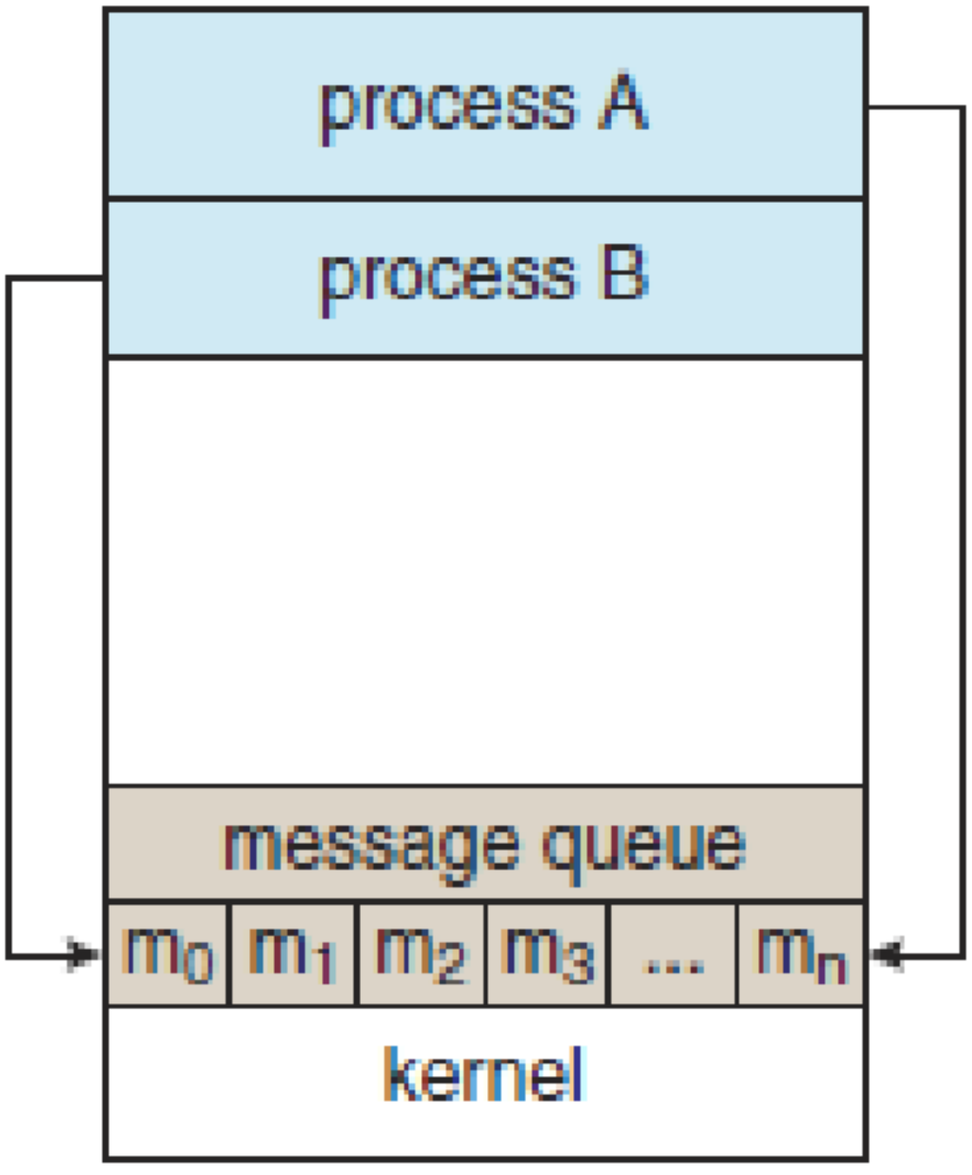
Network and protocols

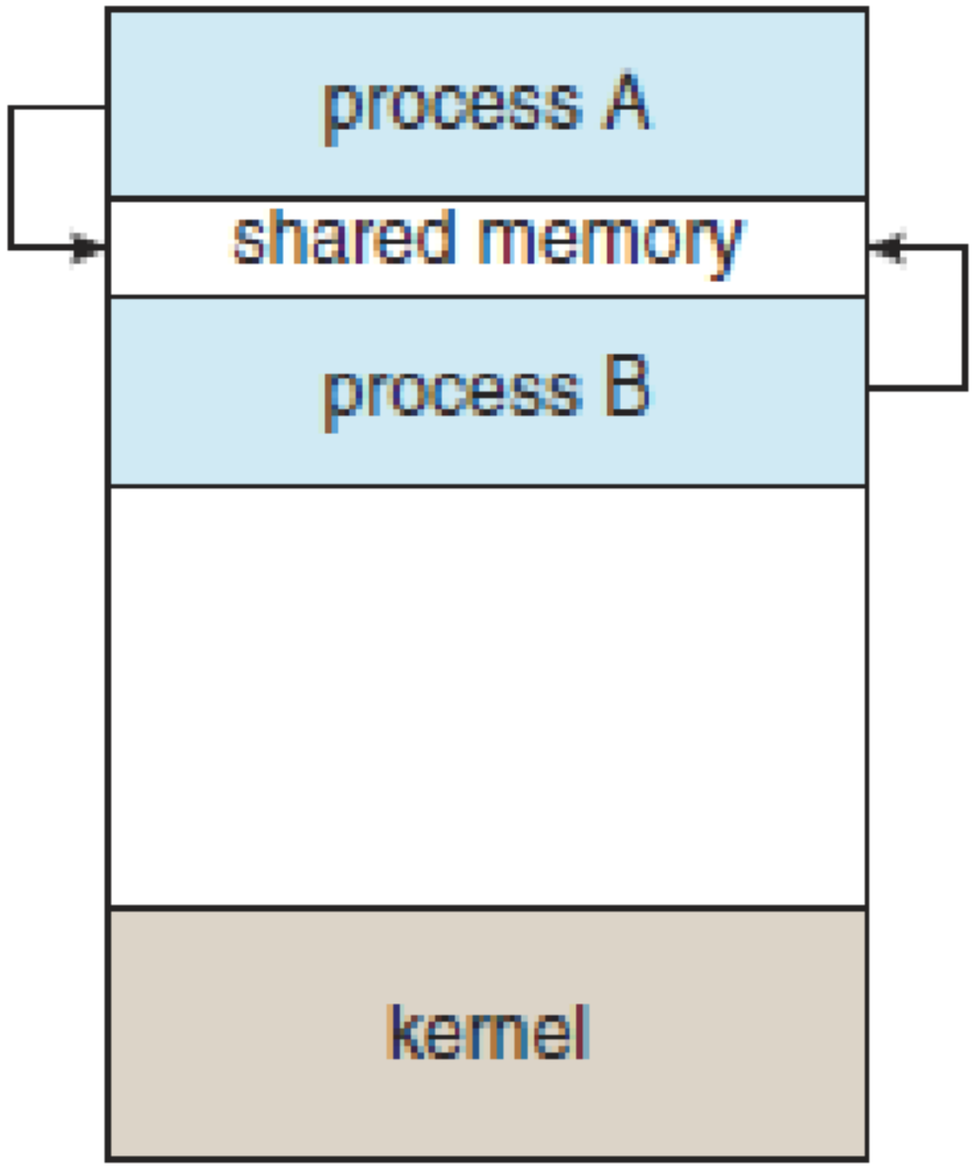
Lecture 09:

Operating Systems and Networks

Behzad Bordbar

processes communicate in two ways





Between machines we need a network or communication medium!

