10
--Automata
Example 1:
{u,d}
Example 2: going to uni and study and return
o: leave home
b: bike
w: walk
s: study
r: return
--What is a language?
Set of strings on an alphabet
--Regular expression
E | E+E | EE | E*
-- Equivalence of regular expressions with Automata
web site that converts

-- Lots of Syntactic sugar:
$ grep "12345" samplefile

^something all lines starting with something
 $ ls -al | grep "^d"
 $ ls -al | grep "^drw"
 $ grep "^Art" samplefile

-i for ignore case
-v for inverse match   i.e. cases that we dont want to match
-n line number
 $ grep -i "art" samplefile
 $ grep -v "art" samplefile
 $ ls -al | grep -v "^drw"

$ for end of line
grep "punishment." samplefile
what does this do?
$ ls .. | grep "sh$"

what does this do?
$ grep -n "^$" samplefile
Empty lines with their number

. for exactly one
*grep ".ion" samplefile

the followings are clear A-Z a-z 1-8

[aeiou] means a or e or i or ...
what is this ?
$grep "19[5-8][0-9]" samplefile

*   0 or more

grep -E “the*” samplefile
+ one or more
? 0 or 1
{5} exactly five
{1,3} means 1,2 or 3
what does this do?
[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}
You need to use extended grep by flag -E or use egrep which is shorthand for grap -E
$ grep -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}" samplefile

.* any number of characters
$ grep -E "1.*" samplefile
search for email with yahoo in it
$ grep -E "@.*yahoo" samplefile

ATTENTION [^chars]: pay attention to difference between “^xyx” and “[^xyz]”
first one says start with x second one says anything but x or y or z
$ ls -al | grep "^d"
$ ls -al | grep "[^d]"
or
egrep "^Article" samplefile
 egrep "[^Article]" samplefile
or
[^a-z] anything except lowercase

– something helpful about grep
You can include and exclude files in your grep. For example imagine you are looking for the word customer in your .java files
$grep -i “customer” . -r –include *.java

There are lots of cheat sheets that you can use such as http://regexpr.com/cheatsheet/

--- cut
grep cuts line-by-line
imagine we want to cut by column
- look at sample file and print peoples name and their
cut -d" " -f 1,3 sample

--would that work if i add a small space between entries? experiment
somedata has a small csv list of uk-500 companies.
print firstname lastname and address.
$ cut -d"," -f 1,2,4 somedata

--- exit status
Every command that is executed return back a value 1 if failed and 0 if succeed. You can get it by $echo $?

--- write about if clauses
see
l10-ifthen.sh

--How can you get rid of "cp: cannot stat ..."

(Hint: bit bucket)
l10-ifthenv2.sh

----using if with numbers
see l10-ifelsenumbercomparison.sh

--comparison operators are
-eq equal to
-ne not equal to
lt less than
-le less than or equal to
-gt greater than
-ge greater than or equal to

ATTENTION: watch out for space after [and before]

--Exercise: write a program to check for a single character input, i.e. reads an input and prints "single character" message
if input is single character.
Hint pipe to wc -c why?
Hint: does it work? Have you considered carriage return? what?

You also have logical operators such as
-a for and
-o for or
see l10-logicaland.sh
Exercise: write a program to extend the previous exercise and check for a single character and then if the character
is a,b or c prints the message "abc" and otherwise print the message "notABC".

---- working with files
- checking somethis is file not a directory
$ if [-f sample]; then echo "file exist"; fi

-- there are a whole bunch of file operations:
-s file exists and is not empty
-f file exists and is not a directory
-d directory exists
-x file is executable
-w file is writable
-r file is readable
-z File has 0 character return t

Exercise: write a script that checks if a file exists and is writeable. Print appropriate messages.

Exercise: write a program to add index.html file into every directory that does not have one.

IP

Lecture 11:

Operating Systems and Networks

Behzad Bordbar

recap

- processes communicating same machine: shared object and pipe
- Need network to communicate across machines
- Different type of network
- Modes of transmission (circuit switching and packet switching)
- protocols: (well-known set of rules and formats to be used for communication between processes to perform a given task)
- OSI view [most layers interacting with layer below or above]

Contents

- IP
- Datagram
- Routing protocol RIP1
- other IP protocols
- ping traceroute….

IP

- TP: transmission mechanism used by TCP and UDP
- uses other protocols ARP, RARP, ICMP …
- Best effort delivery (post office in Romeo): Unreliable and connectionless protocol
- No error checking or tracking
- Datagrams can be lost for various reasons
- noise converting a 0 to 1
- Congested router might drop packages
- loop because of bad networking and datagram times out
- broken link
- IP must be paired with another protocol to become reliable

Datagram

- packets in IP: two parts Header and data

Header (20-60 bytes)

- version (4bits) IPv4
- HLEN: Header Length (4bits) (0…15 multiple of 4)
- service type (8bits)(priority, throughput, delay
- Total length (16bits) 65535 bytes
- …
- Time to live (8bits) how many hops can go
- protocol (8bits)

Datagram

- Header checksum (16 bits)
- Source IP address
- Destination IP address

Body

How can I see the packets?

At home… not here!!!!

tcpdump or wireshark (windump on windows)

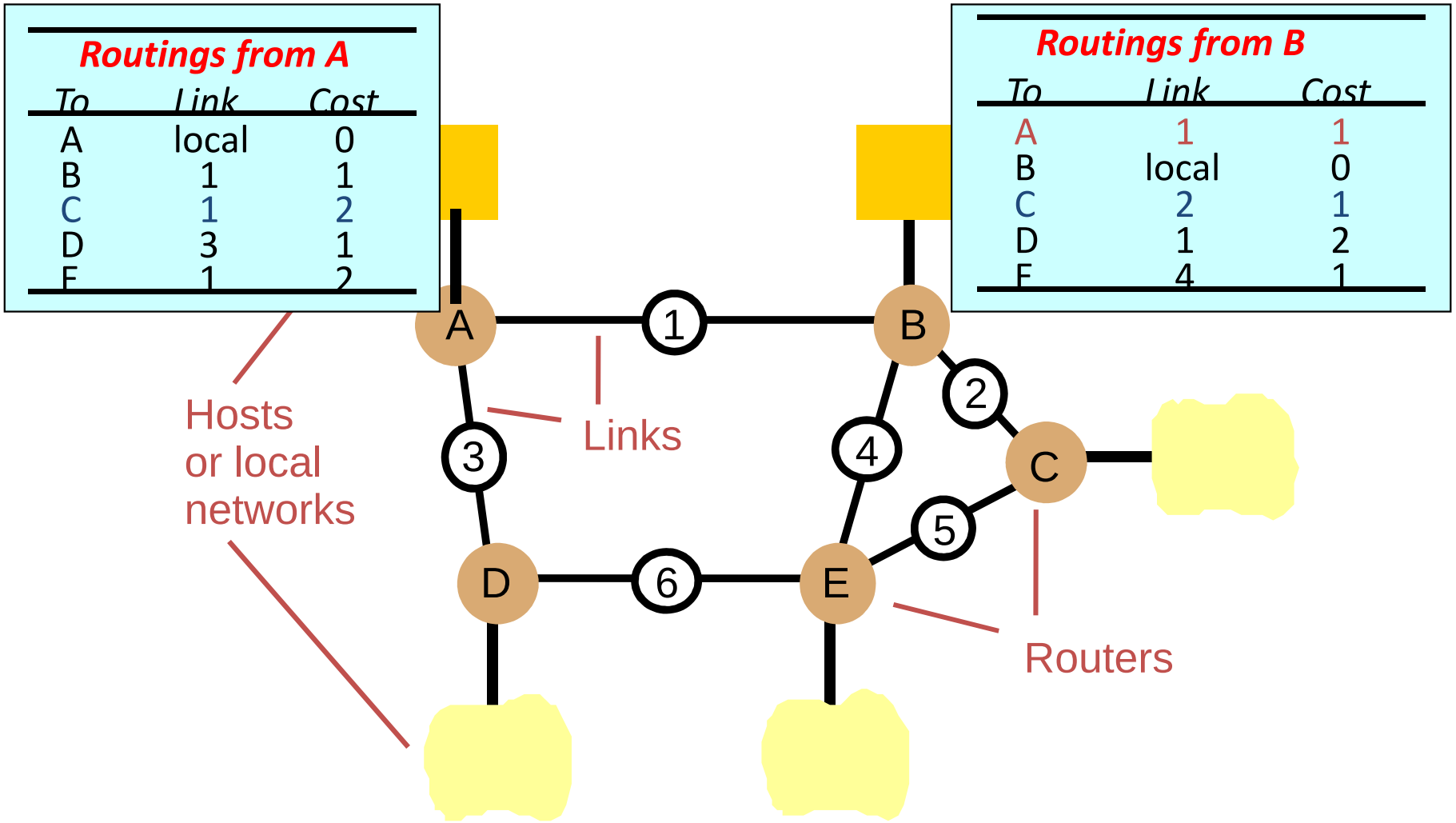
observe mac addresses …

Routing

- Necessary in non-broadcast networks (cf Internet)
- Next we look at a simple routing algorithm which IP is base on called Distance-vector algorithm
- each nodestores table of state
- cost info of links,
- cost infinity for faulty links
- determines route taken by packet (the next hop)
- periodically updates the table and sends to neighbours
- Theoretical foundation [Bellman-Ford]

- Internet similar except
  - use default routes, plus multicast and authentication
  - better convergence

Routing example



Routing tables

Routings from A

To	*Link*	*Cost*
A	local	0
B	1	1
C	1	2
D	3	1
E	1	2

Routings from B

To	*Link*	*Cost*
A	1	1
B	local	0
C	2	1
D	1	2
E	4	1

Routings from C

To	*Link*	*Cost*
A	2	2
B	2	1
C	local	0
D	5	2
E	5	1

Routings from D

To	*Link*	*Cost*
A	3	1
B	3	2
C	6	2
D	local	0
E	6	1

Routings from E

To	*Link*	*Cost*
A	4	2
B	4	1
C	5	1
D	6	1
E	local	0

RIP routing algorithm

Update: Each *30* seconds or when local table changes, send update on each non-faulty outgoing link.

Propagation*:*When router X finds that router Y has a shorter and faster path to router Z, then it will update its local table to indicate this fact. Any faster path is quickly propagated to neighbouring rotes through the ***Update*** process.

Shown to converge by mathematicians (Bertsekas).
See next slide for details.

RIP routing algorithm

Variables: Tl local table, *Tr* table received.

Send: Each *t* seconds or when *Tl* changes, send *Tl* on each non-faulty outgoing link.

Receive: Whenever a routing table *Tr* is received on link *n*:

```
for all rows Rr in Tr {
    if (Rr.link != n) {
        Rr.cost = Rr.cost + 1;
        Rr.link = n;
        if (Rr.destination is not in Tl) add Rr to Tl;
         // add new destination to Tl
        else for all rows Rl in Tl {
            if (Rr.destination = Rl.destination and
                    (Rr.cost < Rl.cost or Rl.link = n)) Rl = Rr;
            // Rr.cost < Rl.cost : remote node has better route
            // Rl.link = n : remote node is more authoritative
        }
    }
}
```

Sample routes

- Send from C to A:
  - to link 2, arrive at B
  - to link 1, arrive at A
- Send from C to A if B table modified to:
  - to link 5, arrive at E
  - to link 4, arrive at B
  - to link 1, arrive at A
- NB extra hop.

Routings from C

To	*Link*	*Cost*
B	2	1
C	local	0
E	5	1
default	5	-

other protocols that IP uses

- Address Resolution Protocol (ARP) associates an IP address with the physical address
- what is the ip address of www.cs.bham.ac.uk
- host makes an arp packet broadcast to everybody… all ignore except the host that ip belongs to

$arp www…

you can use tcpdump to see the arp packets

- Reverse Address Resolution protocol (RARP) the other way

other protocols that IP uses

- Internet Control Message Protocol (ICMP) mechanism to send (by host and routers) send notification about the datagram back to sender. … similar to postcard by julliet.

Exercise:

- learn about IP addresses and Mask
- ping
- traceroute

Lecture 12

--would that work if i add a small space between entries? experiment
somedata has a small csv list of uk-500 companies.
print firstname lastname and address.
$ cut -d"," -f 1,2,4 somedata

--- exit status
Every command that is executed return back a value 1 if failed and 0 if succeed. You can get it by $echo $?

--- write about if clauses
see
l10-ifthen.sh

--How can you get rid of "cp: cannot stat ..."
(Hint: bit bucket)
l10-ifthenv2.sh

----using if with numbers
see l10-ifelsenumbercomparison.sh

--comparison operators are
-eq equal to
-ne not equal to
lt less than
-le less than or equal to
-gt greater than
-ge greater than or equal to

ATTENTION: watch out for space after [and before]

--Exercise: write a program to check for a single character input, i.e. reads an input and prints "single character" message
if input is single character.
Hint pipe to wc -c why?
Hint: does it work? Have you considered carriage return? what?

NOT INCLUDE: if [`echo $var | wc -c` -eq 2] # 2 to count carriage return

You also have logical operators such as
-a for and
-o for or
see l10-logicaland.sh
Exercise: write a program to extend the previous exercise and check for a single character and then if the character
is a,b or c prints the message "abc" and otherwise print the message "notABC".

---- working with files
- checking somethis is file not a directory

$ if [-f sample]; then echo "file exist"; fi

-- there are a whole bunch of file operations:
-s file exists and is not empty
-f file exists and is not a directory
-d directory exists
-x file is executable
-w file is writable
-r file is readable
-z File has 0 character return t

Exercise: write a script that checks if a file exists and is writeable. Print appropriate messages.

Exercise: write a program to add index.html file into every directory that does not have one.

--- until do done
see example l12-01-untildodone.sh

------ Internal file separator used when sequences are separated
see example l12-02-IFS.sh
Good practice is to researve a variable for the old IFS and set it back·

------ sed
Stream editor used when want to replace format
cat afile | sed 's/pattern/replacement'
$ echo window | sed 's/o/0/' | sed 's/i/1/'
w1nd0w
 This replaces the FIRST occurance. If you want all occurrences add g

What would this do?
$ echo `ls -al` | sed 's/o/0/g' | sed 's/i/1/g' > afile

deleting something is done with '/pattern/d'
What does this do?
$ sed '/^$/d' afile

Exercise: write a program that replaces numberical values with number in a file. For example 123 replaced with OneTwoThree. (Hint if you dont want to make a new file just use sed -i)

You can use any regular expression and you can use & to match it. Suppose you want to put words in paranthesis for example abc in (abc)
$ echo My plug in baby |sed 's/\w\+/(&)/g'
(My) (plug) (in) (baby)

a good tutorial on sed is at http://www.grymoire.com/Unix/Sed.html

IP address

Lecture 13:

Operating Systems and Networks

Behzad Bordbar

recap

- interprocess communication across machines?
- application makes data > Application Layer (HTTP, FTP, SMTP, DNS, VoIP) > become message > Transport Layer (TCP, UDP) > become TCP/UDP datagrams> Internet layer (IP, ICMP) > become IP datagram > Network layer (Ethernet and X.25) > becomes Frame
- Frames are through physical medium
- reverse order at the destination!
- We looked at the IP layer

recap (RIP)

- *Variables: Tl* local table, *Tr* table received.
- *Send:* Each *t* seconds or when *Tl* changes, send *Tl* on each non-faulty outgoing link.
- *Receive:* Whenever a routing table *Tr* is received on link *n*:
  - for all rows *Rr* in *Tr* {
    - if (*Rr.link* != *n*) {
      Rr.cost = *Rr.cost* + 1;
      Rr.link = *n*;
      if (*Rr.destination* is not in *Tl*) add *Rr* to *Tl;*
      // add new destination to *Tl*
      else for all rows *Rl* in *Tl* {
        if (*Rr.destination* = *Rl.destination* and
          (*Rr.cost* < *Rl.cost* or *Rl.link* = *n*)) *Rl* = *Rr;*
        // Rr.cost < Rl.cost : remote node has better route
        // *Rl.link = n* : remote node is more authoritative
      - }
    - }
  - }

RIP (continue)

- When a link fails *cost* in the table is set to ∞
- Then, the cost in all table is set to ∞ (1+∞ = ∞)
- 3 timers: periodic, expiration, Garbage collection
- RIP is
  - slow in convergence
  - But, most of the time system stables fast
- RIP 2 addresses some of the above issues and also *Authentication* and *Multicasting* (send packets to all other router.