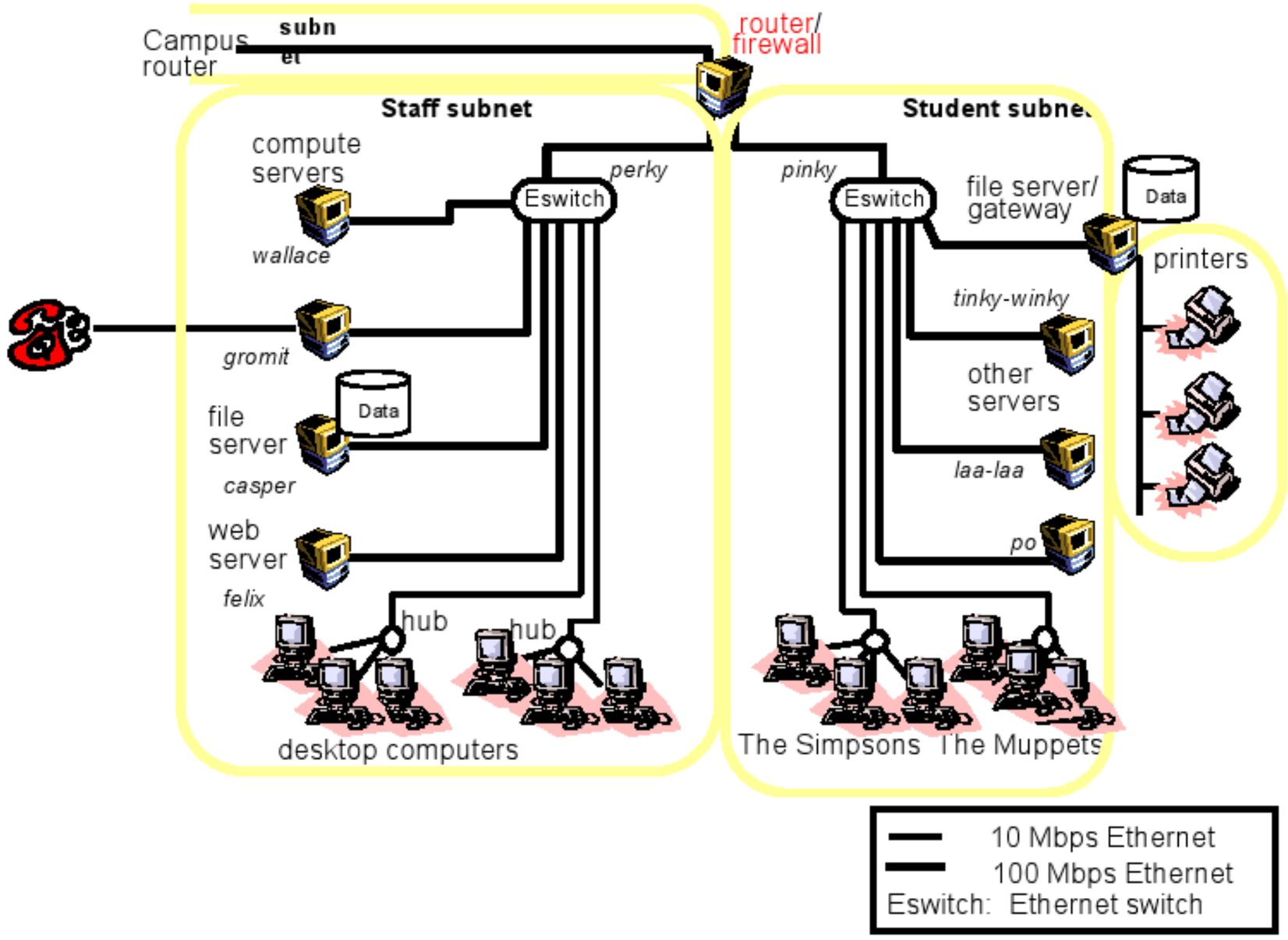
Types of Networks

- 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

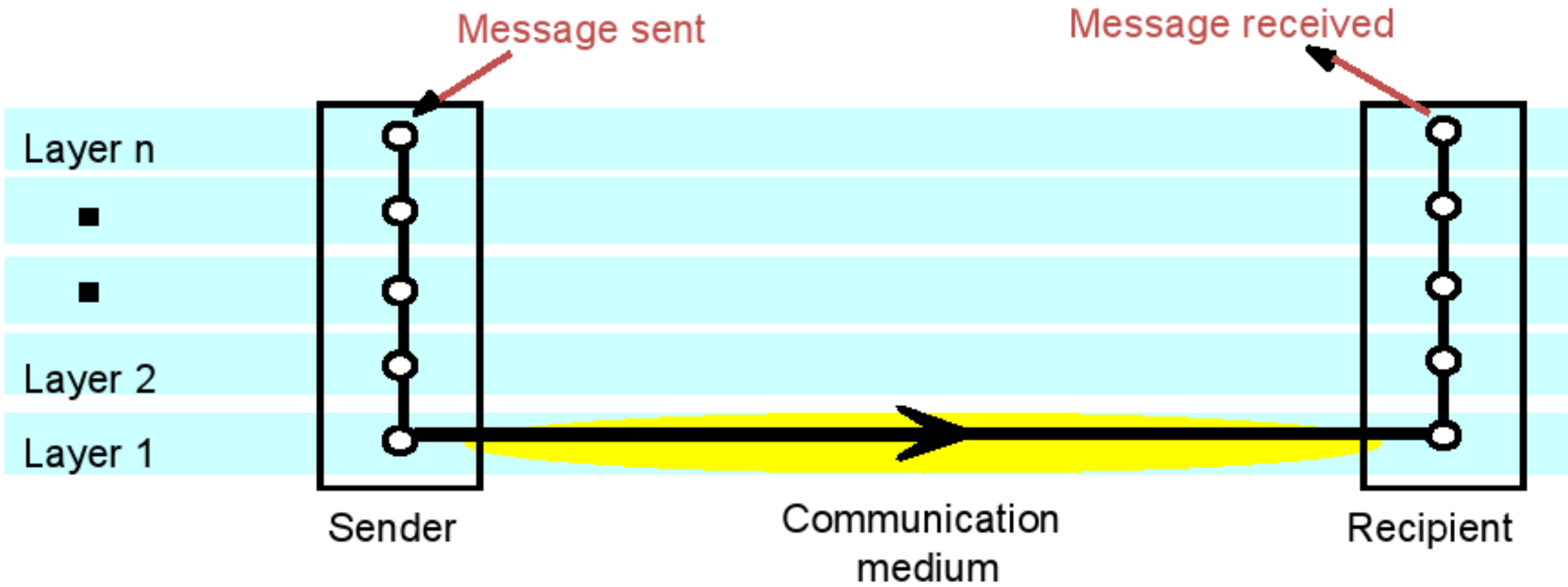
Switching schemes

- 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

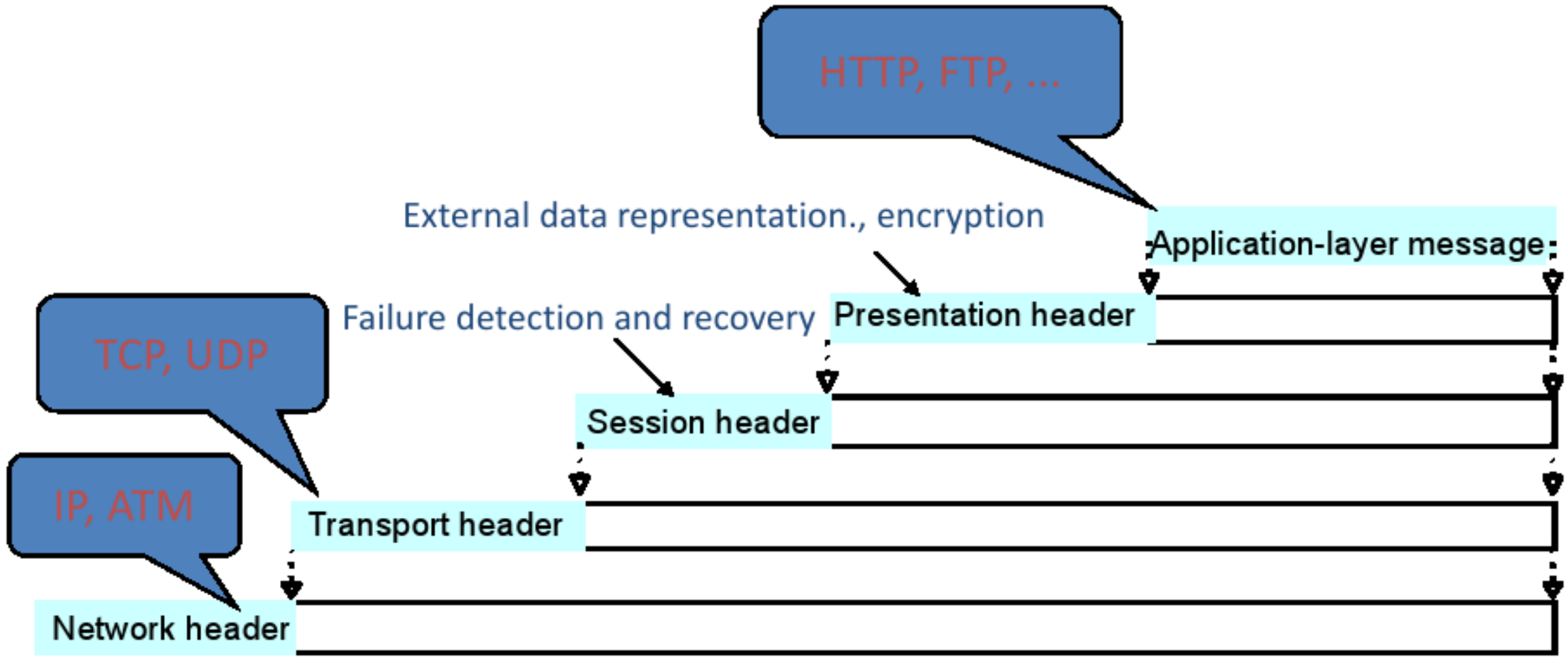
- well-known set of rules and formats to be used for communication between processes to perform a given task
- Two parts:
  - ❑specification of sequence of messagegs that must be exchanged
  - ❑specification of the format of the data in each message

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

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

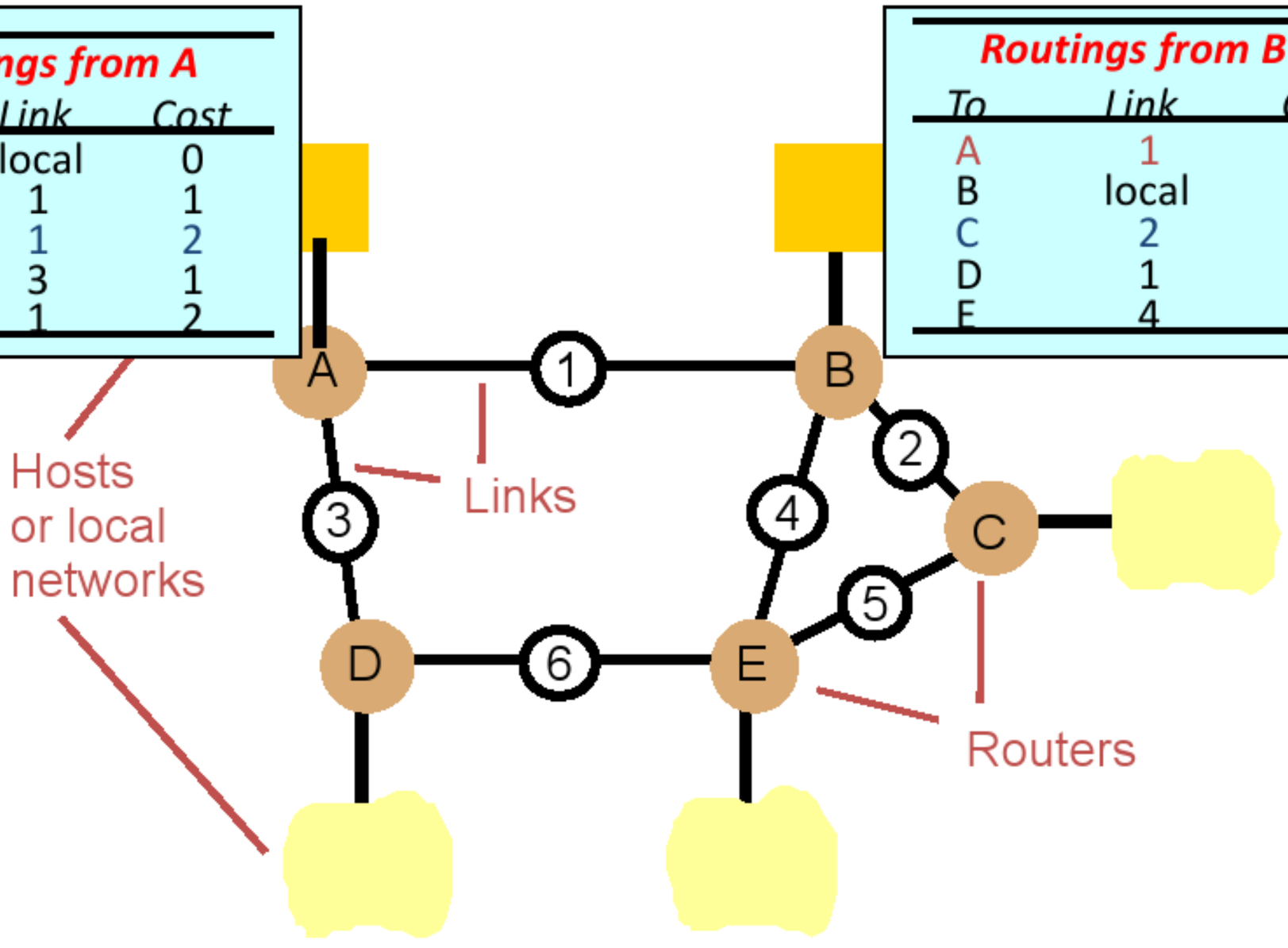
Routing example

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



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

RIP (continue)

- When a link fails *cost* in the table is set to ∞
- Then, the cost in all table is set to ∞ ($1+\infty = \infty$)
- 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)

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

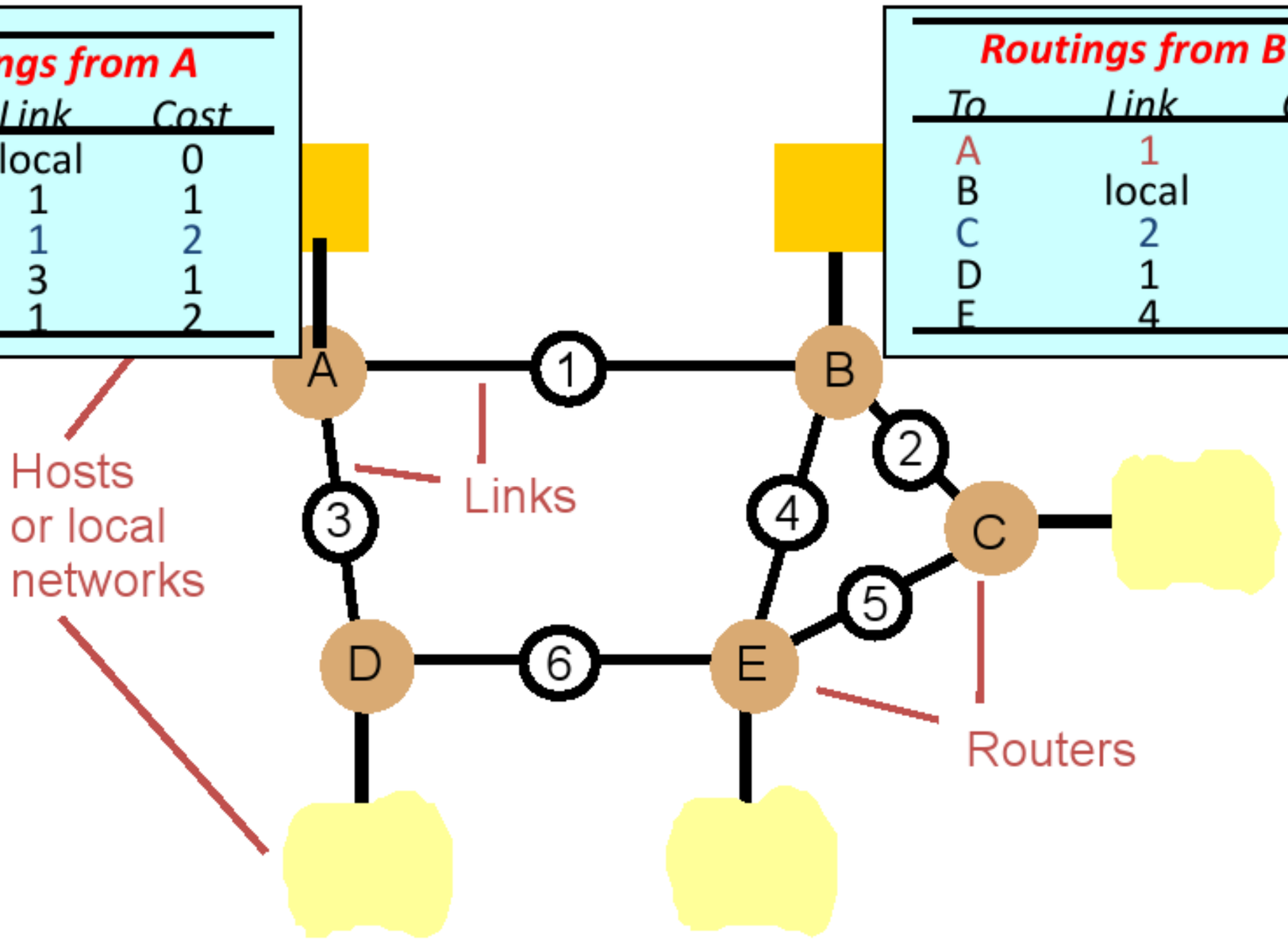
Routing example

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



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

RIP (continue)

- When a link fails *cost* in the table is set to ∞
- Then, the cost in all table is set to ∞ ($1+\infty = \infty$)
- 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)

IP address

Lecture 13:

Operating Systems and Networks

Behzad Bordbar

Contents

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

Contents

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

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?

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

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.

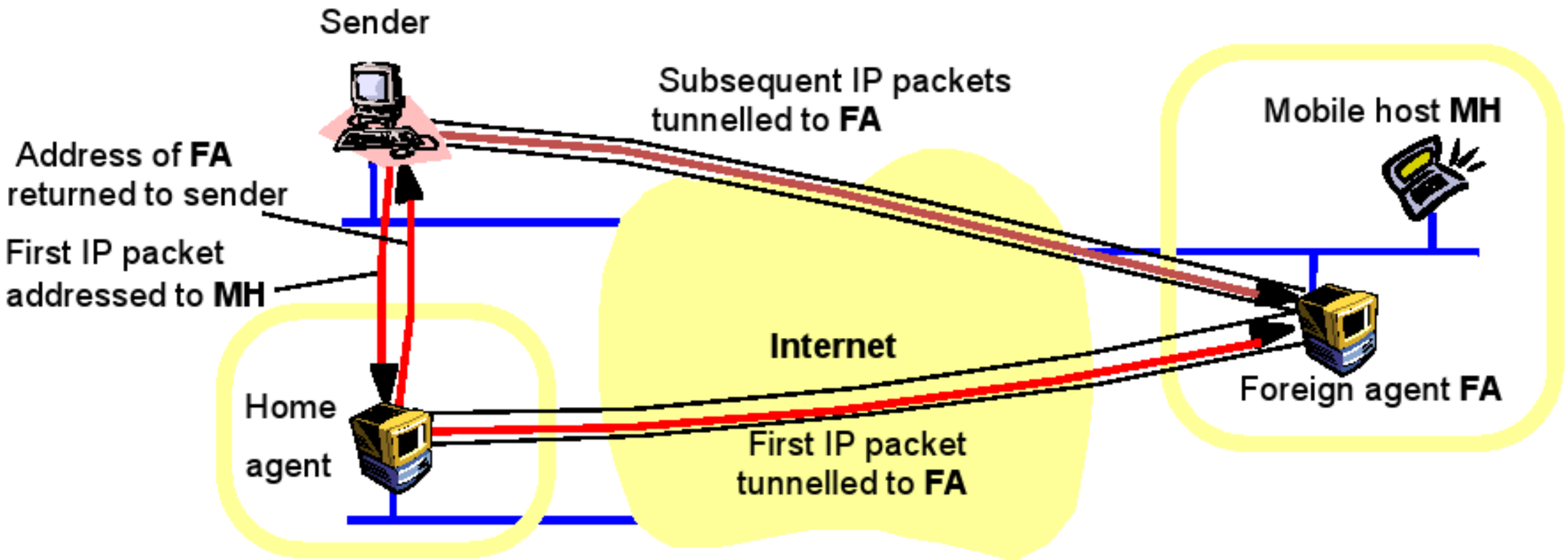
Connection-oriented service

- 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

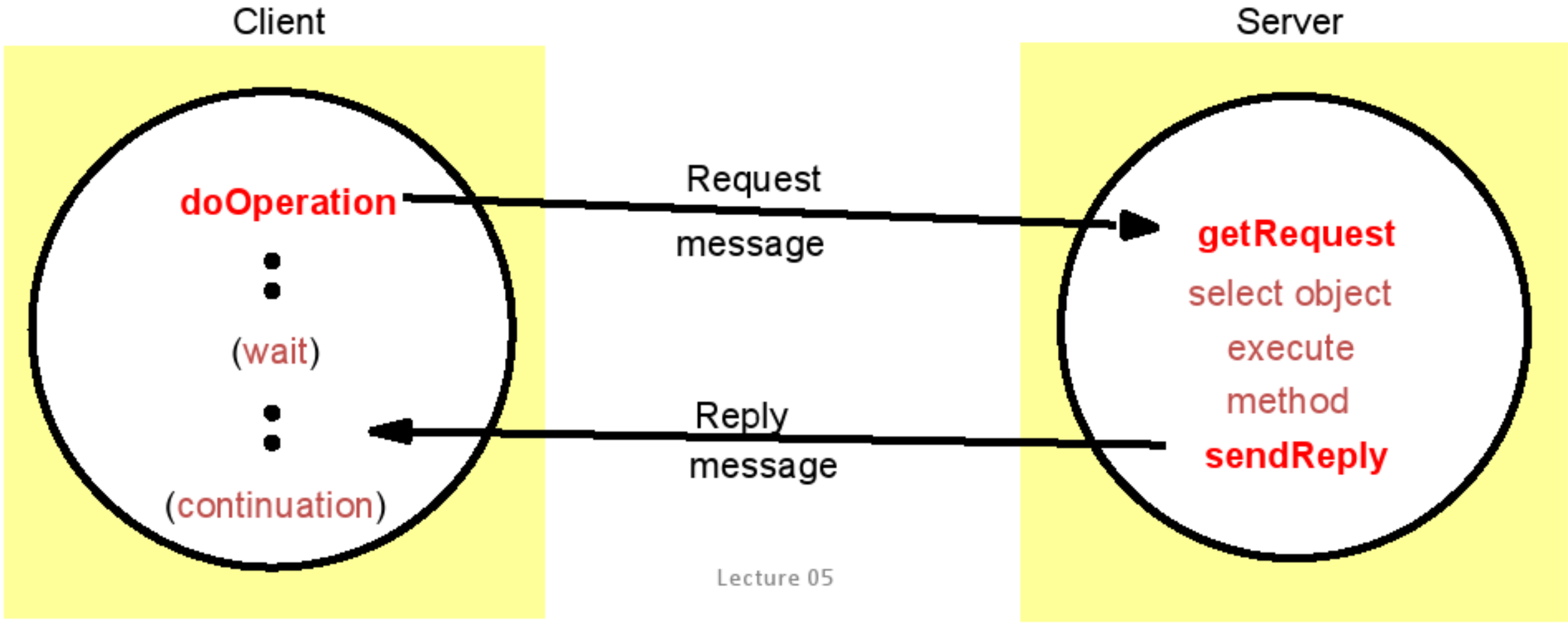


Wireless LAN (802.11)