Congestion control

- When load on network high (80% capacity)
  - packet queues long, links blocked
- Strategies to address the problem
  - packet dropping
    - reliable of delivery at higher levels
    - Dropping some packets is better than others (MPEG)
  - reduce rate of transmission
    - nodes send choke packets (Ethernet)
    - transmission control (TCP)
  - transmit congestion information to each node
    - QoS guarantees (ATM)

Contents

- IP address classes
- Mask
- Private IP address
- subnet
- Classless Interdomain Address
- IPv6
- router gateway and all that

IP address

- identifies a computer
- IPv4 is 32 bits binary number
- four parts (octet) each 0 to 255=1111 1111
- used in ip_src and ip_dst in ip datagram
- finite number 2^32 … what if we run out?

IP address has two parts

- Network portion: part of IP address of where the device is located.
- Host ID portion: IP address that uniquely identifies the device on its network

computer, mobile phone, printer, ….

First octet give the class

class	1st octet range	1st octet highbits	No Networks	No hosts
A	1-126	0	126 ($2^7 - 2$)	16,777,214 ($2^{24} - 2$)
B	128-191	10	16,382 ($2^{14} - 2$)	65,534 ($2^{16} - 2$)
C	192-223	110	2,097,150 ($2^{21} - 2$)	254 ($2^8 - 2$)
D	224-239	1110	multicast	
E	240-254	1111	Experiment and	research

- first octet can NOT be 127 (kept for trouble shooting and testing your local system)
- local host: 127.0.0.1
- why 2 is subtracted?

*-cast

- multicast
- broadcast
- anycast

- back to ip address!

network Mask (or simply Mask)

- identifies the part of address which is network address
- class A: 255.0.0.0
- class B: 255.255.0.0
- class C: 255.255.255.0

Why word mask? ignore part of ip address that (in binary) correspond to 0.

What are important bits if mask is 190= (10111110)?

Answer: all except first and seventh

- Netmask is also used by protocols to decide if a packet is for internal machine or not. It should be handled within LAN or it should go to a router.
- Netmask is not put into packets, because it is only important for sending things NOT for receiving. Net mask are used for working out if computers are on the same network.

private IP address

All above are ip addresses given to machines and servers that are open to outside world.

- Private reserved addresses:
- If a packet with the address which private reserved and hits a router at the edge of organisation, it will not go out.
- class A: 10.0.0.0 to 10.255.255.255
- class B: 172.16.0.0 to 172.31.255.255
- Class C: 192.168.0.0 to 192.168.255.255
- There is another group (automatic, private, non-routables) We will see later. 169.254.0.0 169.254.255.255

Why private IP address?

- originally for testing and training.
- But we have lots of them? count them:
- Class A (private) above is 16M addresses
- Class B (private): 1M
- Class C (private): 65K
- Some companies assign these reserved addresses for their internal use. On the firewall they use Network Address Translation (NAT) to extend the range of addresses used in IPv4
- How does NAT work?

Subnet

- process of dividing a large network to smaller interacting networks to increase efficiency and manageability.
- IP addresses are hierarchical by nature Network Id and host Id, but only at two layers. Allows create multilayers.

Other reasons for using subnets

- security: protect parts differently
- organisational and division of jobs: different department
- political: we want to be independent.

Example

- Consider a C class address 193.171.120.0.
- Mask is 255.255.255.0
- You have 8 bits 0000 0000 turned on for your addresses
- you can choose the first two bits for subnet 1100 0000 or the first three bits 1110 0000
- To do so I can use the subnet mask of 255.255.255.192.
- Why 192?
- cause 1100 0000(base 2)= 128+64 (base 10)
- How many subnet I will have this case?
- Answer 4: 10... 01... 11.. 00... so the 254 addresses are divided into four subnets and within each I can have private address. Put a router in between them
- We wont go into how they are calculated, use a calculator? http://www.subnet-calculator.com/ be aware the more subnet the more broadcast and network addresses.

How to calculate subnet address?

- Given an IP address and a subnet mask calculate the subnet addresses:

Rules:

- if mask 255 corresponding part of the IP address repeated
- if mask 0 corresponding part of the IP address is set to zero
- otherwise bitwise “and” operation
- Example 156.72.56.5 with mask 255.255.200.0
- IP 181 92 56 5
- mask 255 255 200 0

subnet 181 72 8 0

Why 8?

- 56= (0011 1000)
- 200= (1100 1000)
- 0000 1000 which is 8

IP address (continue)

Lecture 14:

Operating Systems and Networks

Behzad Bordbar

recap

- IP address: two parts
- IP address classes

A: 1- 126,B: 128-191, C: 192-223,D: 224- 239
E:240-254 (127, broadcast, network address)

- Mask (both used for identifying Network/host id and telling if destination is on network without intermediate router)
- private IP addresses and NAT
- more than two level of hierarchy (subnet)
- what is the IP address of the subnet?

How to calculate subnet address?

- Given an IP address and a subnet mask calculate the subnet addresses:

Rules:

- if mask 255 corresponding part of the IP address repeated
- if mask 0 corresponding part of the IP address is set to zero
- otherwise bitwise “and” operation
- Example 156.72.56.5 with mask 255.255.200.0
- IP 181 92 56 5
- mask 255 255 200 0

subnet 181 72 8 0

Why 8?

- 56= (0011 1000)
- 200= (1100 1000)
- 0000 1000 which is 8

Classless Inter domain Address (CIDR)

- suppose a company with 1000 machines (say SoCS), what type of IP address should I buy? see slide on classes
- purchase 5 class C address >> must manage multiple address!
- buy a class B address >> 64000 addresses go to waste!
- Solution Classless Interdomain Routing (CIDR) with Variable Length Subnet Masks (VLSM)

CIDR (Continue)

- address form are xy.xyz.xyz/N, when N is the offset (in binary) showing network address
- How many address are there in the CIDR address 192.168.100.0/22 ?
- 192= (1100 0000)
- 168= (1010 1000)
- 100 = (0110 0100)

so the ip address is

1100 0000.1010 1000.0110 0100.0000 0000

^ 22nd digit

- Range is between 00 0000 0000 and 11 1111 1111, which gives total number of 11 1111 1111 +1 = 100 0000 0000 which is 1024 numbers.
- solved the problem of 1000 machine with one address!
- routing table (CIDR routers) modified to have the mask.

how to assign IP address to devices?

- Static: manually add the ip address, network mask and default router (router that this machine is connected to)
- dynamic DHCP A server that has a pool of IP addresses and assigns them to users
- Automated Private IP Addressing APIPA [to produce network stack for applications that fail to work without a network stack]
  - A temporary IP address in the range
    169.254.0.1 to 169.254.255.254
  - if your DHCP server is down or your static ip address is not set correctly.
  - Not publically available addresses.
  - Randomly pick one address.
  - Computers on the same range can speak to each other.
  - But your device can not work with outside world.

IPv6

- permanent solution to shortage of IP address
- 128 bits in hex
- the number that
- 7X10^23 ip addresses per square meter of the earth....
- What is ipv6 google? How about pinging?

$ping6 ipv6.google.com or $host www.go...

Not only larger addresses?

- no fragmentation or checksum to increase routing speed, TCP takes are of this

IPv6 (continue)

- provision for real-time traffic. To ensure QoS (not having jitter throughput…) some packets are marked as *flow* and will be processed faster.
- IPv6 supports anycast
- Security in IPv4 is by authentication and encryption at higher layers. IPV6 pushes the security down to IP. For authentication and payload encryption, header is extended.

For further discussion on IPv6 see Tanenbaum.

router, gateway and all that

- Repeater:
  - extends the physical length of network. Signals can travel fixed distance.
  - don’t change the functionality.
  - produce a refreshed copy of the signal before it becomes too weak.
  - only operate at physical layer.
- Bridge:
  - Divide a large network into smaller segments.
  - used to relay frames between two separate LAN.
  - can have logic to filter network traffic for each segment separate.
  - also do the job of repeaters by regenerating packets and send them to the right segment.
  - can be single port connecting only two segment or multiports… .
  - have a table for directly of packets.
  - can be transparent, they learn and build the table as they dirct the packets.
  - operates both at physical and datalink layer of the OSI model.
  - (due to cheapness of switches bridges are not popular any more)
- Switch: provides bridges functionality with greater efficiency. They act like multiport bridges connecting segments in LAN.

router, gateway and all that

Bridges and repeaters are simple devices

Router:

- relay packets among multiple interconnected networks.
- operate across physical and datalink layer and networklayer.

Gateways:

- operate across all seven OSI layers.
- A router by itself transfers, accepts, and relays packets only across networks using similar protoocls.
- A gateway is a protocol converter. [packets in AppleTalk can be turned into TCP/IP packets before forwarding]
- Gateways are a software installed within a router
- It understands the protocols and adjust the header, datarate, size and format.
- Default gateway is the IP address of a router that a computer uses to go outside its network.
- Draw a picture!

Now that you have learnt

Experiment with your network.

- learn about ping, tcpdump, traceroute, netstat,
- **explore** you IP configuration and your network: ifconfig, iwconfig, iwlist,… what are these?

Knowing your shell programming you can do lots of cool things!

Review of some Protocols

- IP: transfer datagram from one host to another
  - Unreliable best effort
  - only header checksum
- TCP and UDP
  - main transport level protocols used by IP
- MobileIP
  - connectivity for mobile devices, even in transit
  - device retains single IP address
  - re-routing by Home (HA) and Foreign Agents (FA)
  - transparent
- Wireless LAN (IEEE 802.11)
  - radio or infra-red communications
  - CSMA/CA based

IP

- Internet Protocol is unreliable and connectionless
- Best effort (no error checking or Ack)
- Packets called Datagram have Header
  - IP address of source and destination
  - Containing version, Header Length (HLEN), length (Header +data) …
  - higher level protocol ?? (info encapsulated UDP, TCP, RIP2, …)
  - header checksum
  - Fragmentation
  - Timestamp (IP address of the router + Universal time)
- …
- Not all exploited by all higher level protocols

Transport layer protocols

- UDP (basic, used for some IP functions)
  - uses IP address + port number
  - no guarantee of delivery, optional checksum
  - messages up to 64KB
- TCP (more sophisticated, most IP functions)
  - data stream abstraction, reliable delivery of all data
  - messages divided into segments, sequence numbers
  - sliding window, acknowledgement+retransmission
  - buffering (with timeout for interactive applications)
  - checksum (if no match segment dropped)

Both are process-to-process communication

Draw packets!

Communication service types

- Connectionless: UDP
  - 'send and pray' unreliable delivery
  - efficient and easy to implement
- Connection-oriented: TCP
  - with basic reliability guarantees
  - less efficient, memory and time overhead for error correction

Connectionless service

- UDP (User Datagram Protocol)
  - messages possibly lost, duplicated, delivered out of order, without telling the user
  - maintains no state information, so cannot detect lost, duplicate or out-of-order messages
  - each message contains source and destination address
  - may discard corrupted messages due to no error correction (simple checksum) or congestion
- Used e.g. for DNS (Domain Name System) or RIP.

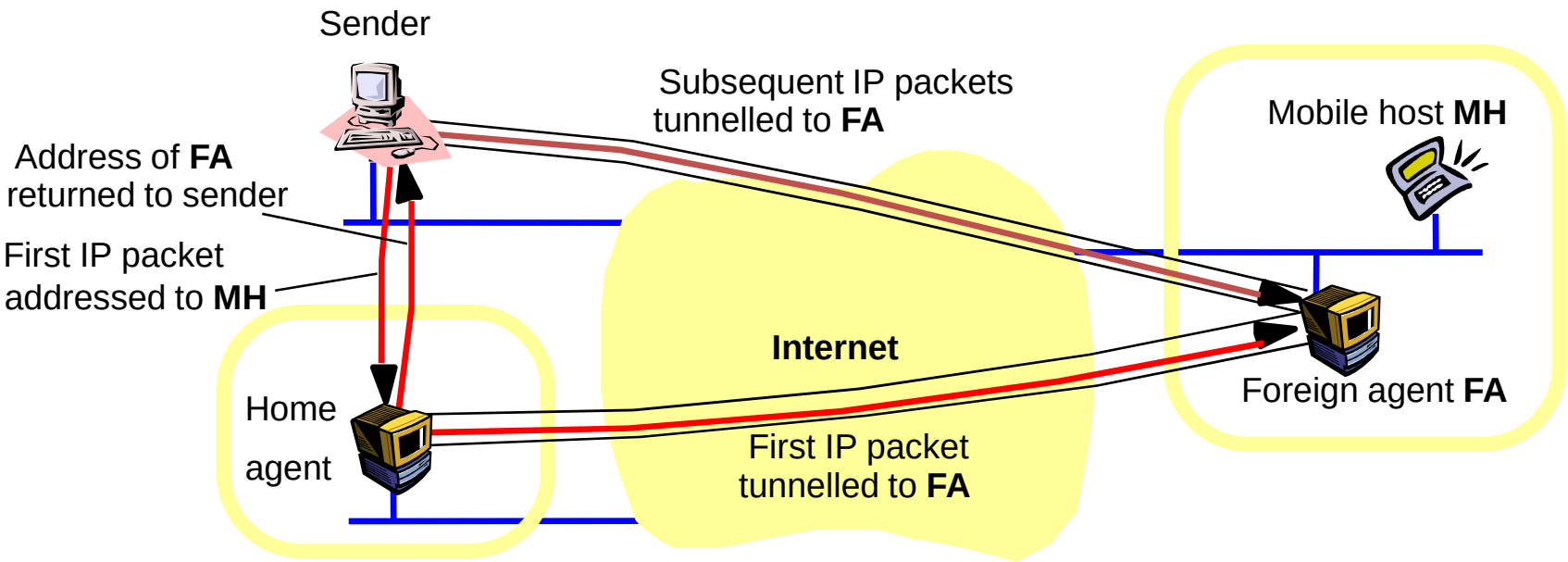
Connection-oriented service

- TCP (Transmission Control Protocol)
  - establishes data stream connection to ensure reliable, in-sequence delivery
  - error checking and reporting to both ends
  - attempts to match speeds (timeouts, buffering)
  - sliding window: state information includes
    - unacknowledged messages
    - message sequence numbers
    - flow control information (matching the speeds)
- Used e.g. for HTTP, FTP, SMTP on Internet.

MobileIP

- At home normal, when elsewhere mobile host:
  - notifies HA before leaving
  - informs FA, who allocates temporary care-of IP address & tells HA
- Packets for mobile host:
  - first packet routed to HA, encapsulated in MobileIP packet and sent to FA (tunnelling)
  - FA unpacks MobileIP packet and sends to mobile host
  - sender notified of the care-of address for future communications which can be direct via FA
- Problems
  - efficiency low, need to notify HA

MobileIP routing



Wireless LAN (802.11)

- Radio broadcast (fading strength, obstruction)
- Collision avoidance by
  - slot reservation mechanism by Request to Send (RTS) and Clear to Send (CTS)
  - stations in range pick up RTS/CTS and avoid transmission at the reserved times
  - collisions less likely than Ethernet since RTS/CTS short
  - random back off period
- Problems
  - security (eavesdropping), use shared-key authentication

Transport layer protocols

Lecture 15:

Operating Systems and Networks

Behzad Bordbar

recap

- interprocess communication across machines?
- application makes data > Application Layer (HTTP, FTP, SMTP, DNS, VoIP) > become message > Transport Layer (TCP, UDP) > become TCP/UDP datagrams> Internet layer (IP, ICMP) > become IP datagram > Network layer (Ethernet and X.25) > becomes Frame
- Frames are through physical medium
- reverse order at the destination!
- We looked at the IP layer
- IP Address, mask , subnet, IPv6

Contents

- Transport layer protocols
- TCP, UDP
- Format of headers
- comparison

Transport layer protocols: UDP

- IP is between machines using IP address.
- TCP and UDP between processes. How?
- Port (16 bits address): delivery of messages within a particular computer. IP delivers to computer, TCP and UDP software delivers to process.
- UDP (basic, used for some IP functions)
- encapsulated inside IP packet (IP addresses of computers)
- Header of UDP datagram has : source port number 16 bits, destination port number 16 bits, total length 16 bits, Checksum 16 bits
- no guarantee of delivery, optional checksum
- messages up to 64KB

Transport layer protocols: TCP

- TCP (reliable, stream transport, port-to-port protocol)
- stream means connection-oriented: connection must be established before data is transferred.
- virtual circuit is created between sender and receiver which remains active (may be routed different ways)
- TCP alert receiver the data is coming.
  - data stream abstraction, reliable delivery of all data
  - messages divided into segments, sequence numbers
  - sliding window, acknowledgement+retransmission
  - buffering (with timeout for interactive applications)
  - checksum (if no match segment dropped)

Transport layer protocols: TCP

- Header of TCP segment
- source/destination port number each 16 bits
- Sequence number (32 bits): shows position of the segment in the original data stream
- Acknowledgement number (32 bits): if ack bit is on, it has number of next sequence expected (ackn… current one)
- HLEN (4 bits) showing number of 32bits words in TCP header
- reserved (6 bits) for future use
- six one bit control fields: URG, ACK,…

Transport layer protocols: TCP

- Window size (16 bits) size of sliding window
- Checksum (16 bits) error detection
- Urgent pointer (16 bits) if URG turned on, what part is urgent data
- padding

Communication service types

- Connectionless: UDP
  - 'send and pray' unreliable delivery
  - efficient and easy to implement
- Connection-oriented: TCP
  - with basic reliability guarantees
  - less efficient, memory and time overhead for error correction

Connectionless service

- UDP (User Datagram Protocol)
  - messages possibly lost, duplicated, delivered out of order, without telling the user
  - maintains no state information, so cannot detect lost, duplicate or out-of-order messages
  - each message contains source and destination address
  - may discard corrupted messages due to no error correction (simple checksum) or congestion
- Used e.g. for DNS (Domain Name System) or RIP.