- 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

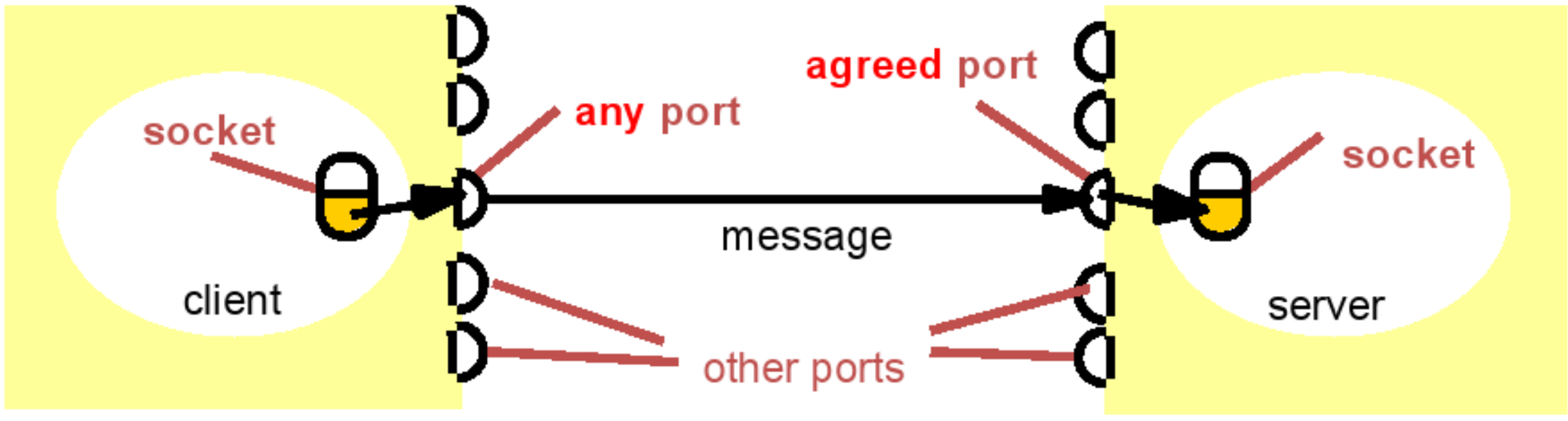
- Synchronize communication: sender and receiver
- synchronise on every message, i.e. *blocking* operations
- Asynchronous

❑ 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

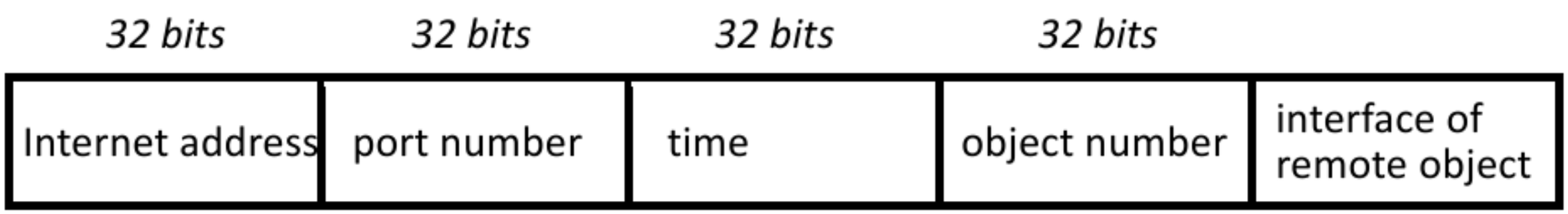
- Class *InetAddress*
  - uses DNS (Domain Name System)

```
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.

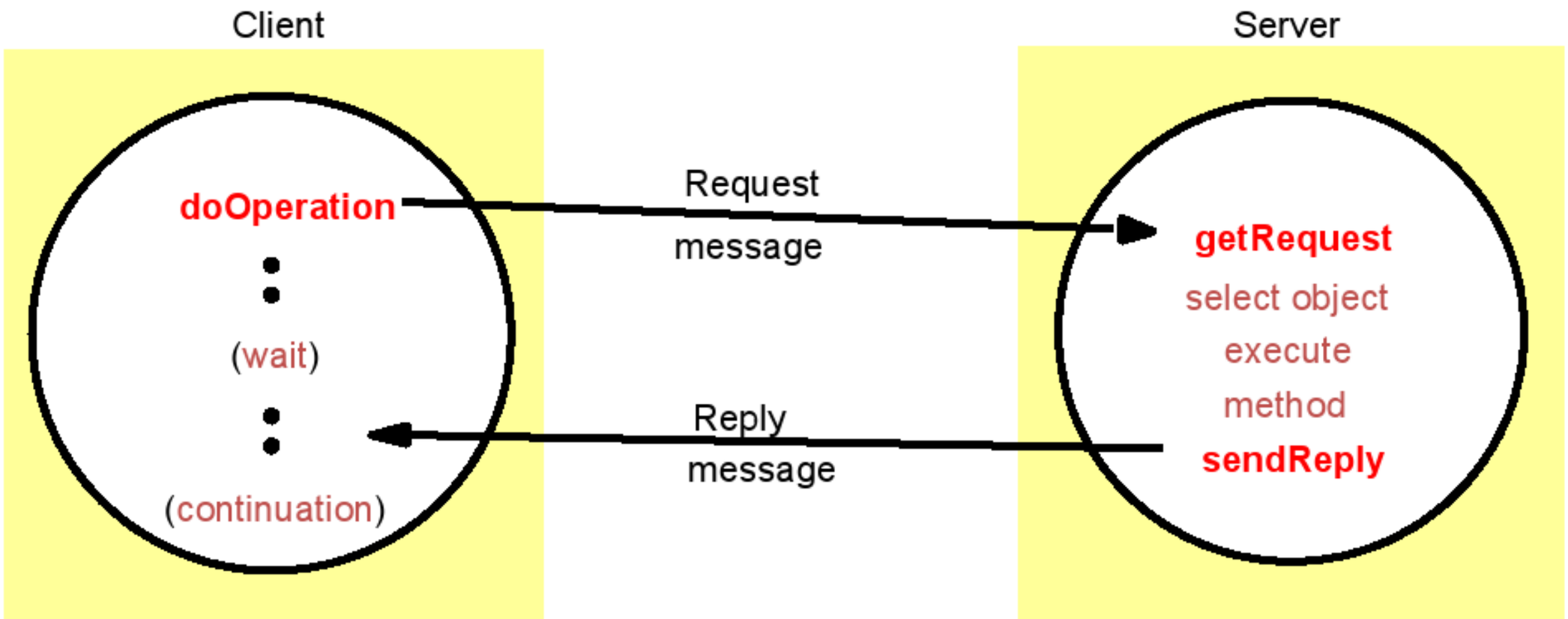
Timing

- **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

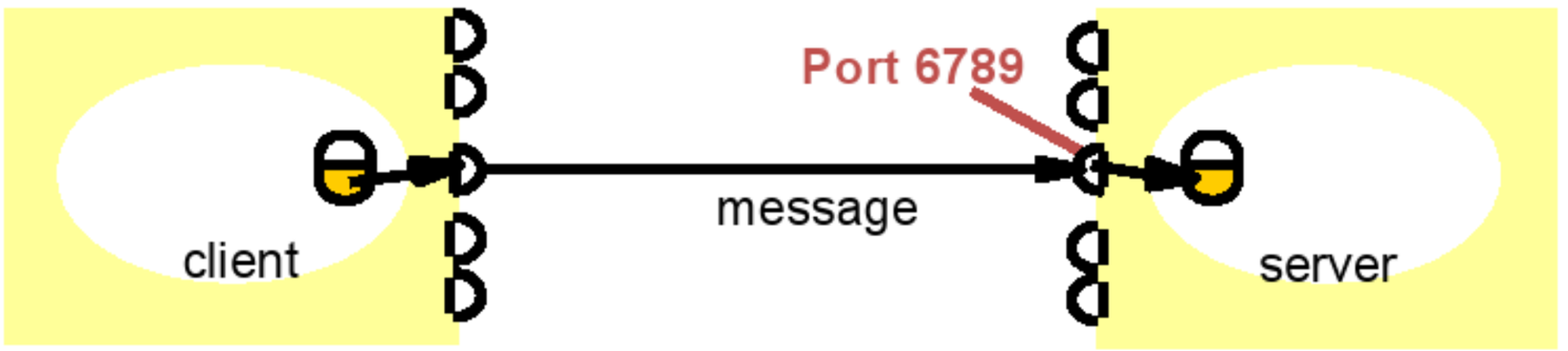
  - packets may be transmitted between sockets
  - packets truncated if too long
  - provides *getData*, *getPort*, *getAddress*

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

```
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();}
}
```

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

check course book

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)

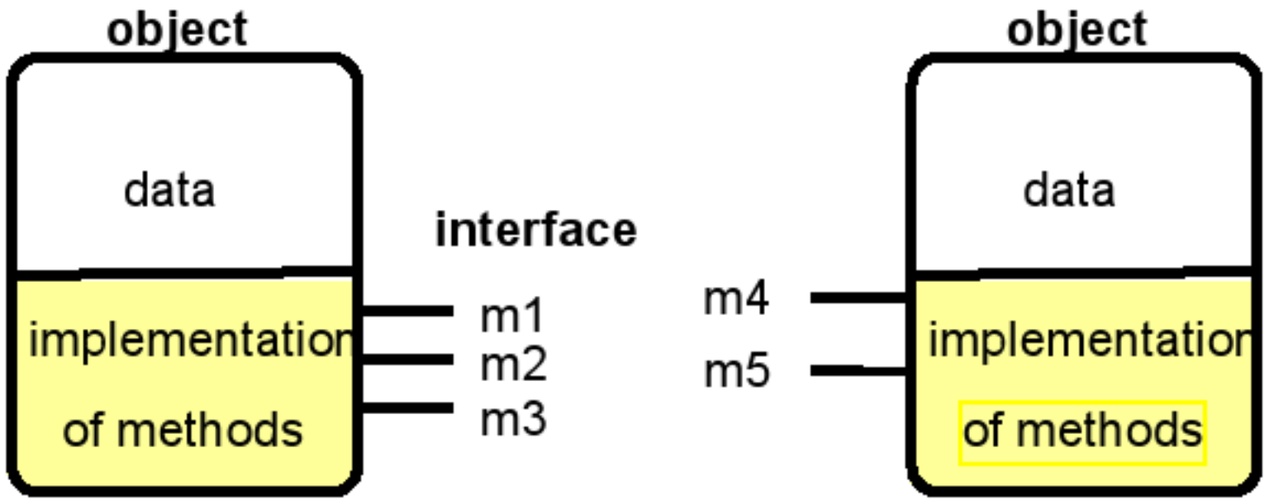
Recap

- 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

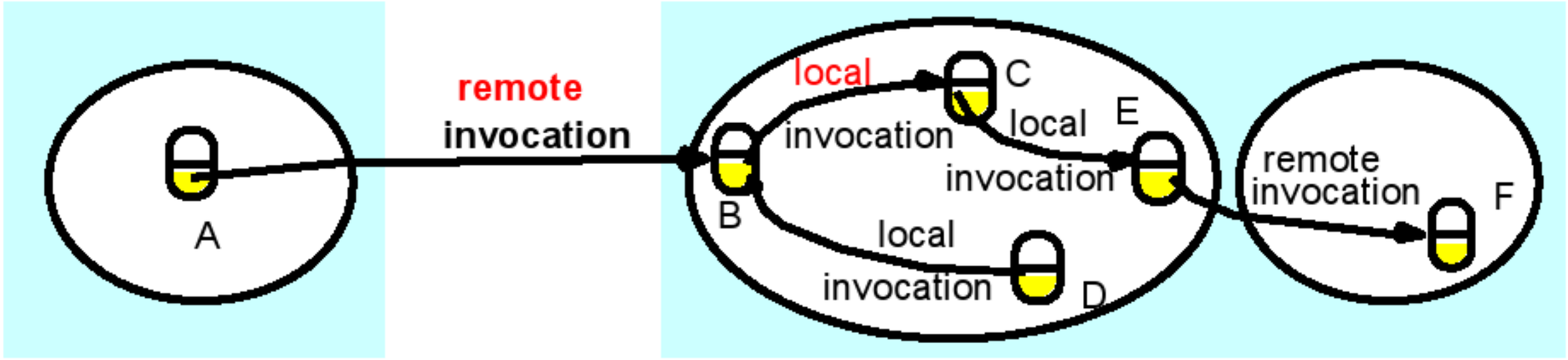


- Objects = Data (attributes) + Operations (methods)
  - encapsulating Data and Methods
  - State of Objects: value of its attributes
- Interact via interfaces:
  - define types of arguments and exceptions of methods

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



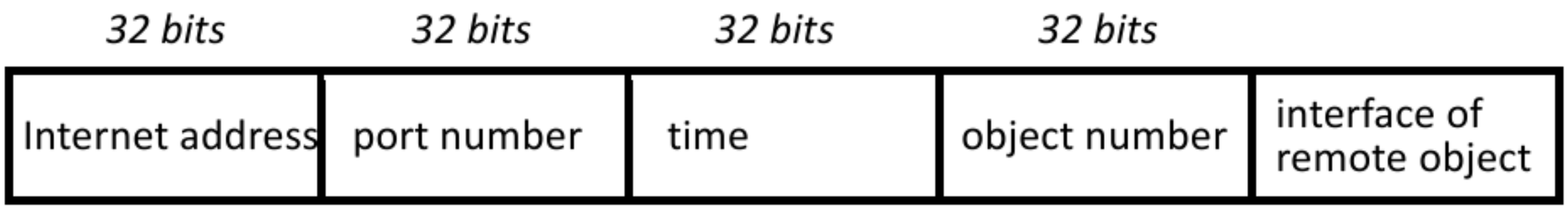
- Objects distributed (client-server models)
- Extend with
  - Remote object reference
  - Remote interfaces
  - Remote Method Invocation (RMI)

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

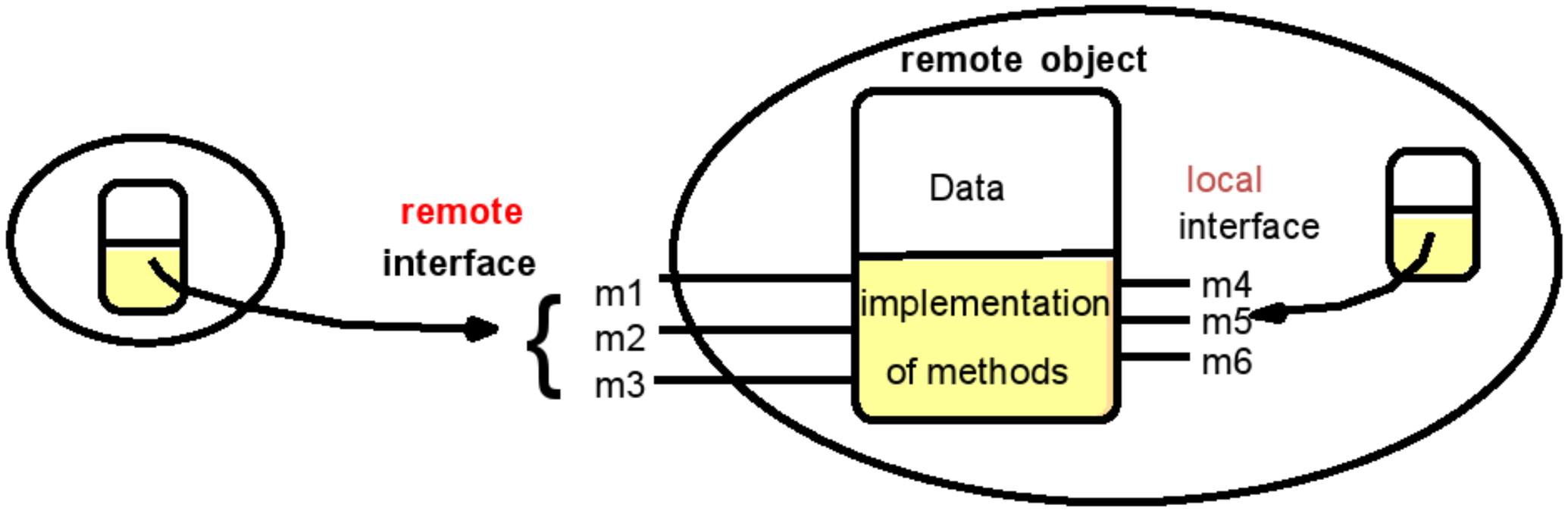
- Constructing unique remote object reference