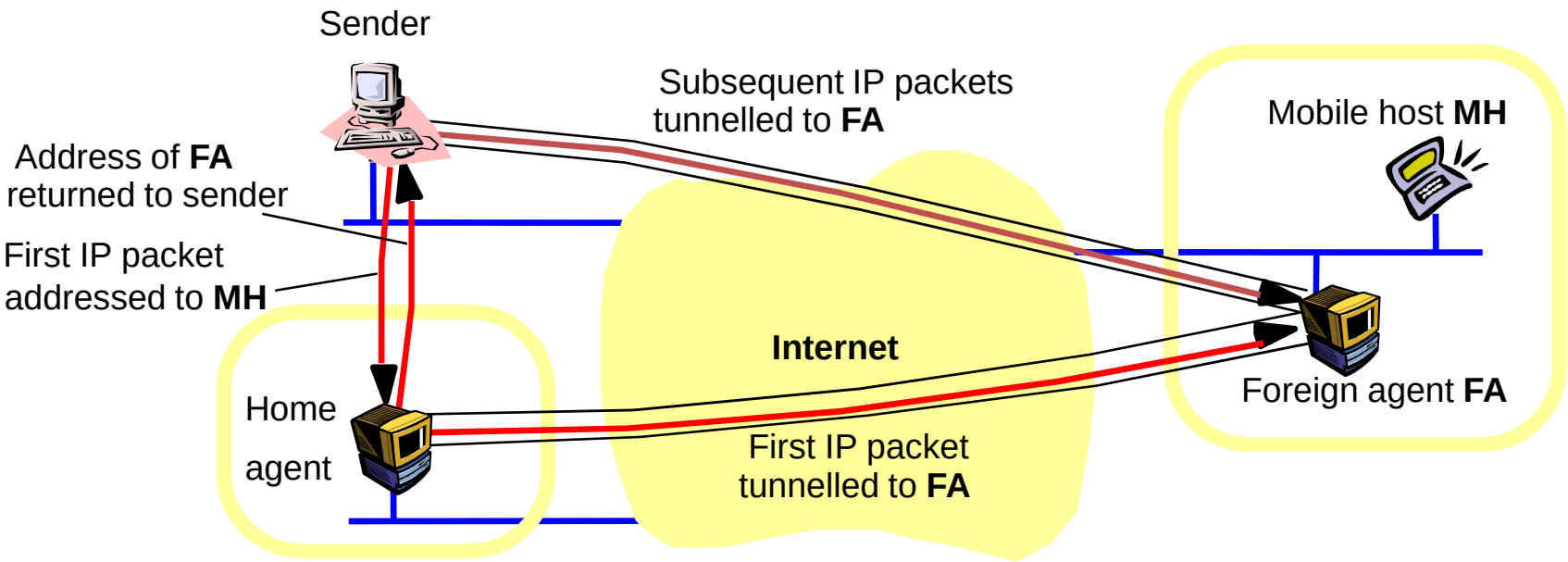
Connection-oriented service

- TCP (Transmission Control Protocol)
  - establishes data stream connection to ensure reliable, in-sequence delivery
  - error checking and reporting to both ends
  - attempts to match speeds (timeouts, buffering)
  - sliding window: state information includes
    - unacknowledged messages
    - message sequence numbers
    - flow control information (matching the speeds)
- Used e.g. for HTTP, FTP, SMTP on Internet.

MobileIP

- At home normal, when elsewhere mobile host:
  - notifies HA before leaving
  - informs FA, who allocates temporary care-of IP address & tells HA
- Packets for mobile host:
  - first packet routed to HA, encapsulated in MobileIP packet and sent to FA (tunnelling)
  - FA unpacks MobileIP packet and sends to mobile host
  - sender notified of the care-of address for future communications which can be direct via FA
- Problems
  - efficiency low, need to notify HA

MobileIP routing

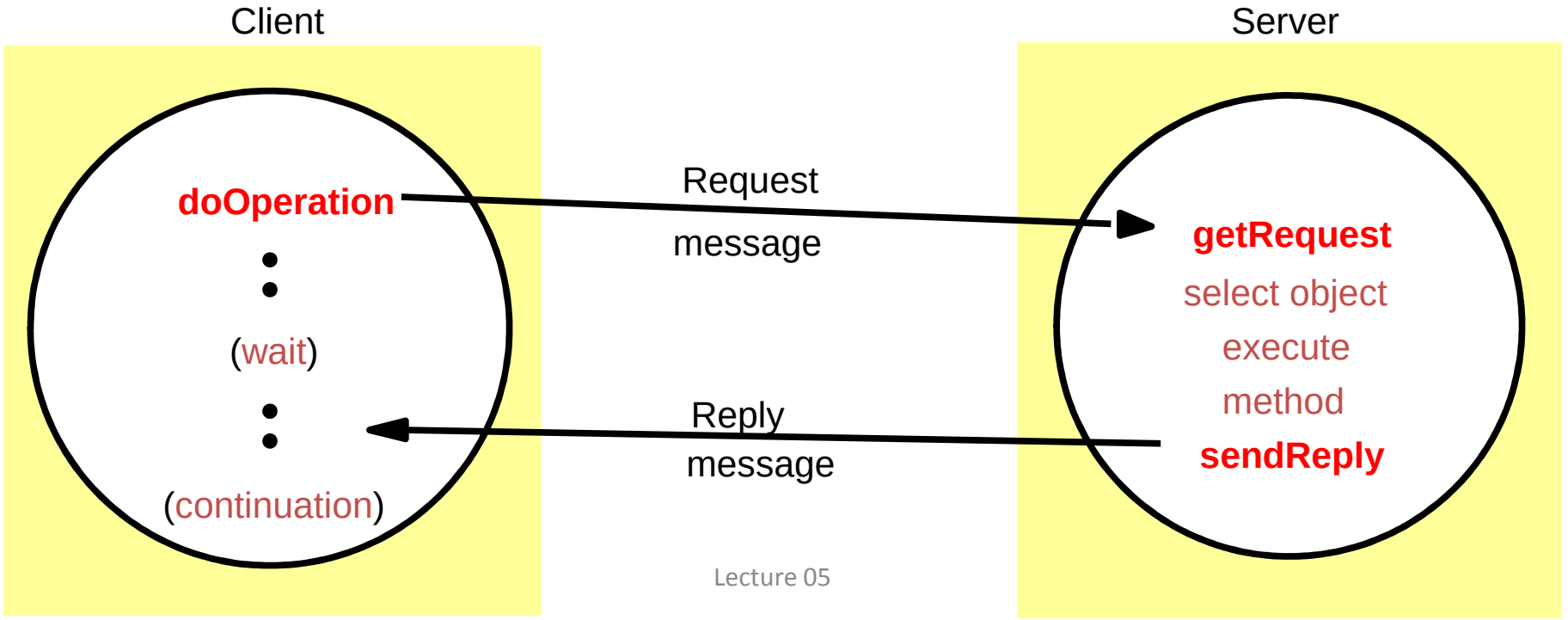


Wireless LAN (802.11)

- Radio broadcast (fading strength, obstruction)
- Collision avoidance by
  - slot reservation mechanism by Request to Send (RTS) and Clear to Send (CTS)
  - stations in range pick up RTS/CTS and avoid transmission at the reserved times
  - collisions less likely than Ethernet since RTS/CTS short
  - random back off period
- Problems
  - security (eavesdropping), use shared-key authentication

Interprocess communication

- Synchronous and asynchronous comm.
- Message destination
- Reliability
- Ordering



Asynchronous vs. Synchronous

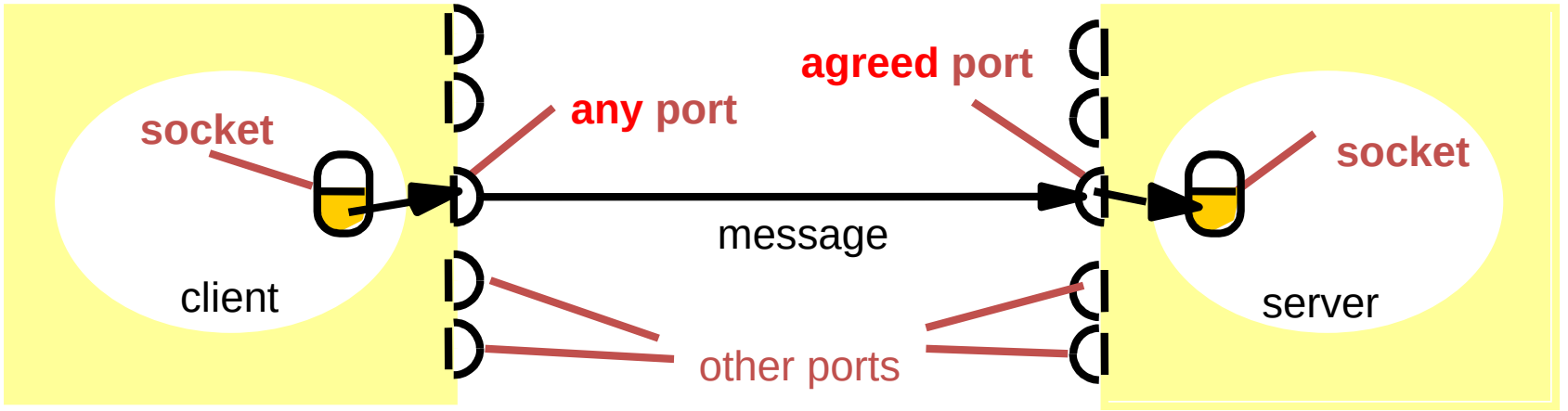
- Synchronize communication: sender and receiver
- synchronise on every message, i.e. *blocking* operations
- Asynchronous

❑ send is non-blocking

❑ receive can be blocking/non-blocking

Which one is better?

Message destination: Socket +Port



Socket = Internet address + port number.

Only one receiver but multiple senders per port.

Sockets

- Characteristics:
  - endpoint for inter-process communication
  - message transmission between sockets
  - socket associated with either UDP or TCP
  - processes bound to sockets, can use multiple ports
- Implementations
  - originally BSD Unix, but available in Linux, Windows,…
  - here Java API for Internet programming

Operations of Request-Reply

- *public byte[] doOperation (RemoteObjectRef o, int methodId, byte[] arguments)*
  - sends a request message to the remote object and returns the reply.
  - the arguments specify the remote object, the method to be invoked and the arguments of that method.
- *public byte[] getRequest ();*
  - acquires a client request via the server port.
- *public void sendReply (byte[] reply, InetAddress clientHost, int clientPort);*
  - sends the reply message reply to the client at its Internet address and port.

Java API for Internet addresses

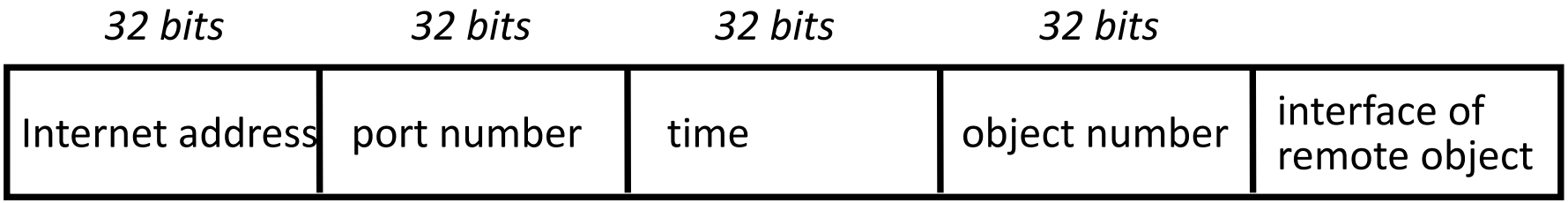
- Class *InetAddress*
  - uses DNS (Domain Name System)

```
InetAddress aC =
   InetAddress.getByName("gromit.cs.bham.ac.uk");
```

  - throws *UnknownHostException*
  - encapsulates detail of IP address (4 bytes for IPv4 and 16 bytes for IPv6)

Remote Object Reference

- An identifier for an object that is valid throughout the distributed system
  - must be unique
  - may be passed as argument, hence need external representation



Reliability

- Reliable communication:
- messages are guaranteed to be delivered despite a
- ‘reasonable’ number of packets being dropped or lost

Unreliable communication:

- messages are not guaranteed to be
- delivered in the face of even a single packet dropped or lost
- >>> Failure

Failure in point 2 point comm.

- DSs expected to continue if failure has occurred:
  - message failed to arrive
  - process stopped (and others may detect this)
  - process crashed (and others cannot detect this)
- Types of failures
  - benign
    - omission, stopping, timing/performance
  - arbitrary (called Byzantine)
    - corrupt message, wrong method called, wrong result

Omission and arbitrary failures

Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send*, but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

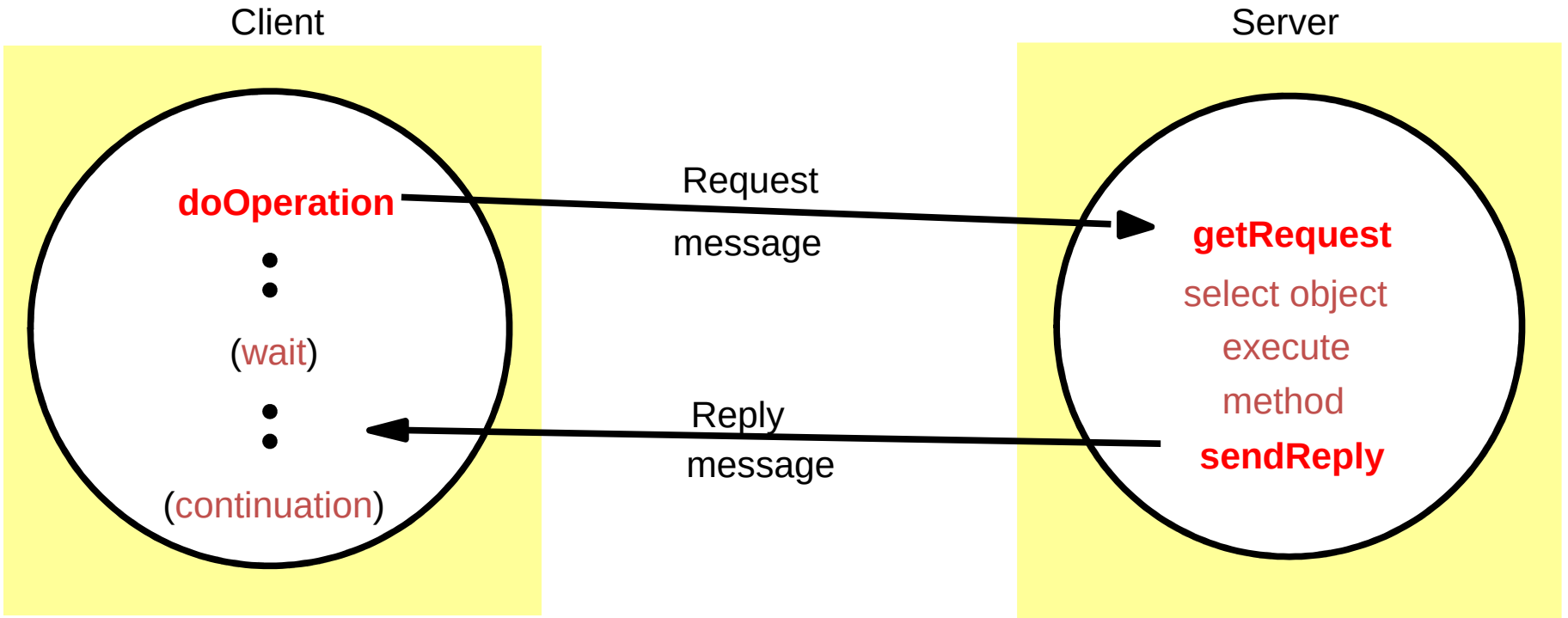
Timing

- **failure can be caused because of timing:**
- No global time
- Computer clocks
  - may have varying drift rate
  - rely on GPS radio signals (not always reliable), or synchronise via clock synchronisation algorithms
- Event ordering (message sending, arrival)
  - carry timestamps
  - may arrive in wrong order due to transmission delays (cf email)

Types of interaction

- Synchronous interaction model:
  - known upper/lower bounds on execution speeds, message transmission delays and clock drift rates
  - more difficult to build, conceptually simpler model
- Asynchronous interaction model (more common, cf Internet, more general):
  - arbitrary process execution speeds, message transmission delays and clock drift rates
  - some problems impossible to solve (e.g. agreement)
  - if solution valid for asynchronous then also valid for synchronous.

Request-Reply Communication



Java API for Datagram Comms

- Simple send/receive, with messages possibly lost/out of order
- Class *DatagramPacket*

message (=array of bytes)	message length	Internet addr	port no

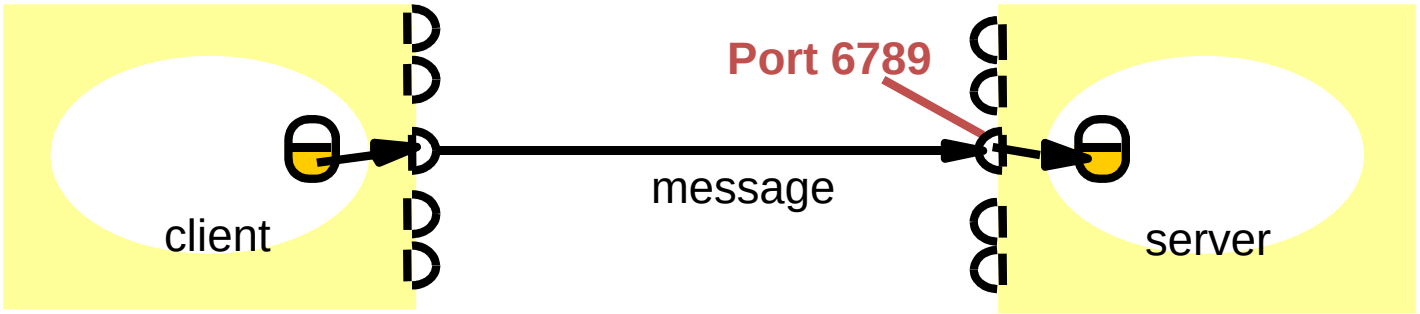
  - packets may be transmitted between sockets
  - packets truncated if too long
  - provides *getData*, *getPort*, *getAddress*

Java API for Datagram Comms

- Class *DatagramSocket*
  - *socket constructor* (returns free port if no arg)
  - *send* a *DatagramPacket*, non-blocking
  - *receive* *DatagramPacket*, blocking
  - *setSoTimeout* (receive blocks for time T and throws *InterruptedIOException*)
  - *close* DatagramSocket
  - throws *SocketException* if port unknown or in use
- See textbook site cdk3.net/ipc for complete code.

In the example...

- UDP Client
  - sends a message and gets a reply
- UDP Server
  - repeatedly receives a request and sends it back to the client



See textbook website for Java code

UDP client example

```
public class UDPClient{
public static void main(String args[]){
// args give message contents and server hostname
 DatagramSocket aSocket = null;
   try {      aSocket = new DatagramSocket();
              byte [] m = args[0].getBytes();
               InetAddress aHost = InetAddress.getByName(args[1]);
               int serverPort = 6789;
               DatagramPacket request = new
                          DatagramPacket(m,args[0].length(),aHost,serverPort);
               aSocket.send(request);
               byte[] buffer = new byte[1000];
               DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
               aSocket.receive(reply);
   }catch (SocketException e){System.out.println("Socket: " + e.getMessage());
  }catch (IOException e){System.out.println("IO: " + e.getMessage());}
} finally {if(aSocket != null)  aSocket.close(); }
}}
```

UDP server example

For further example for TCP Client/Server and explanation of stream communication,

check course book

```
public class UDPServer{
  public static void main(String args[]){
   DatagramSocket aSocket = null;
    try{
      aSocket = new DatagramSocket(6789);
      byte[] buffer = new byte[1000];
      while(true) {
            DatagramPacket request = new DatagramPacket(buffer, buffer.length);
            aSocket.receive(request);
            DatagramPacket reply = new DatagramPacket(request.getData(),
                        request.getLength(), request.getAddress(), request.getPort());
            aSocket.send(reply);
                  }
        }catch (SocketException e){System.out.println("Socket: " + e.getMessage());
       }catch (IOException e) {System.out.println("IO: " + e.getMessage());}
   }finally {if(aSocket != null) aSocket.close();}
}
```

Group Communication:

- Multicast: an operation that sends a single message from one process to each of the members of a
- group of processes
- Fault tolerance based on replicated services
- Finding the discovery servers in spontaneous networking
- Better performance through replicated data
- Propagation of event notifications

IP multicast

- multicast group is specified by a class D Internet address
- membership is dynamic, to join make a socket
- programs using multicast use UDP and send datagrams to mutlicast addresses and (ordinary) port

(For example of Java code see book)

Transport layer protocols

Lecture 16:

Operating Systems and Networks

Behzad Bordbar

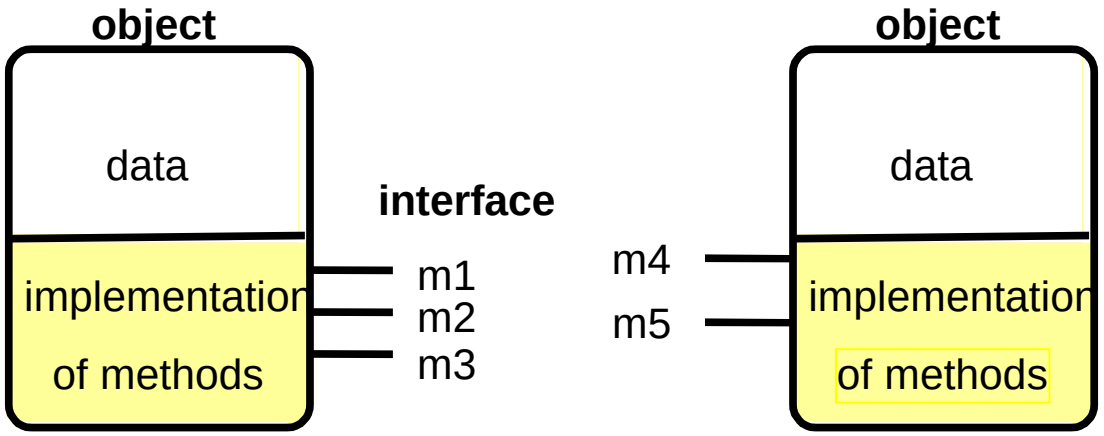
Recap

- Interprocess communication
- ❑ Synchronous and Asynchronous communication
- ❑ use of Socket for comm.
- ❑ various types of failure
- ❑ “no global time”
- ❑ Synchronous and Asynchronous interaction model
- ❑ Java API for UDP ….

Overview

- Distributed applications programming
  - distributed objects model
  - RMI, invocation semantics
  - RPC
  - events and notifications
- Products
  - Java RMI, CORBA, DCOM
  - Sun RPC

Objects

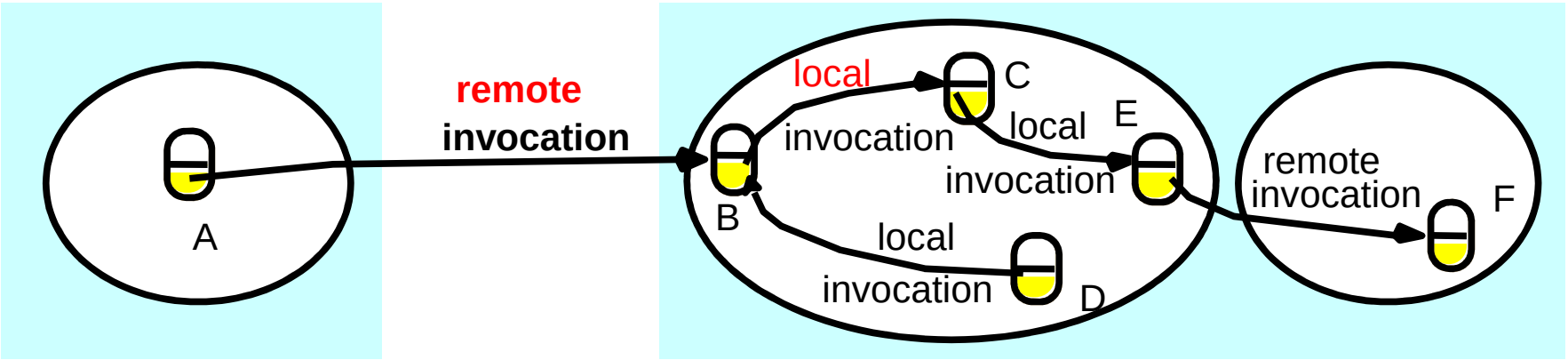


- Objects = Data (attributes) + Operations (methods)
  - encapsulating Data and Methods
  - State of Objects: value of its attributes
- Interact via interfaces:
  - define types of arguments and exceptions of methods

The object (local) model

- Programs:
  - a collection of objects
- Interfaces
  - the only means to access data, make them remote?
- Actions
  - via method invocation
  - interaction, chains of invocations
  - may lead to exceptions, specified in interfaces
- Garbage collection
  - reduced effort, error-free (Java, not C++)

In contrast: distributed object model



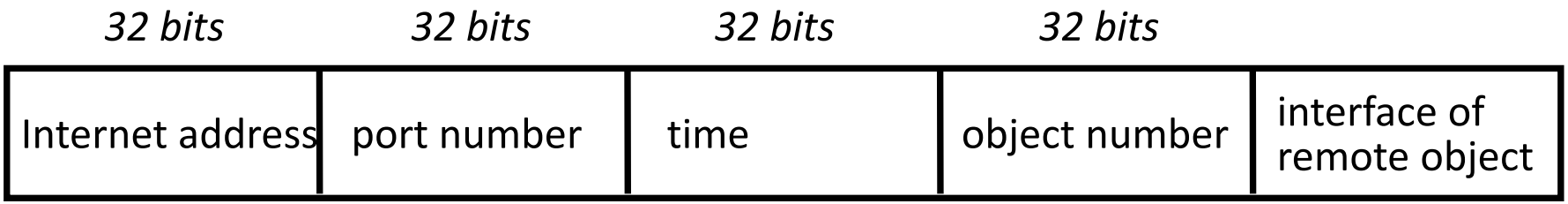
- Objects distributed (client-server models)
- Extend with
  - Remote object reference
  - Remote interfaces
  - Remote Method Invocation (RMI)

Remote object reference

- Object references
  - used to access objects which live in processes
  - can be passed as arguments, stored in variables,...
- Remote object references
  - object identifiers in a distributed system
  - must be unique in space and time
  - error returned if accessing a deleted object
  - can allow relocation (as in CORBA)

Remote object reference

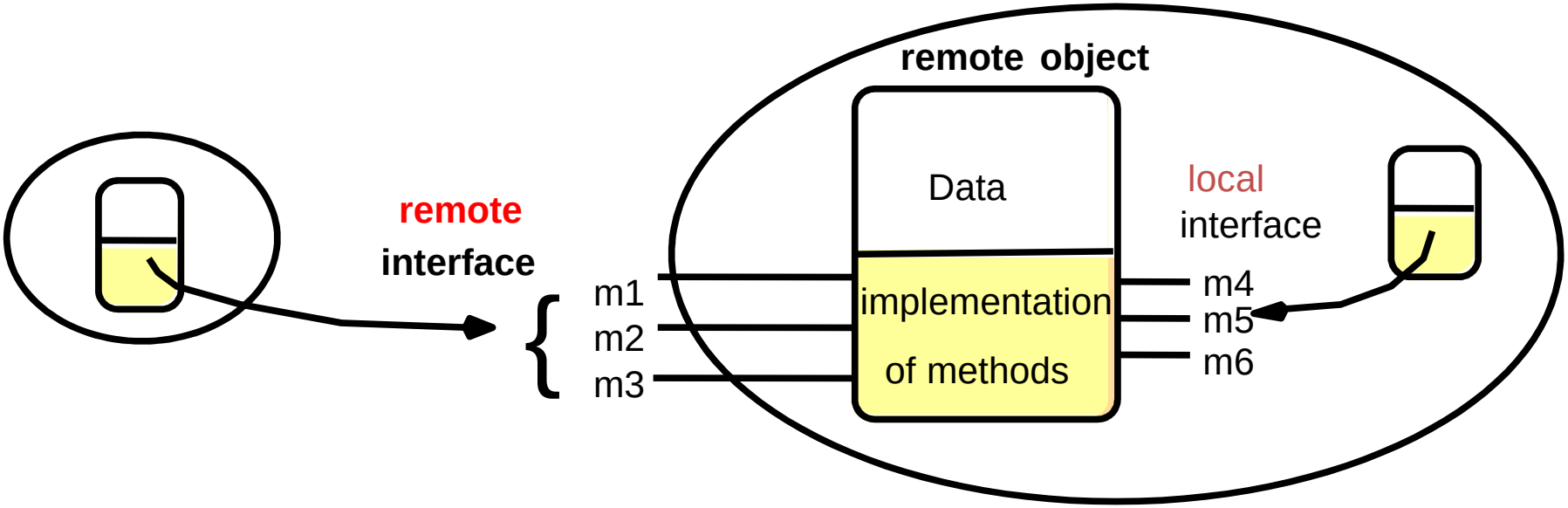
- Constructing unique remote object reference
  - IP address, port, interface name
  - time of creation, local object number (new for each object)
- Use the same as for local object references
- If used as addresses
  - cannot support relocation (alternative in CORBA)



Remote interfaces

- Specify externally accessed
  - variables and procedures
  - no direct references to variables (no global memory)
  - local interface separate
- Parameters
  - input, output or both,
  - instead of call by value, call by reference
- No pointers
- No constructors

Remote object and its interfaces



- CORBA: Interface Definition Language (IDL)
- Java RMI: as other interfaces, keyword *Remote*

Handling remote objects

- Exceptions
  - raised in remote invocation
  - clients need to handle exceptions
  - timeouts in case server crashed or too busy
- Garbage collection
  - distributed garbage collection may be necessary
  - combined local and distributed collector
  - cf Java reference counting

RMI issues

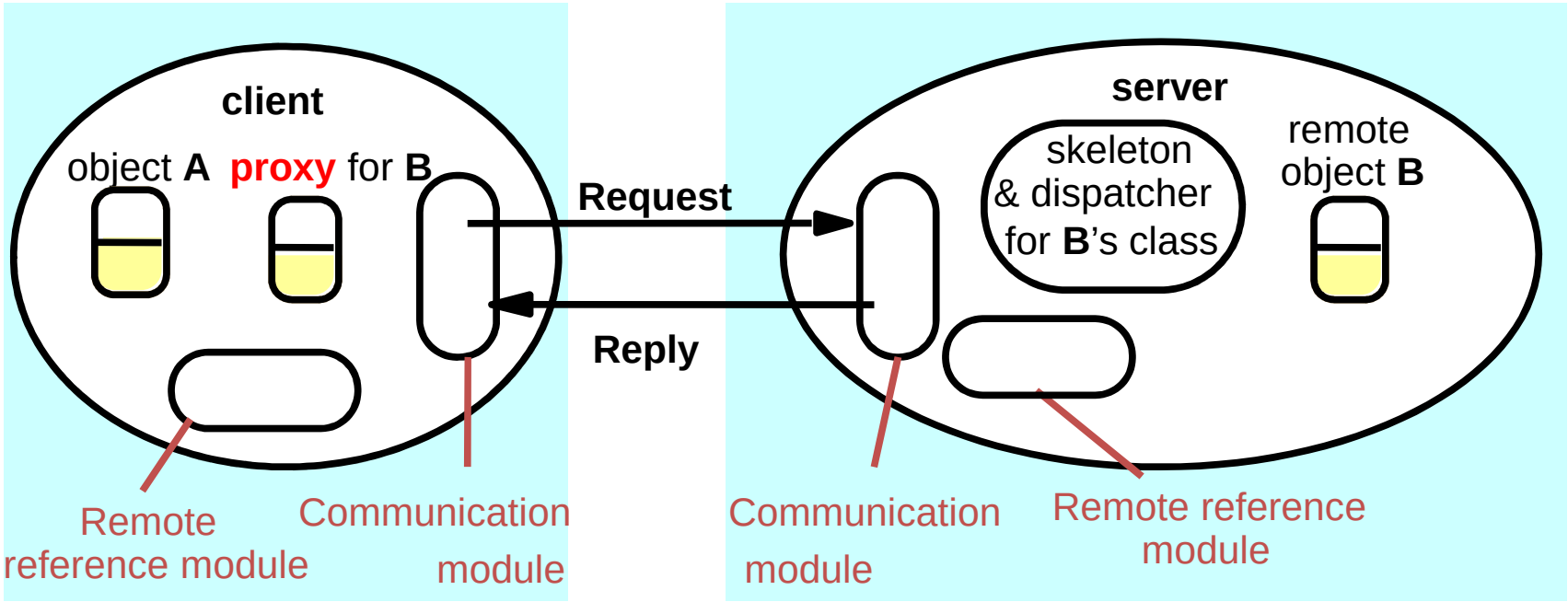
- Local invocations
  - executed exactly once
- Remote invocations
  - via Request-Reply (see *DoOperation*)
  - may suffer from communication failures!
    - ❑retransmission of request/reply
    - ❑message duplication, duplication filtering
  - no unique semantics…

Invocation semantics summary

Fault tolerance measures			*Invocation semantics*
Retransmit request message	*Duplicate filtering*	*Re-execute procedure or retransmit reply*	
No	Not applicable	Not applicable	***Maybe***
Yes	No	Re-execute procedure	***At-least-once***
Yes	Yes	Retransmit reply	***At-most-once***

Re-executing a method sometimes dangerous...