  - IP address, port, interface name
  - time of creation, local object number (new for each object)
- Use the same as for local object references
- If used as addresses
  - cannot support relocation (alternative in CORBA)



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

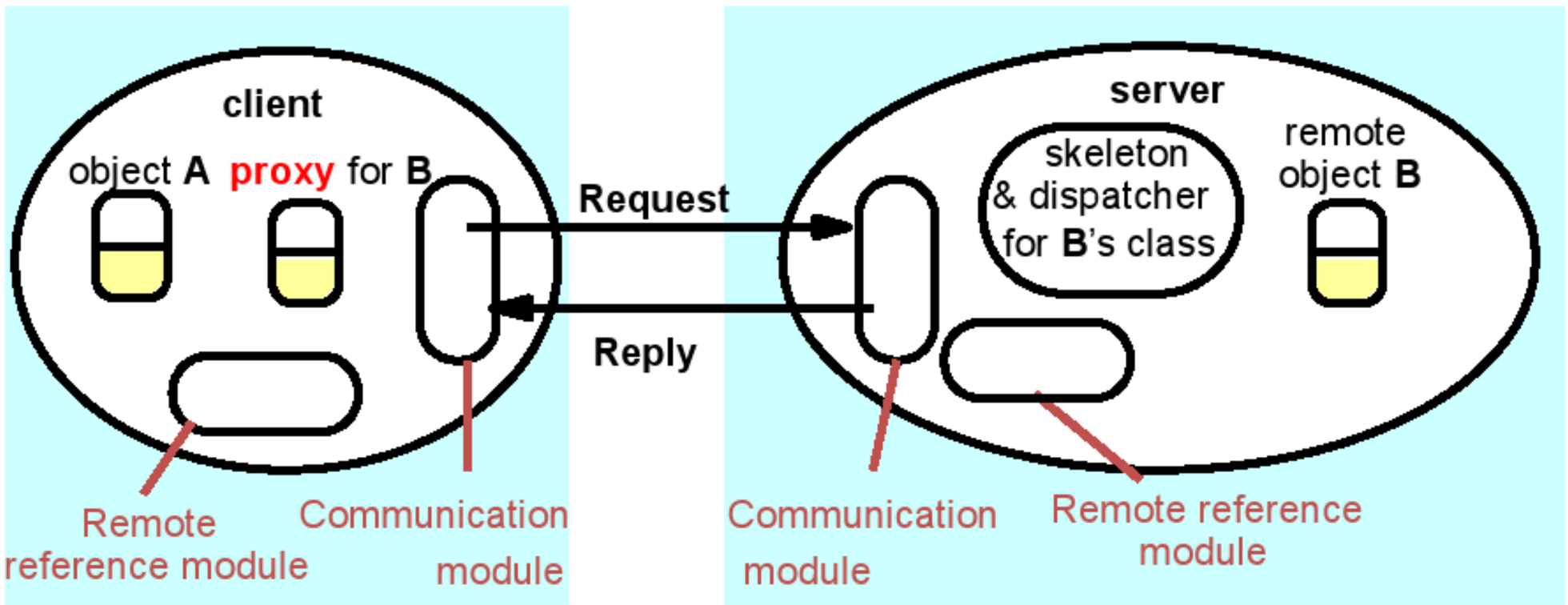
- Local invocations
  - executed exactly once
- Remote invocations
  - via Request-Reply (see *DoOperation*)
  - may suffer from communication failures!
    - ❑retransmission of request/reply
    - ❑message duplication, duplication filtering
  - no unique semantics…

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

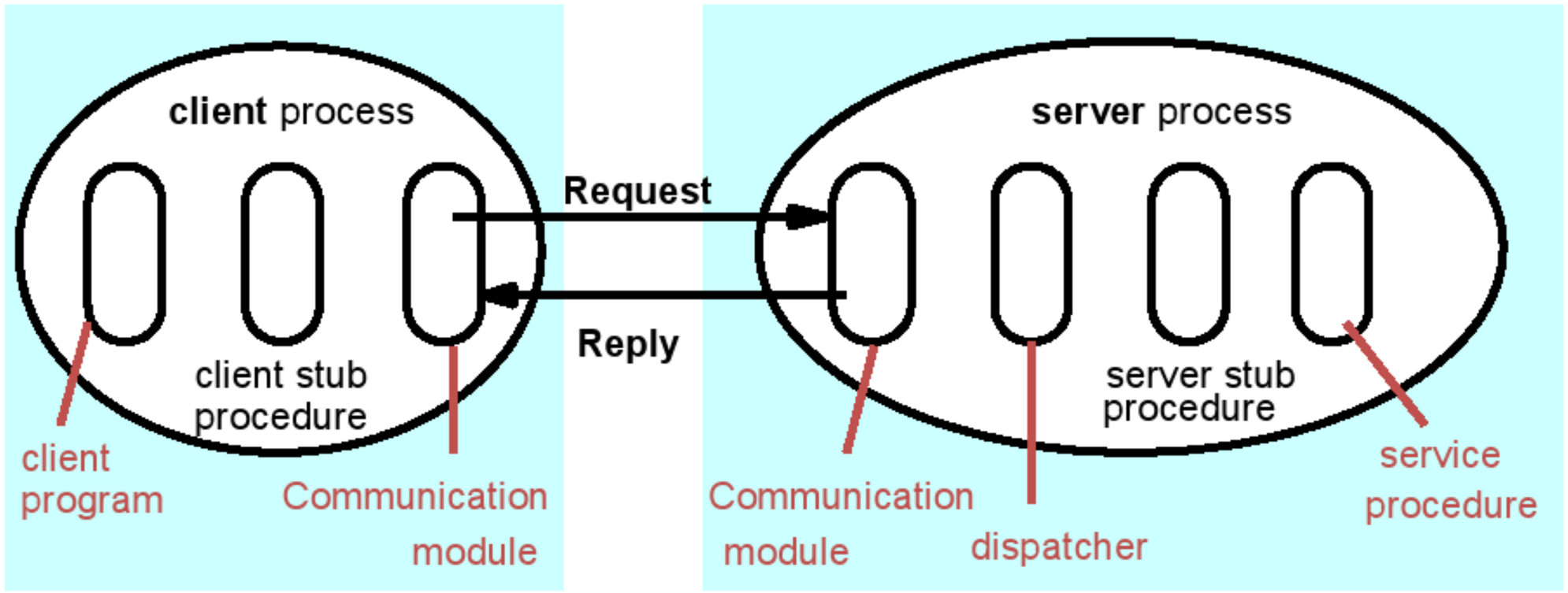
Object location issues

- 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

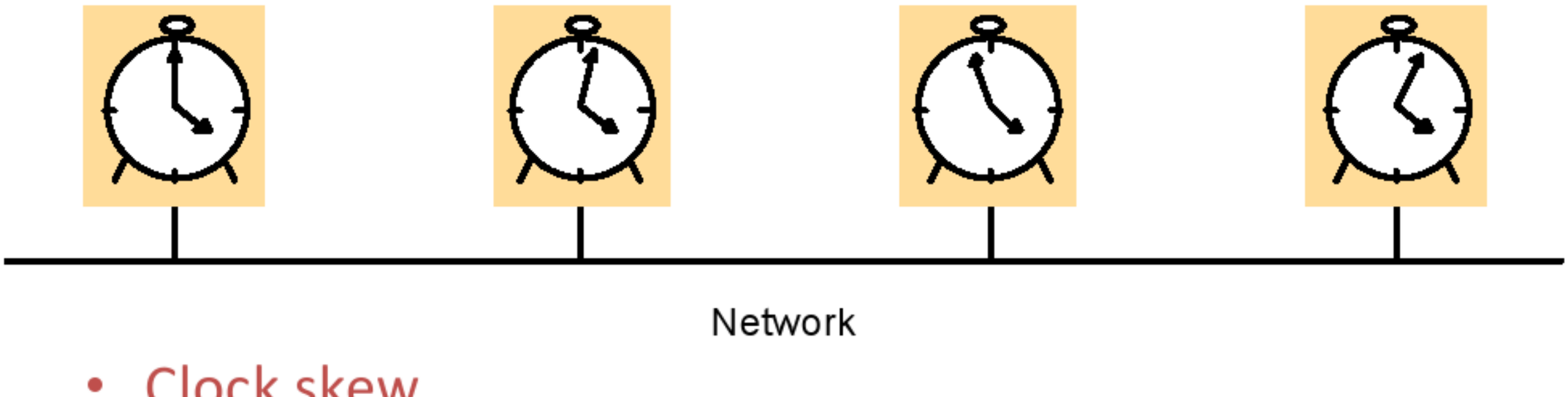
Time service

- 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

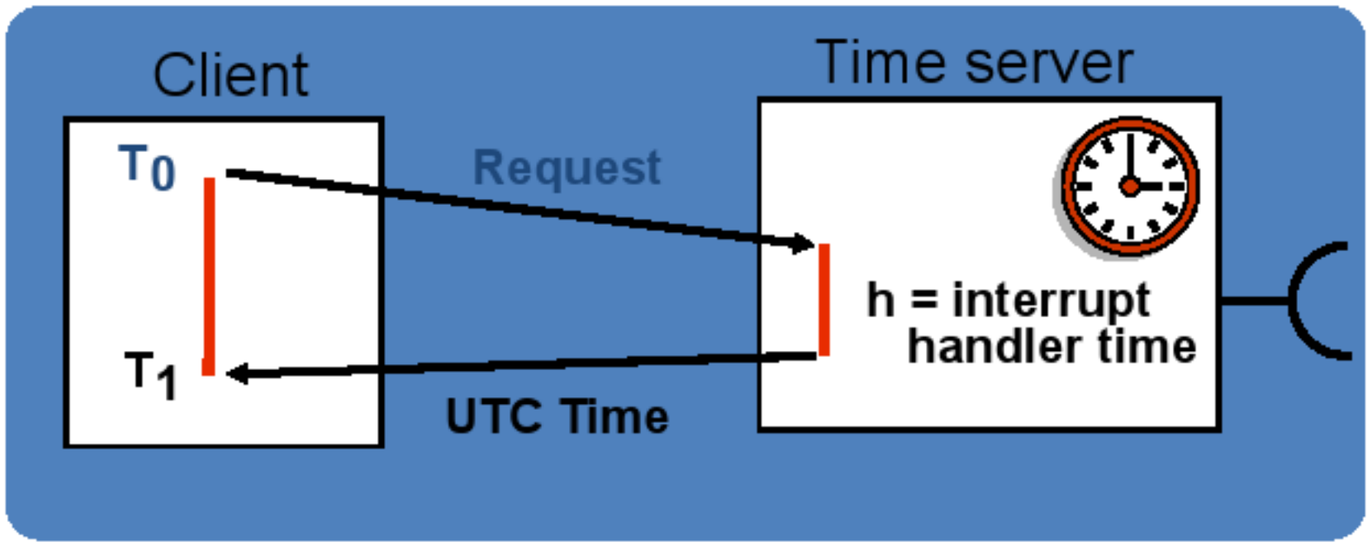
Synchronisation methods

- 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

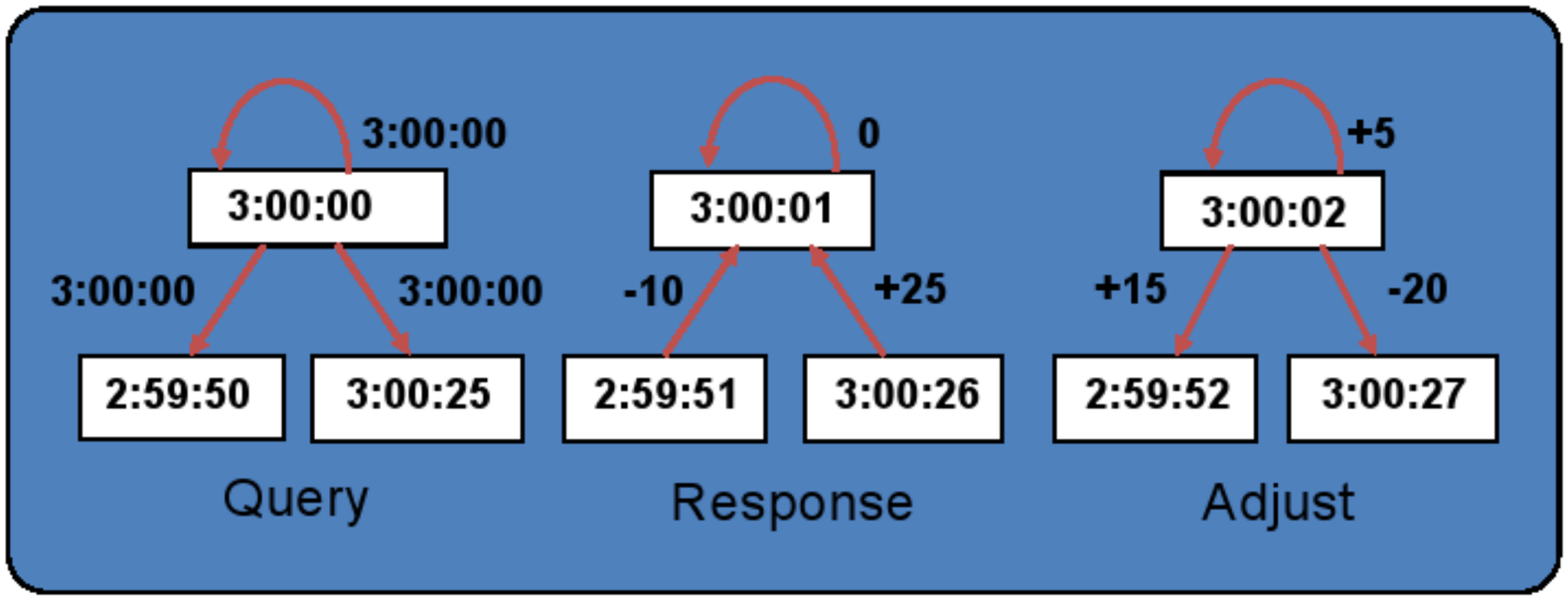
- but discard any that are over a threshold (could be congestion)
- or take minimum values as the most accurate

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.

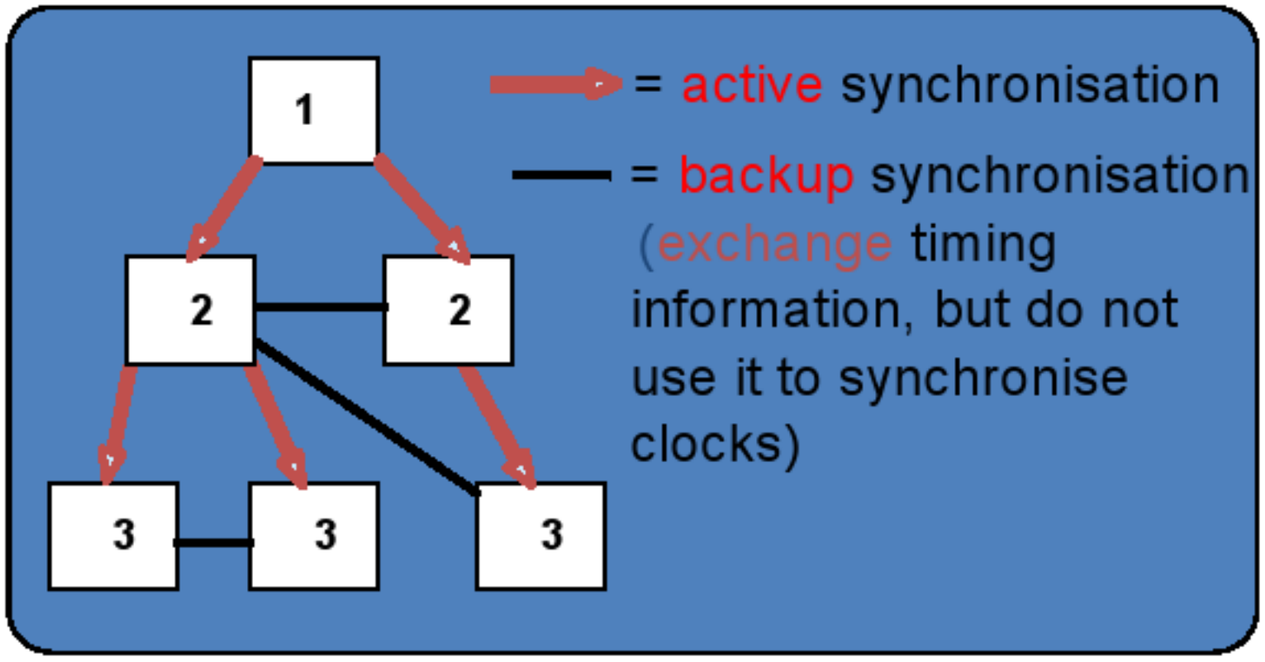
The Berkeley algorithm

- 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

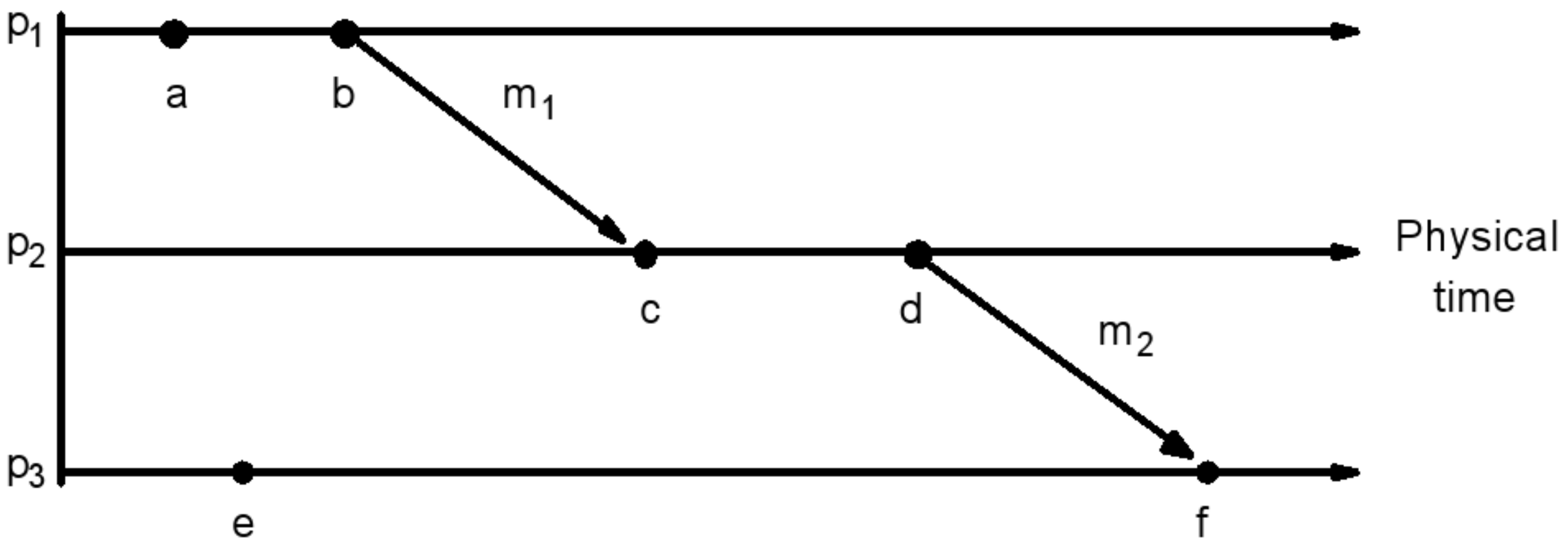
- For many purposes it is sufficient to agree on the same time (e.g. internal consistency) which need not be UTC time
- Can deduce causal event ordering

  $a \rightarrow b$ (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.

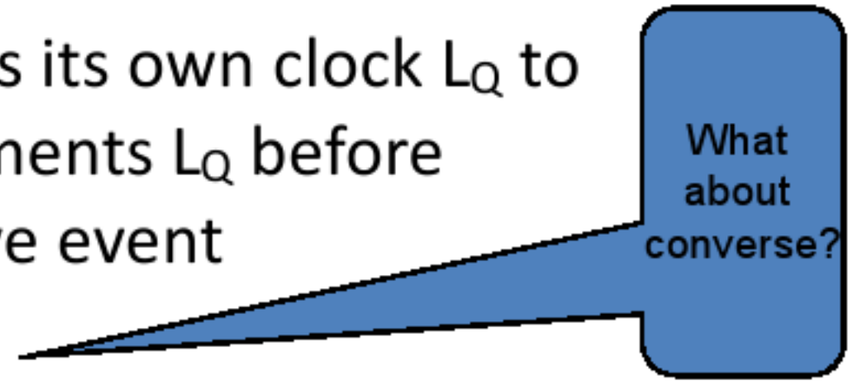
Example of causal ordering



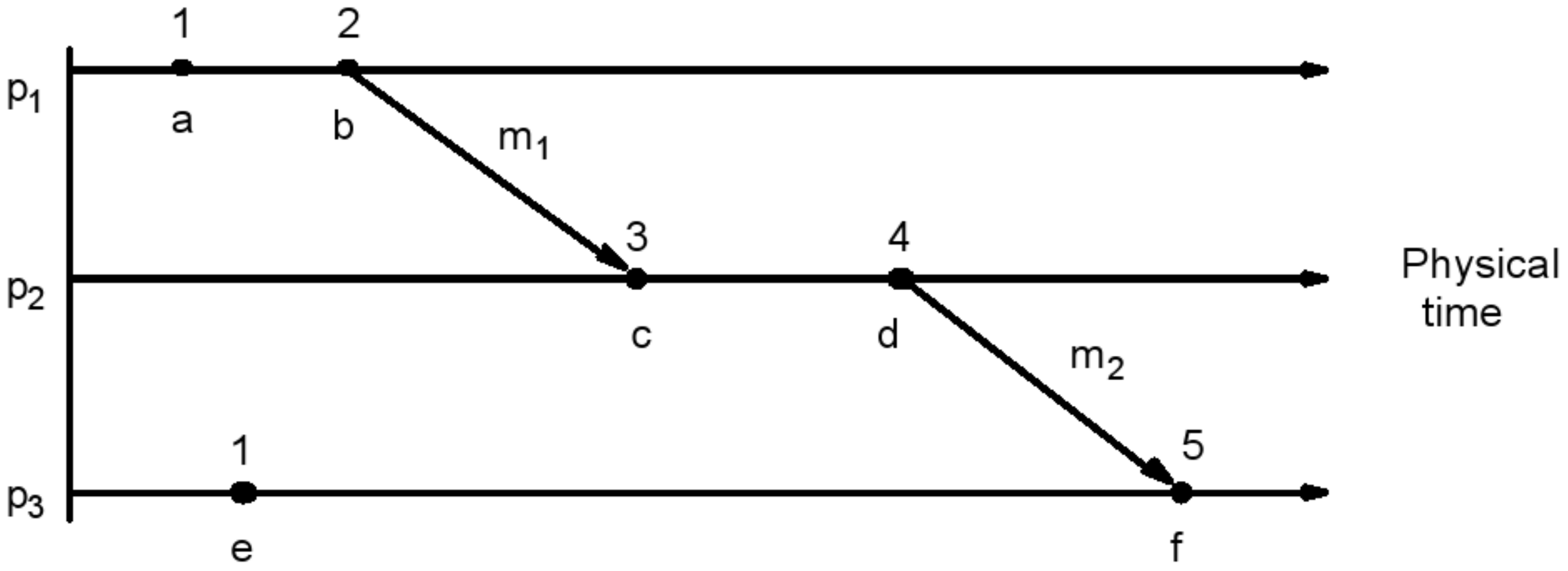
- a $\rightarrow$ b, c $\rightarrow$ d
- b $\rightarrow$ c, d $\rightarrow$ f
- a || e

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

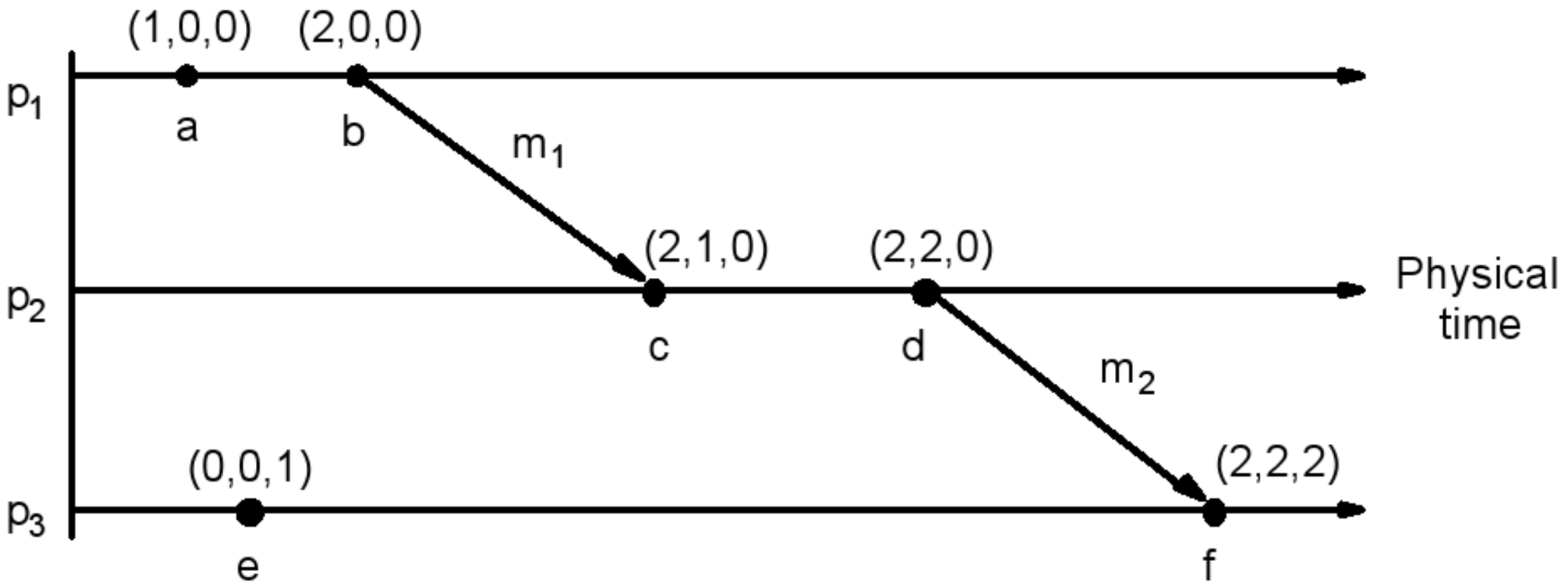


- Problem: T(a) = T(e), and yet a, e distinct.