Implementation of RMI



Object A invokes a method in a remote object B:
communication module, remote reference module, RMI software.

Communication modules

- Reside in client and server
- Carry out Request-Reply jointly
  - use unique message ids (new integer for each message)
  - implement given RMI semantics
- Server's communication module
  - selects dispatcher within RMI software
  - converts remote object reference to local

Remote reference module

- Creates remote object references and proxies
- Translates remote to local references (object table):
  - correspondence between remote and local object references (proxies)
- Directs requests to proxy (if exists)
- Called by RMI software
  - when marshalling/unmarshalling

RMI software architecture

- Proxy (for transparency)
  - behaves like local object to client
  - forwards requests to remote object
- Dispatcher
  - receives request
  - selects method (methodID) and passes on request to skeleton
- Skeleton
  - implements methods in remote interface
    - unmarshals data, invokes remote object
    - waits for result, marshals it and returns reply

Binding and activation

- The binder
  - mapping from textual names to remote object references
  - used by clients as a look-up service (cf Java RMIregistry)
- Activation
  - objects active (within running process) and passive (=implementation of methods + marshalled state)
  - activation = create new instance of class + initialise from stored state
- Activator
  - records location of passive and active objects
  - starts server processes and activates objects within them

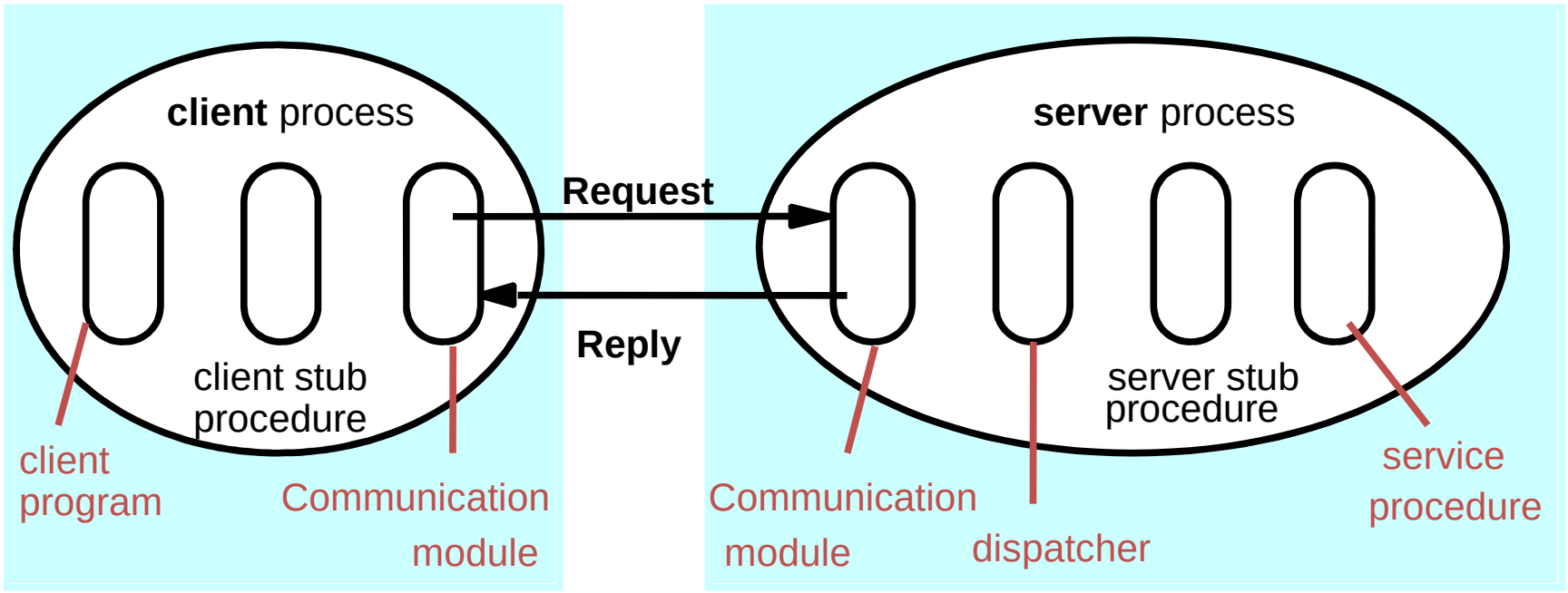
Object location issues

- Persistent object stores
  - stored on disk, state in marshalled form
  - readily available
  - cf Persistent Java
- Object migration
  - need to use remote object reference and address
- Location service
  - assists in locating objects
  - maps remote object references to probable locations

Remote Procedure Call (RPC)

- RPC
  - historically first, now little used
  - over Request-Reply protocol
  - usually at-least-once or at-most-once semantics
  - can be seen as a restricted form of RMI
  - cf Sun RPC
- RPC software architecture
  - similar to RMI (communication, dispatcher and stub in place of proxy/skeleton)

RPC client and server



Implemented over Request-Reply protocol.

Summary

- Distributed object model
  - capabilities for handling remote objects (remote references, etc)
  - RMI: maybe, at-least-once, at-most-once semantics
  - RMI implementation, software architecture
- Other distributed programming paradigms
  - RPC, restricted form of RMI, less often used

Further reading: chapter 5

Lecture 17: Time

- ***Operating Systems and Networks***
- Behzad Bordbar

Recap

- How a computer work?
  - CPU, Kernel, system call, ….
  - hands on: shell programming
  - how two process on the same machine interact? (to do useful things)
- interprocess communication across machines?
  - IP (IP Address, mask , subnet, IPv)
  - UDP, TCP
  - sockets… finally answered? RMI (semantics) and RPC
- Lots of things are different when going from local to remote communication
  - there is no global time!

Overview

- Time service
  - requirements and problems
  - sources of time
- Clock synchronisation algorithms
  - clock skew & drift
  - Cristian algorithm
  - Berkeley algorithm
  - Network Time Protocol
- Logical clocks
  - Lamport's timestamps

Time service

- Why needed?
  - to measure delays between distributed components
  - to synchronise streams, e.g. sound and video
  - to establish event ordering
    - causal ordering (did A happen before B?)
    - concurrent/overlapping execution (no causal relationship)
  - for accurate timestamps to identify/authenticate
    - business transactions
    - serializability in distributed databases
    - security protocols

Clocks

- Internal hardware clock
  - built-in electronic device
  - counts oscillations occurring in a quartz crystal at a definite frequency
  - store the result in a counter register
  - interrupt generated at regular intervals
  - interrupt handler reads the counter register, scales it to convert to time units (seconds, nanoseconds) and updates software clock

Clock skew and drift

- Clock skew
  - difference between the readings of two clocks
- Clock drift
  - difference in reading between a clock and a nominal perfect reference clock per unit of time of the reference clock
    - typically 10^{-6} seconds/second = 1 sec in 11.6 days

Sources of time

- Universal Coordinated Time (UTC, from French)
  - based on atomic time but leap seconds inserted to keep in phase with astronomical time (Earth's orbit)
  - UTC signals broadcast every second from radio and satellite stations
    - ❑land station accuracy 0.1-10ms due to atmospheric conditions
- Global Positioning System (GPS)
  - broadcasts UTC
- Receivers for UTC and GPS
  - available commercially
  - used to synchronise local clocks

Clock synchronisation

- External: synchronise with authoritative source of time
  - the absolute value of difference between the clock and the source is bounded above by D at every point in the synchronisation interval
  - time accurate to within D
- Internal: synchronise clocks with each other
  - the absolute value of difference between the clocks is bounded above by D at every point in the synchronisation interval
  - clocks agree to within D (not necessarily accurate time)

Clock compensation

- Assume 2 clocks can each drift at rate R msecs/sec
  - maximum difference 2R msecs/sec
  - must resynchronise every D/2R to agree within D
- Clock correction
  - get UTC and correct software clock
- Problems!
  - what happens if local clock is 5 secs fast and it is set right?
  - timestamped versions of files get confused
  - time must never run backwards!
  - better to scale the value of internal clock in software without changing the clock rate

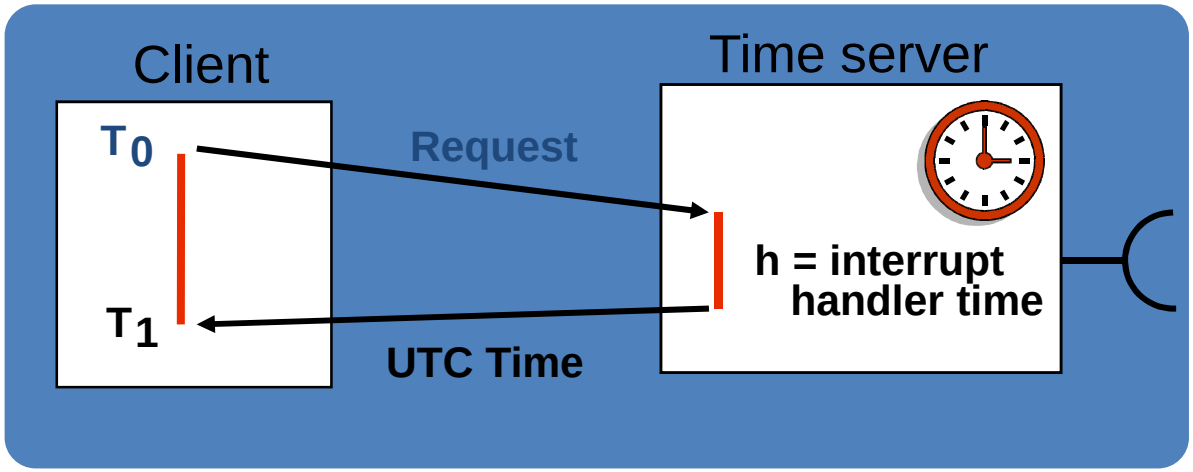
Synchronisation methods

- Synchronous systems
  - simpler, relies on known time bounds on system actions
- Asynchronous systems
  - intranets
    - Cristian's algorithm
    - Berkeley algorithm
  - Internet
    - The Network Time Protocol

Synchronous systems case

- Internal synchronisation between two processes
  - know bounds MIN, MAX on message delay
  - also on clock drift, execution rate
- Assume One sends message to Two with time t
  - Two can set its clock to t + (MAX+MIN)/2 (estimate of time taken to send message)
  - then the skew is at most (MAX-MIN)/2
  - why not t + MIN or t + MAX?
    - ❑maximum skew is larger, could be MAX-MIN

Cristian's algorithm



Time Server with UTC receiver gives accurate current time

Estimate message propagation time by $p=(T_1-T_0-h)/2$ (=half of round-trip of request-reply)

Set clock to UTC+p

Make multiple requests, at spaced out intervals, measure T_1-T_0

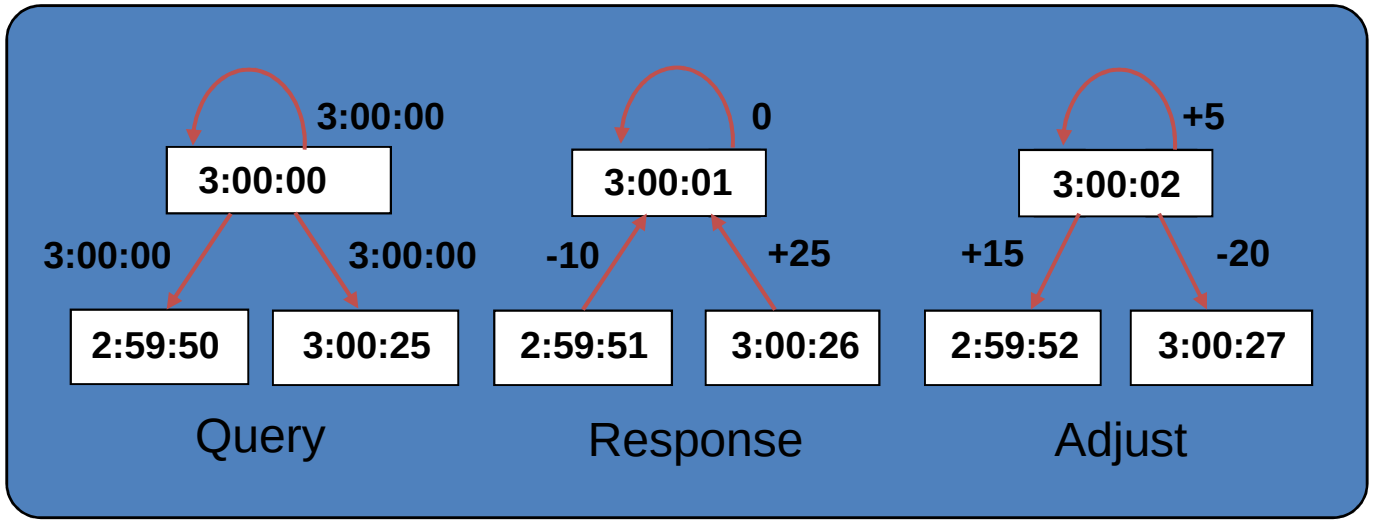
- but discard any that are over a threshold (could be congestion)
- or take minimum values as the most accurate

Cristian's algorithm

- Probabilistic behaviour
  - achieves synchronisation only if round-trip short compared to required accuracy
  - high accuracy only for message transmission time close to minimum
- Problems
  - single point of failure and bottleneck
  - could multicast to a group of servers, each with UTC
  - an impostor or faulty server can wreak havoc
    - use authentication
    - agreement protocol for N > 3f clocks, f number of faulty clocks

The Berkeley algorithm

- Choose master co-ordinator which periodically polls slaves
- Master estimates slaves' local time based on round-trip
- Calculates average time of all, ignoring readings with exceptionally large propagation delay or clocks out of synch
- Sends message to each slave indicating clock adjustment



Synchronisation feasible to within 20-25 msec for 15 computers, with drift rate of 2×10^{-5} and max round trip propagation time of 10 msec.

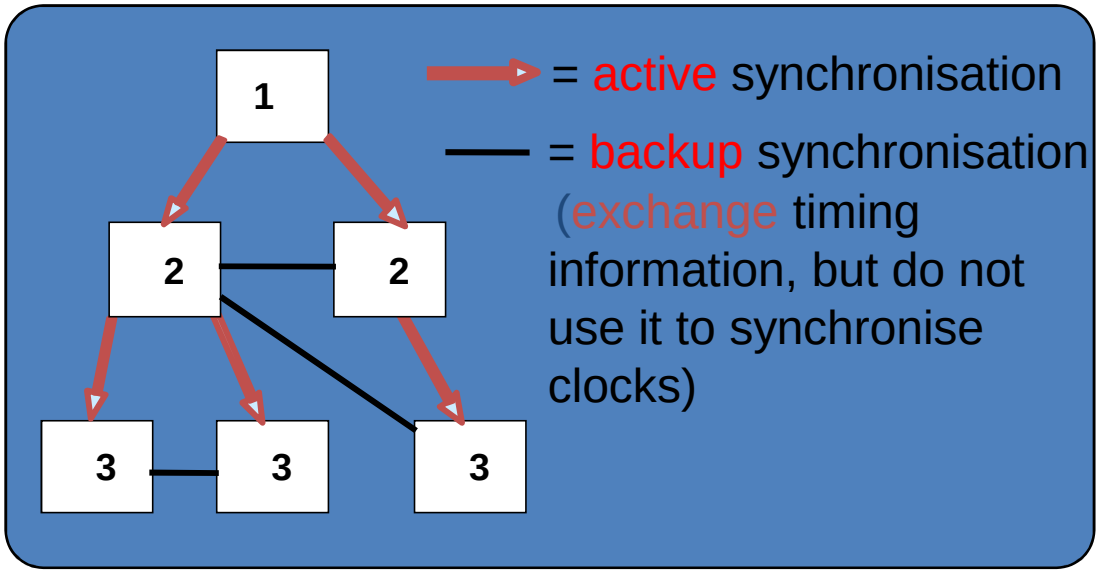
The Berkeley algorithm

- Accuracy
  - depends on the round-trip time
- Fault-tolerant average:
  - eliminates readings of faulty clocks - probabilistically
  - average over the subset of clocks that differ by up to a specified amount
- What if master fails?
  - elect another leader

  How?

Network Time Protocol (NTP)

- Multiple time servers across the Internet
- Primary servers: directly connected to UTC receivers
- Secondary servers: synchronise with primaries
- Tertiary servers: synchronise with secondary, etc
- Scales up to large numbers of servers and clients



Copes with failures of servers – e.g. if primary's UTC source fails it becomes a secondary, or if a secondary cannot reach a primary it finds another one.

Authentication used to check that time comes from trusted sources

NTP Synchronisation Modes

- Multicast
  - one or more servers periodically multicast to other servers on high speed LAN
  - they set clocks assuming small delay
- Procedure Call Mode
  - similar to Cristian's algorithm: client requests time from a few other servers
  - used for higher accuracy or where no multicast
- Symmetric protocol
  - used by master servers on LANs and layers closest to primaries
  - highest accuracy, based on pairwise synchronisation

Logical time

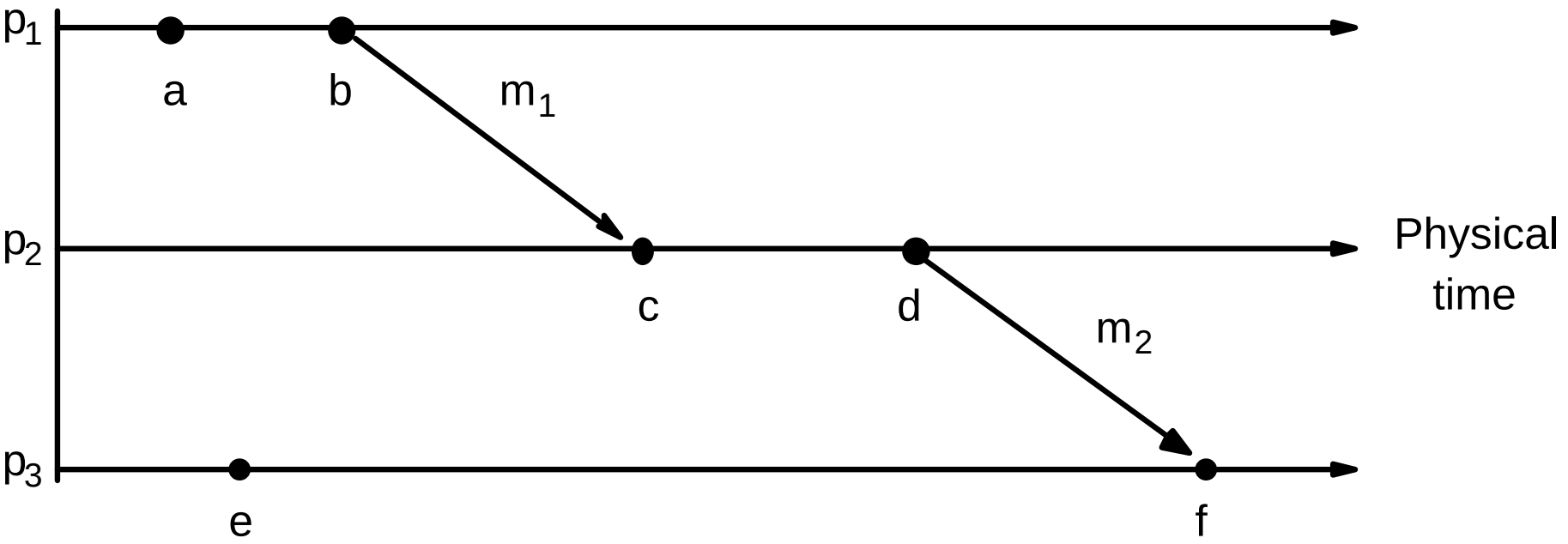
- For many purposes it is sufficient to agree on the same time (e.g. internal consistency) which need not be UTC time
- Can deduce causal event ordering

  $a \rightarrow b$ (a occurs before b)
- Logical time denotes causal relationships
- but the $\rightarrow$ relationship may not reflect real causality, only accidental

Event ordering

- Define a $\rightarrow$ b (a occurs before b) if
  - a and b are events in the same process and a occurs before b, or
  - a is the event of message sent from process A and B is the event of message receipt by process B
- If a $\rightarrow$ b and b $\rightarrow$ c then a $\rightarrow$ c.
- $\rightarrow$ is partial order.
- For events such that neither a $\rightarrow$ b nor b $\rightarrow$ a we say a, b are concurrent, denoted a || b.

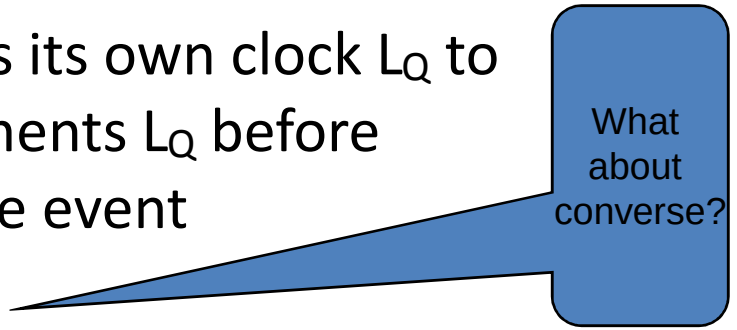
Example of causal ordering



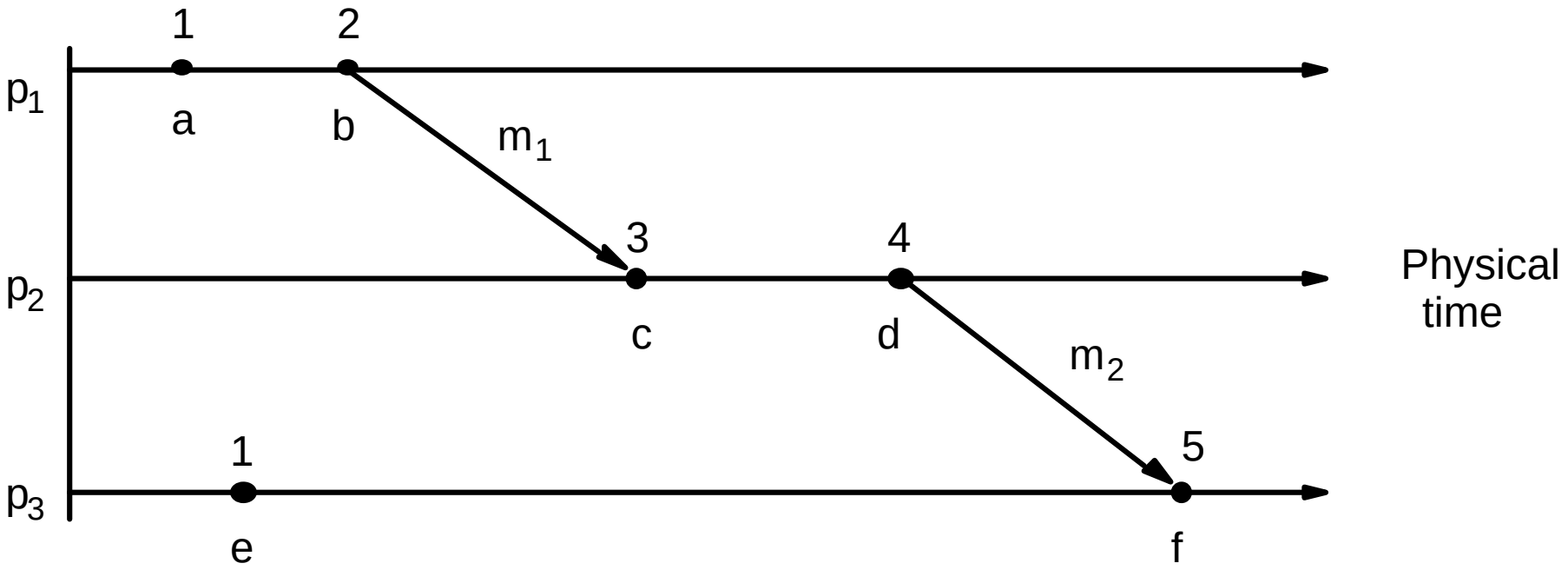
- $a \rightarrow b, c \rightarrow d$
- $b \rightarrow c, d \rightarrow f$
- $a \mid\mid e$

Logical clocks [Lamport]

- Logical clock = monotonically increasing software counter (not real time!)
  - one for each process P, used for timestamping
- How it works
  - L_P incremented before assigning a timestamp to an event
  - when P sends message m, P timestamps it with current value t of L_P (after incrementing it), piggybacking t with m
  - on receiving message (m,t), Q sets its own clock L_Q to maximum of L_Q and t, then increments L_Q before timestamping the message receive event
- Note $a \rightarrow b$ implies $T(a) < T(b)$



Totally ordered logical clocks

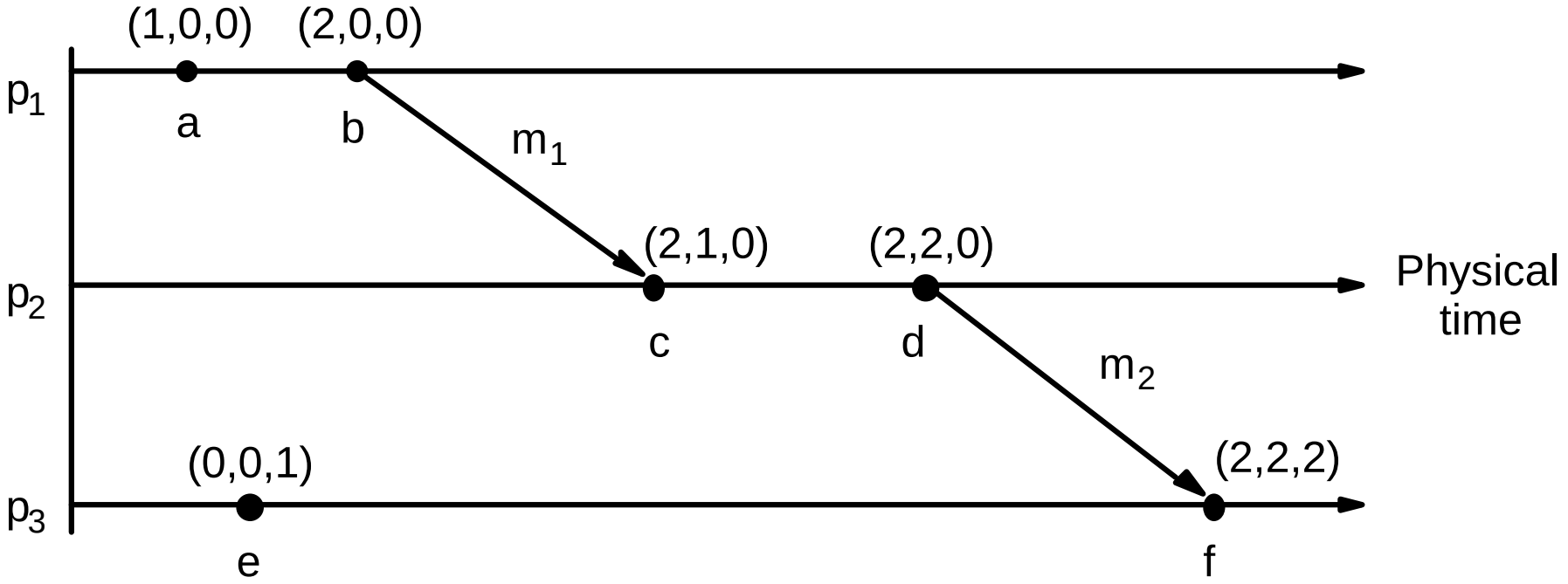


- Problem: T(a) = T(e), and yet a, e distinct.
- Create total order by taking account of process ids.
- Then (T(a),pid) < (T(b),qid) iff T(a) < T(b) or T(a)=T(b) and pid < qid.

Vector clocks

- Totally ordered logical clocks
  - arbitrary event order, depends on order of process ids
  - i.e. (T(a),pid) < (T(b),qid) does not imply a $\rightarrow$ b, see a, e
- Vector clocks
  - array of N logical clocks in each process, if N processes
  - vector timestamps piggybacked on the messages
  - rules for incrementing similar to Lamport's, except
    - processes own component in array modified
    - componentwise maximum and comparison
- Problems
  - storage requirements

Vector timestamps



- VT(b) < VT(c), hence b $\rightarrow$ c
- neither VT(b) < VT(e), nor VT(b) < VT(e), hence b || e

Summary

- Local clocks
  - drift!
  - but needed for timestamping
- Synchronisation algorithms
  - must handle variable message delays
- Clock compensation estimate average delays
  - adjust clocks
  - can deal with faulty clocks
- Logical clocks
  - sufficient for causal ordering

Lecture 18: Security

- ***Operating System and networks***
- Behzad Bordbar

Overview

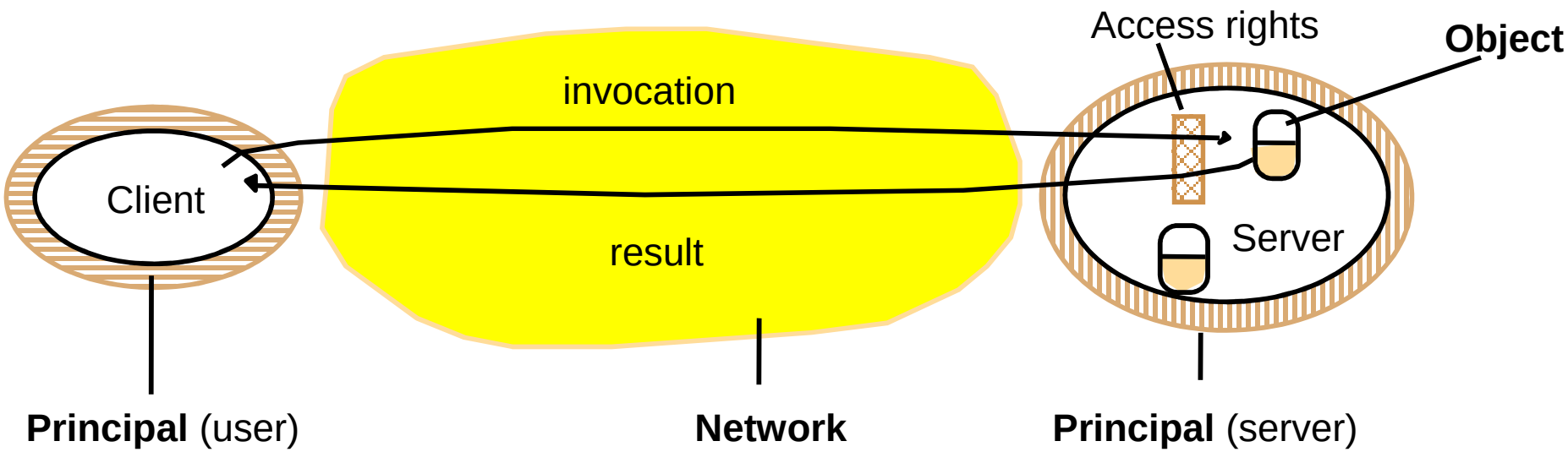
- What is security?
  - policies and mechanisms
  - threats and attacks
- Security of electronic transactions
  - secure channels
  - authentication and cryptography
- Security techniques
  - access control
  - firewalls
  - cryptographic algorithms

Security

- Definition
  - set of measures to guarantee the **privacy, integrity and availability** of resources:
    - objects, databases, servers, processes, channels, etc
  - involves protection of objects and securing processes and communication channels
- Security policies
  - specify who is authorised to access resources (e.g. file ownership)
- Security mechanisms
  - enforce security policy (e.g. file access control)

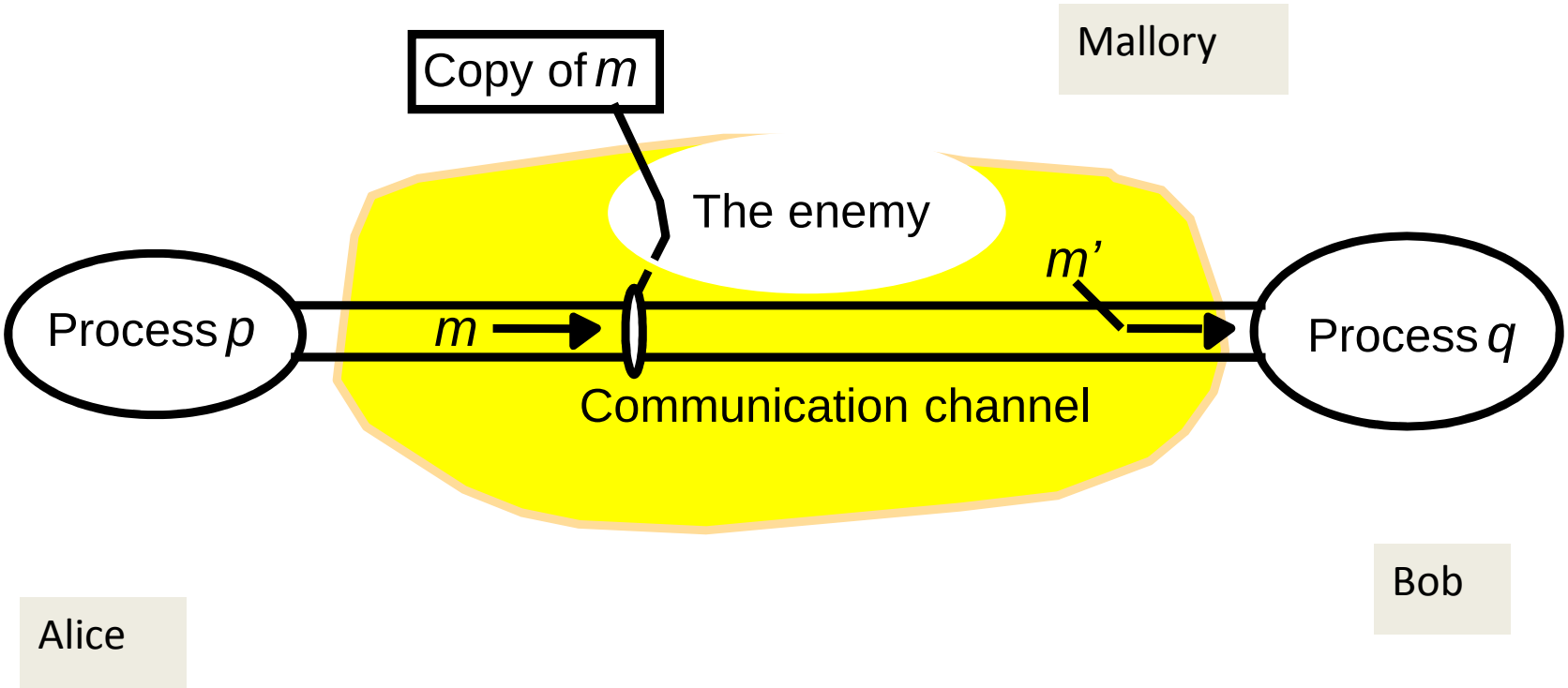
Security model

- Object: intended for use by different clients, via remote invocation
- Principal: authority on whose behalf invocation is issued



The enemy

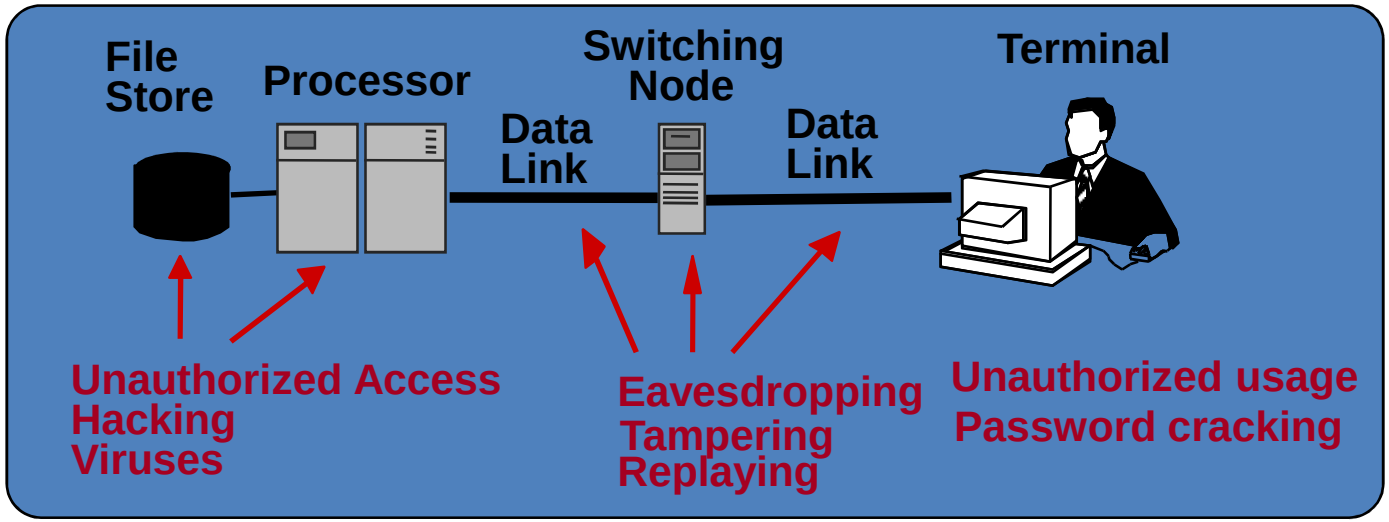
- Processes: encapsulate resources, interact by messages
- Messages: exposed to attack by enemy



Security threats: examples

- Online shopping/banking
  - intercept credit card information
  - purchase goods using stolen credit card details
  - replay bank transaction, e.g. credit an account
- Online stock market information service
  - observe frequency or timing of requests to deduce useful information, e.g. the level of stock
- Website
  - flooding with requests (denial of service)
- My computer
  - receive/download malicious code (virus)

Security threats: what & where



Security threats fall into three categories

Leakage:acquisition of info by unauthorised recipient

Tampering: unauthorised alteration

Vandalism: interference with the property of a system without gain to the perpetrator.

Types of security threats

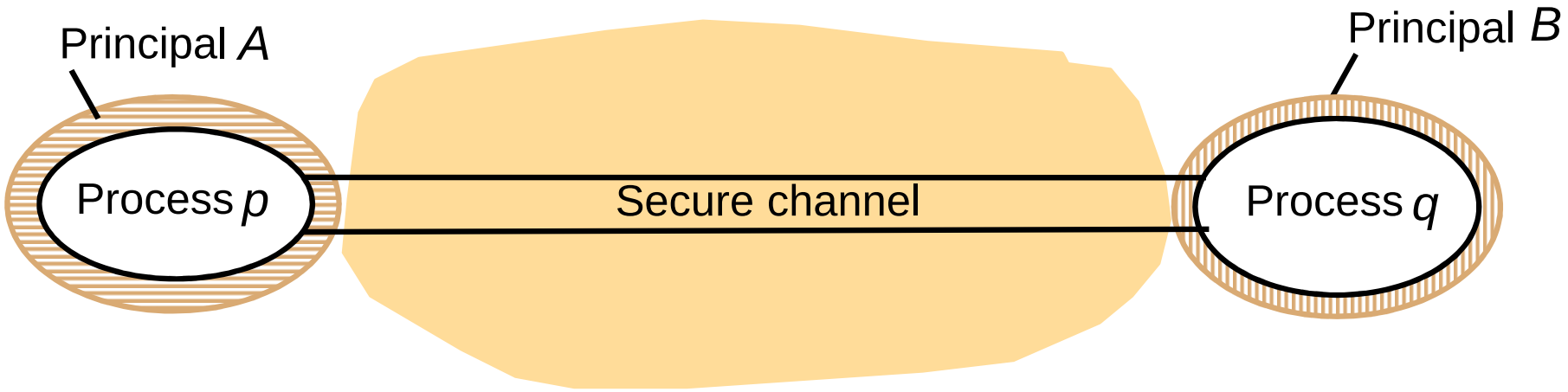
- Eavesdropping
  - obtaining copies of messages without authority
- Masquerading
  - sending/receiving messages using the identity of another principal without their authority
- Message tampering
  - intercepting and altering messages
- Replaying
  - intercepting, storing and replaying messages
- Denial of service
  - flooding a channel with requests to deny access to others

Defeating the enemy: how?

- Encryption (scrambling a message to hide its contents)
  - does not prove identity of sender
- Shared secrets (keys)
  - messages encrypted with the shared key
  - can only be decrypted if the key is known
- Identification (are you who you are?)
  - password protection, etc
- Authentication (are you who you say you are?)
  - include in message identity of principal/data, timestamp
  - encrypt with shared key

Secure channels

- Processes: reliably know identity of principal
- Messages: protected against tampering, timestamped to prevent replaying/reordering.



Threats due to mobility...

- Mobile code (Java JVM)
  - applets, mobile agents (travel collecting information)
  - downloaded from server, run locally
- Security issues: what if the program...
  - illegally writes to a file?
  - writes over another program's memory?
  - crashes?
- Some solutions
  - stored separately from other classes
  - type-checking and code-validation (instruction subset)
  - still does not guard fully against programming errors...

Designing secure systems

- Basic message
  - networks are insecure
  - interfaces are exposed
- Threat analysis
  - assume worst-case scenario
  - list all threats - complex scenarios!!!
- Design guidelines
  - log at points of entry so that violations detected
  - limit the lifetime and scope of each secret
  - publish algorithms, restrict access to shared keys
  - minimise trusted base

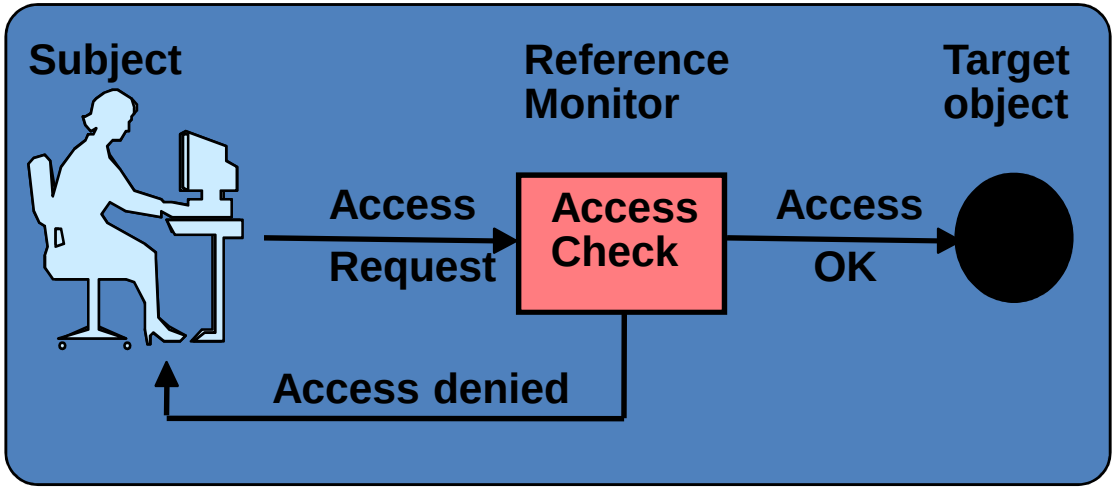
Main security techniques

- Access control
  - implement resource protection, e.g. file protection
  - essential in distributed systems (remote login)
- Firewalls
  - monitor traffic into and out of intranet
- Cryptographic algorithms
  - ciphers
  - authentication
  - digital signatures

Access control

- Definition
  - ensure that users/processes access computer resources in a controlled and authorised manner
- Protection domain
  - is a set of rights for each resource, e.g. Unix files
  - associated with each principal
- Two implementations of protection domains
  - Capabilities
    - request accompanied by key, simple access check
    - open to key theft, or key retained when person left company
  - Access control lists
    - list of rights stored with each resource
    - request requires authentication of principal

Access control



How it works: Reference Monitor

intercepts all access attempts

authenticates request and principal's credentials

applies access control

- ❑ if Yes, access proceeds
- ❑ if No, access is denied, error message returned to the subject

Firewalls

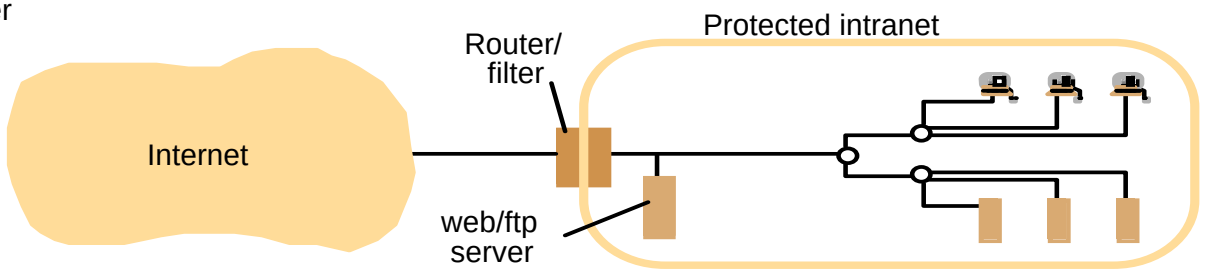
- Monitor and control all communication into and out of an intranet.
- Service control:
  - filter requests for services on internal hosts
  - e.g. reject HTTP request unless to official webserver
- Behaviour control
  - prevent illegal or anti-social behaviour
  - e.g. filter ‘spam’ messages
- User control:
  - allow access to authorised group of users
  - e.g. dial-up services

How does it work...

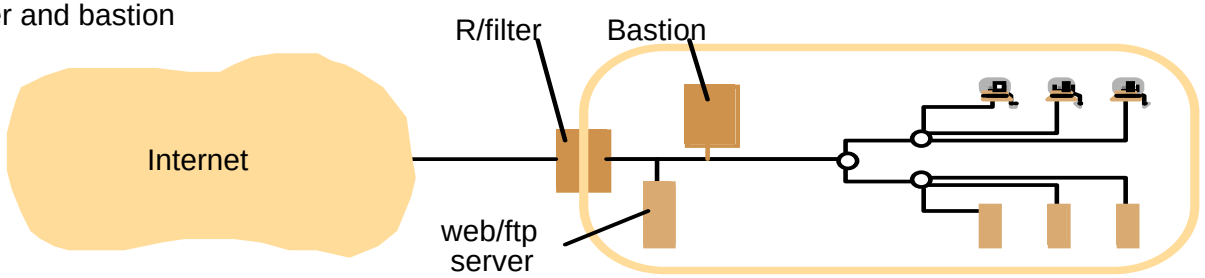
- A set of processes, at different protocol levels:
- IP packet filtering
  - screening of source & destination, only 'clean' packets proceed
  - performed in OS kernel of router
- TCP gateway
  - monitors TCP connection requests
- Application-level gateway
  - runs proxy for an application on TCP gateway, e.g. Telnet
- Bastion
  - separate computer within intranet
  - protected by IP packet filtering, runs TCP/application gateway

Firewall configurations

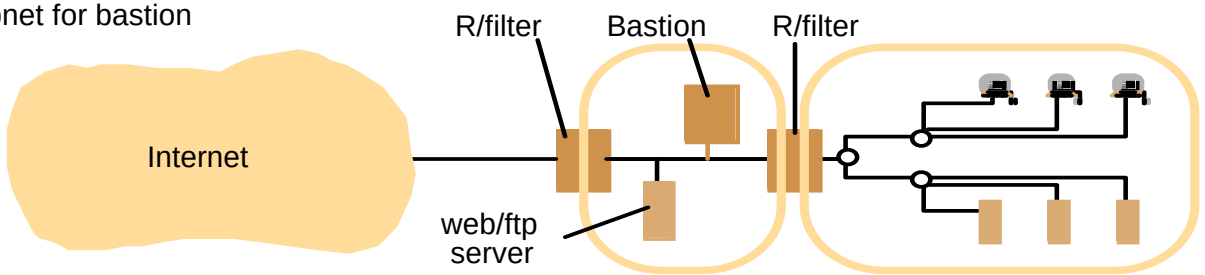
a) Filtering router



b) Filtering router and bastion



c) Screened subnet for bastion



Cryptographic algorithms

- Encryption
  - apply rules to transform *plaintext* to *ciphertext*
  - defined with a function F and key K
  - denote message M encrypted with K by

$$F_K(M) = \{M\}_K$$

- Decryption
  - uses inverse function

$$F^{-1}_K(\{M\}_K) = M$$

  - can be symmetric (based on secret key known to both parties)
  - or asymmetric (based on public key)

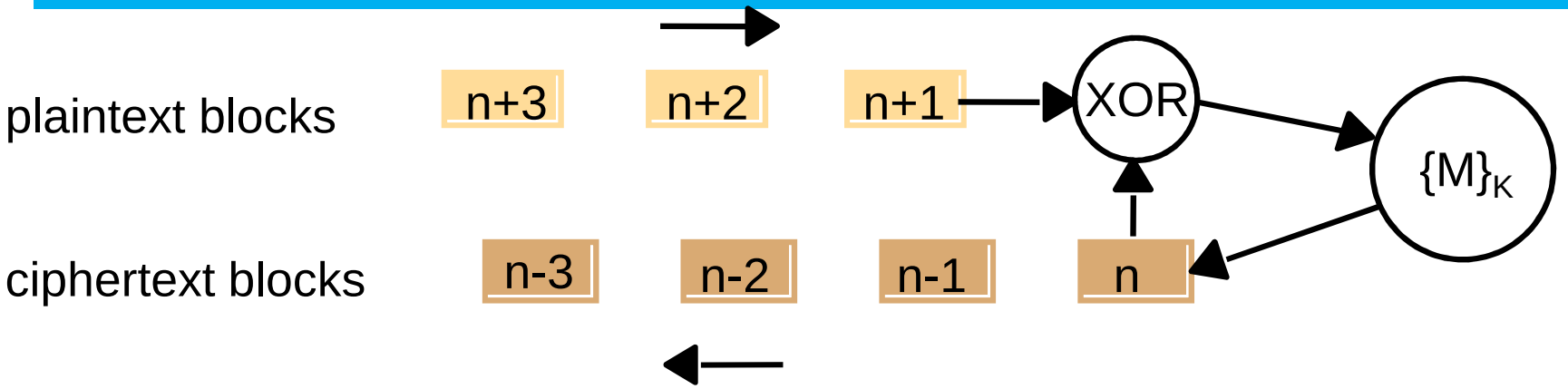
Symmetric cryptography

- One-way functions
  - encryption function F_K easy to compute
  - decryption function F^{-1}_K hard, not feasible in practice
- Idea
  - difficult to discover F_K given $\{M\}_K$
  - difficulty increases with K, to prevent brute-force attack
  - combine blocks of *plaintext* with key through series of XOR, bit shifting, etc, obscuring bit pattern
- Examples
  - several algorithms: TEA, DES
  - secure secret key size 128 bits

Asymmetric cryptography

- Trap-door functions
  - pair of keys (e.g. large numbers)
  - encryption function easy to compute (e.g. multiply keys)
  - decryption function infeasible unless secret known (e.g. factorise the product if one key not known)
- Idea
  - two keys produced: encryption key made public, decryption key kept secret
  - anyone can encrypt messages, only participant with decryption key can operate the trap door
- Examples
  - a few practical schemes: RSA

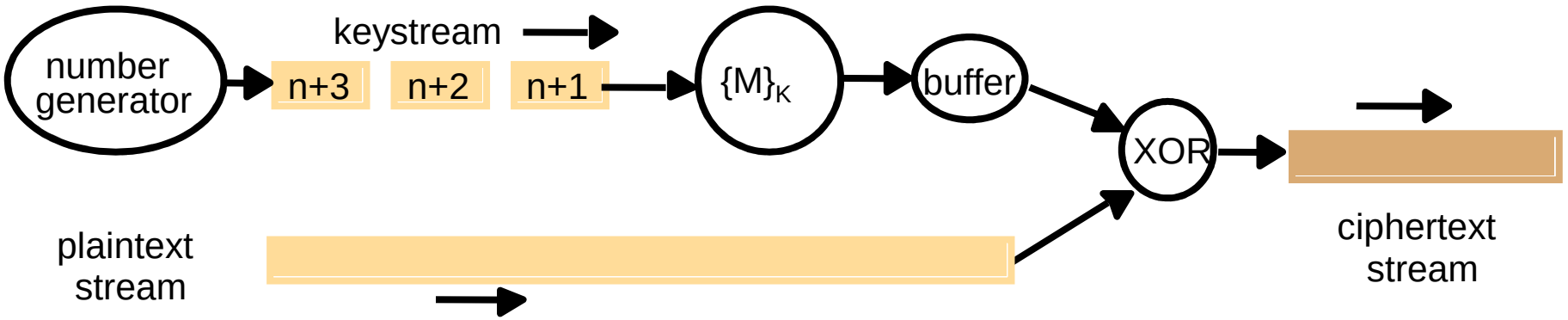
Block ciphers



- How it works
  - message divided into fixed size blocks (64 bits), padded
  - encryption block by block
- Cipher block chaining
  - combine each *plaintext* block with preceding *ciphertext* block using XOR before encryption

Stream ciphers

- How it works
  - used when no block structure (e.g. voice)
  - encryption performed bit by bit
  - generate sequence of numbers, concatenate into secure bit stream



summary

- Security
  - achieves privacy, integrity and availability of distributed systems
- Main techniques
  - access control
  - firewalls
  - cryptography
- Design issues
  - consider worst-case scenario of threats
  - balance cost versus risk
  - log and detect

Lecture 19: cryptographic algorithms

Operating Systems and Networks

Behzad Bordbar

School of Computer Science, University of Birmingham, UK

Overview

- Cryptographic algorithms
  - symmetric: TEA
  - asymmetric: RSA
- Digital signatures
  - digital signatures with public key
  - secure digest function
- Authentication
  - secret-key Needham-Schroeder
  - scenarios

Cryptographic algorithms

- Symmetric (secret key): TEA, DES
  - secret key shared between principals
  - encryption with non-destructive opns (XOR) plus transpose
  - decryption possible only if key known
  - brute force attack (check $\{M\}_K$ for all values of key) hard (exponential in no of bits in key)
- Asymmetric (public key): RSA
  - pair of keys (very large numbers), one public and one private
  - encryption with public key
  - decryption possible only if private key known
  - factorising large numbers (over 150 decimal digits) hard

Tiny Encryption Algorithm(TEA)

- Simple, symmetric (secret key) algorithm
  - written in C [Wheeler & Needham 1994]
- How it works
  - *key* 128 bits (*k[0]..k[3]*)
  - *plaintext* 64 bits (2 x 32 bits, *text[0], text[1]*)
  - in 32 rounds combines *plaintext* and *key,* swapping the two halves of *plaintext*
  - uses reversible addition of unsigned integers, XOR (^) and bitwise shift (<<, >>)
  - combines *plaintext* with constant *delta* to obscure *key*
- Decryption via inverse operations.

TEA Encryption function

```
void encrypt(unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];                          1
    unsigned long delta = 0x9e3779b9, sum = 0; int n;                2
    for (n= 0; n < 32; n++) {                                        3
        sum += delta;                                                4
        y += ((z << 4) + k[0]) ^ (z+sum) ^ ((z >> 5) + k[1]);        5
        z += ((y << 4) + k[2]) ^ (y+sum) ^ ((y >> 5) + k[3]);        6
    }
    text[0] = y;  text[1] = z;                                       7
}
```

TEA Decryption function

```
void decrypt(unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];
    unsigned long delta = 0x9e3779b9, sum = delta << 5;  int n;
    for (n= 0; n < 32; n++) {
        z -= ((y << 4) + k[2]) ^ (y + sum) ^ ((y >> 5) + k[3]);
        y -= ((z << 4) + k[0]) ^ (z + sum) ^ ((z >> 5) + k[1]);
        sum -= delta;
    }
    text[0] = y; text[1] = z;
}
```

Other symmetric algorithms

- TEA
  - simple & concise, yet secure and reasonably fast
- DES (The Data Encryption Standard 1977)
  - US standard for business applications till recently
  - 64 bit plaintext, 56 bit key
  - cracked in 1997 (secret challenge message decrypted)
  - triple-DES (key 112 bits) still secure, poor performance
- AES (Advanced Encryption Standard)
  - invitation for proposals 1997
  - in progress
  - key size 128, 192 and 256 bits

RSA

- Rivest, Shamir and Adelman '78
- How it works
  - relies on N = P × Q (product of two very large primes)
  - factorisation of N hard
  - choose keys e, d such that

    $e \times d = 1 \; mod \; Z$ where $Z = (P-1) \times (Q-1)$
- It turns out...
  - can encrypt M by $M^e \; mod \; N$
  - can decrypt by $C^d \; mod \; N$ (C is encrypted message)
- Thus
  - can freely make e and N public, while retaining d

RSA: past, present and future

- In 1978...
  - Rivest *et al* thought factorising numbers > 10^{200} would take more than four billion years
- Now (ca 2000)
  - faster computers, better methods
  - numbers with 155 (= 500 bits) decimal digits successfully factorised
  - 512 bit keys insecure!
- The future?
  - keys with 230 decimal digits (= 768 bits) recommended
  - 2048 bits used in some applications (e.g. defence)

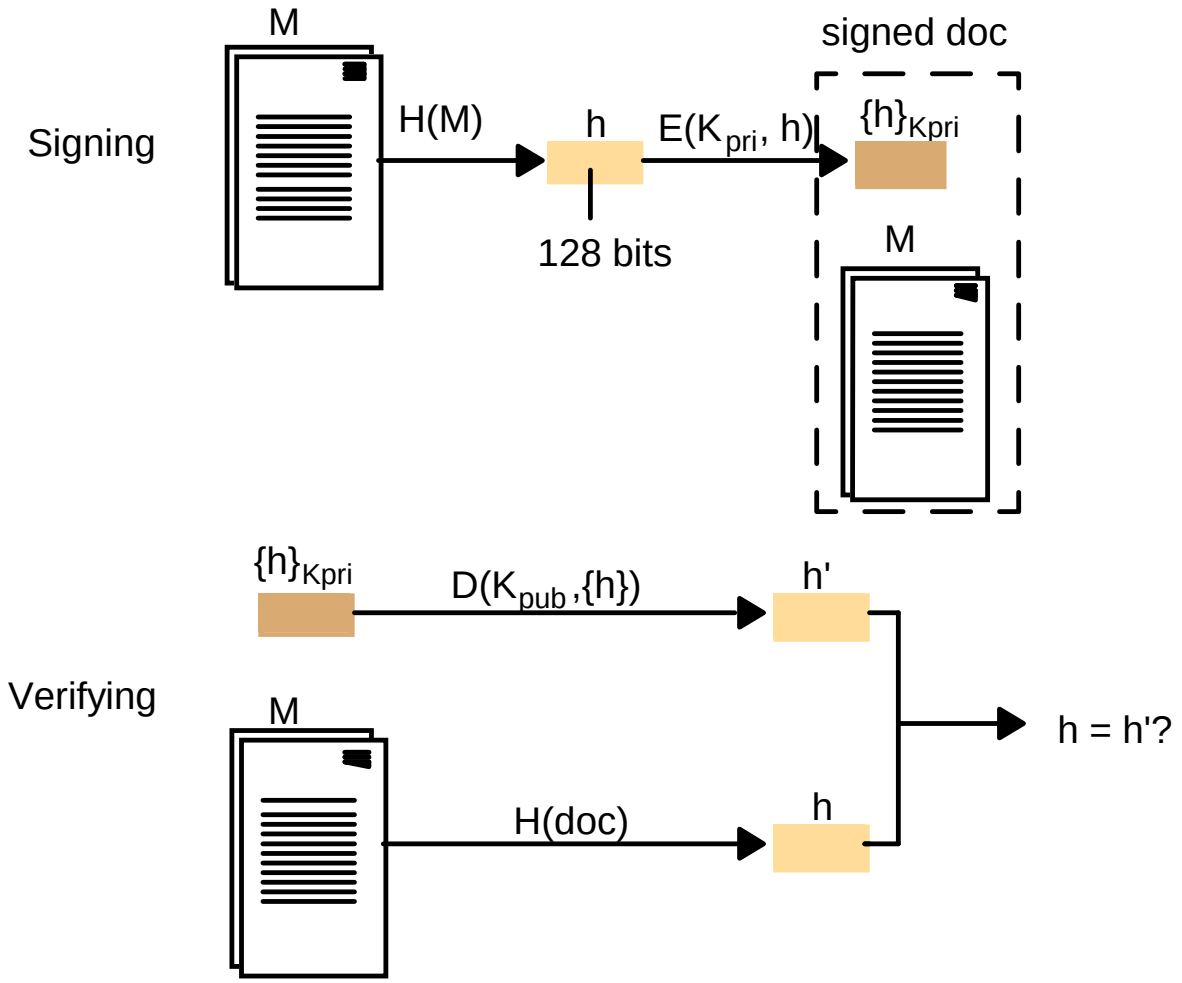
Digital signatures

- Why needed?
  - alternative to handwritten signatures
  - authentic, difficult to forge and undeniable
- How it works
  - relies on secure hash functions which compress a message into a so called *digest*
  - sender encrypts *digest* and appends to message as a signature
  - receiver verifies signature
  - generally public key cryptography used, but secret key also possible

Digital signatures with public key

- Keys
  - sender chooses key pair K_{pub} and K_{pri}; key K_{pub} made public
- Sending signed message M
  - sender uses an agreed secure hash function h to compute *digest* h(M)
  - *digest* h(M) is encrypted with private key K_{pri} to produce signature $S = \{h(M)\}_{Kpri}$; the pair M, S sent
- Verifying signed message M, S
  - when pair M, S received, signature S decrypted using K_{pub}, digest h(M) computed and compared to decrypted signature
- Note
  - RSA can be used, but roles of keys reversed.

Digital signatures with public key



Secure digest functions

- Based on one-way hash functions:
  - given M, easy to compute h(M)
  - given h, hard to compute M
  - given M, hard to find another M' such that h(M) = h(M')
- Note
  - operations need not be information preserving
  - function not reversible
- Example: MD5 [Rivest 1992][
  - 128 bit digest, using non-linear functions applied to segments of source text

Authentication

- Definition
  - protocol for ensuring authenticity of the sender
- Secret-key protocol [Needham & Schroeder '78]
  - based on secure key server that issues secret keys
  - see this lecture and textbook (5 steps)
  - flaw corrected '81
  - implemented in Kerberos
- Public-key protocol [Needham & Schroeder '78]
  - does not require secure key server (7 steps)
  - flaw discovered with CSP/FDR
  - SSL (Secure Sockets Layer) similar to it

Needham-Schroeder secret-key

- Principals
  - client A (initiates request), server B
  - secure server S
- Secure server S
  - maintains table with name + secret key for each principal
  - upon request by client A, issues key for secure communication between client A and server B, transmitted in encrypted form ('ticket')
- Messages
  - labelled by nonces (integer values added to message to indicate freshness)

Needham-Schroeder secret-key

Header	*Message*	*Notes*
1. A->S:	A, B, N_A	A requests S to supply a key for communication with B.
2. S->A:	$\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_B}\}_{K_A}$	S returns a message encrypted in A's secret key, containing a newly generated key K_{AB} and a 'ticket' encrypted in B's secret key. The nonce N_A demonstrates that the message was sent in response to the preceding one. A believes that S sent the message because only S knows A's secret key.
3. A->B:	$\{K_{AB}, A\}_{K_B}$	A sends the 'ticket' to B.
4. B->A:	$\{N_B\}_{K_{AB}}$	B decrypts the ticket and uses the new key K_{AB} to encrypt another nonce N_B.
5. A->B:	$\{N_B - 1\}_{K_{AB}}$	A demonstrates to B that it was the sender of the previous message by returning an agreed transformation of N_B.

Problems!

- In step 3
  - message need not be fresh...
- So...
  - intruder with K_{AB} and $\{K_{AB}, A\}_{KB}$ (left in cache, etc) can initiate exchange with B, impersonating A
  - secret key K_{AB} compromised
- Solution
  - add nonce or timestamp to message 3, yielding

    $\{K_{AB}, A, t\}_{KBpub}$
  - B decrypts message and checks t recent
  - adapted in Kerberos

Summary

- Symmetric encryption
  - DES: most widely used till recently, 56-bit key insecure
  - 3DES, AES or IDEA an alternative
- Asymmetric encryption
  - RSA: 512-bit key insecure, use with 768-bit keys or above
- Authentication with secret-key
  - Kerberos, based on [Needham-Schroeder ‘78]
- Authentication with public-key
  - SSL (Secure Sockets Layer)
  - used in electronic commerce