- Create total order by taking account of process ids.
- Then (T(a),pid) < (T(b),qid) iff T(a) < T(b) or T(a)=T(b) and pid < qid.

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 → 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 → 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

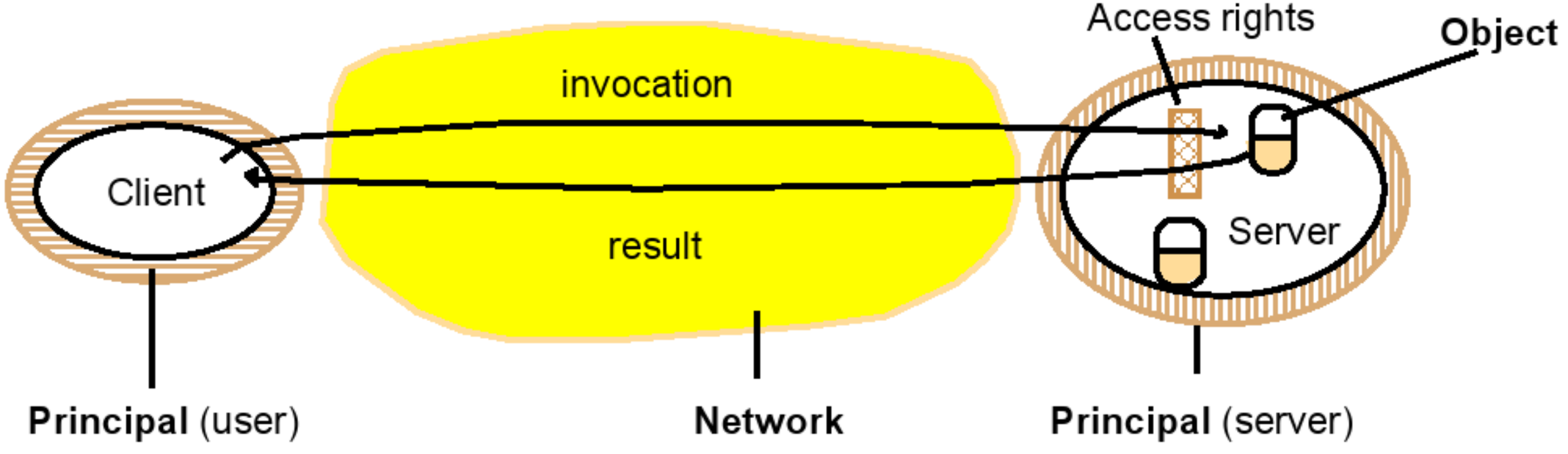
- What is security?
  - policies and mechanisms
  - threats and attacks
- Security of electronic transactions
  - secure channels
  - authentication and cryptography
- Security techniques
  - access control
  - firewalls
  - cryptographic algorithms

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)

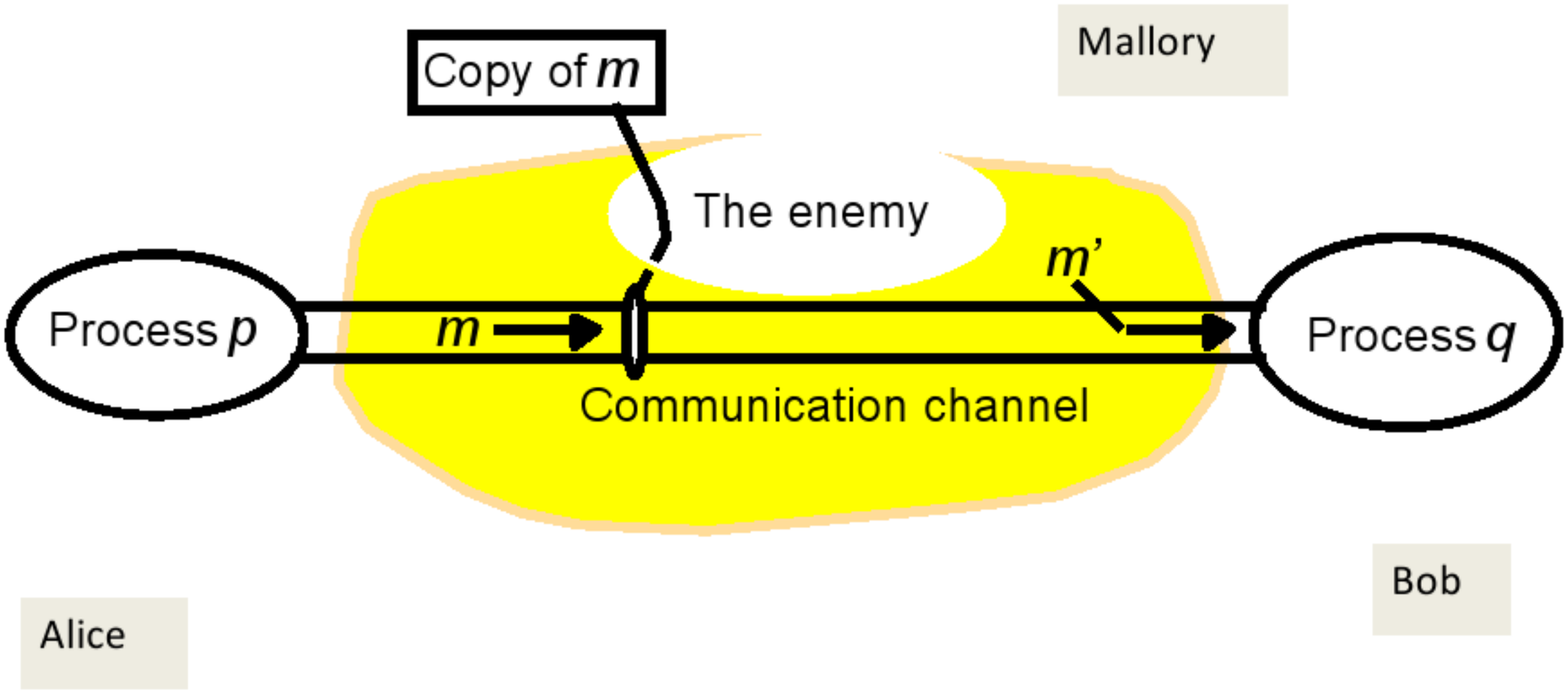
Security model

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



The enemy

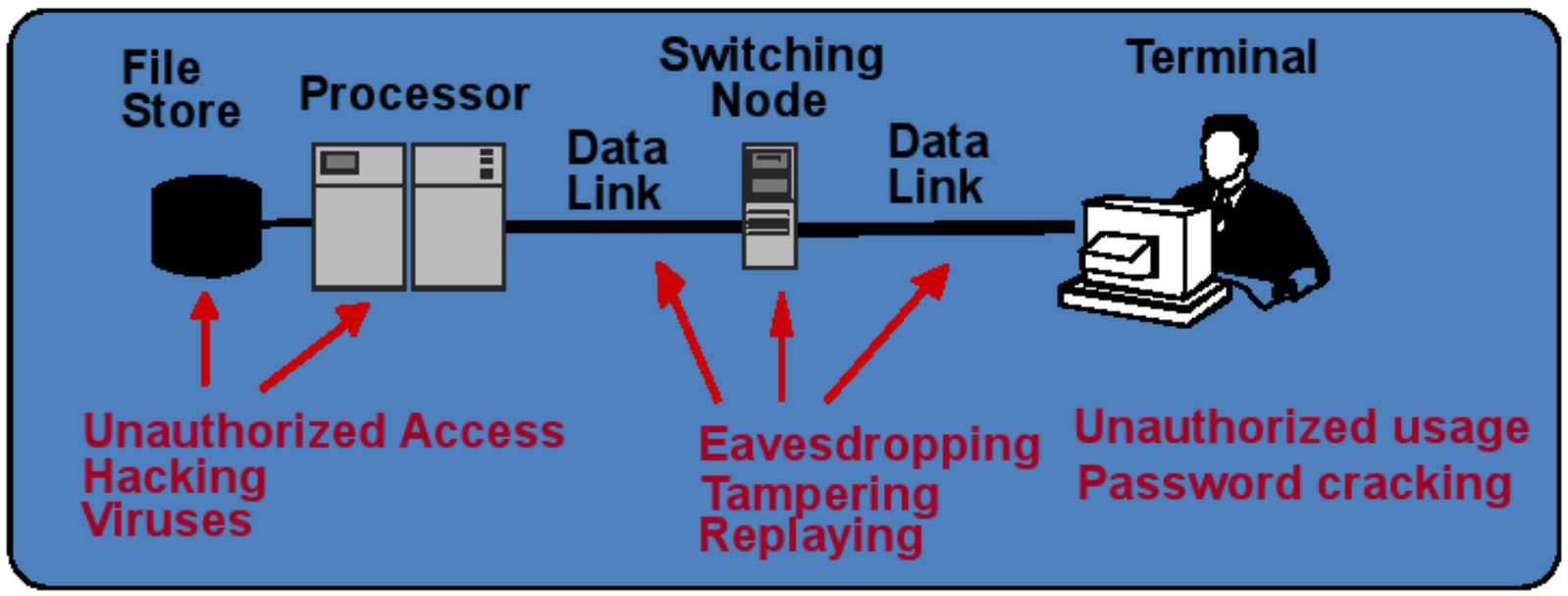
- Processes: encapsulate resources, interact by messages
- Messages: exposed to attack by enemy



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

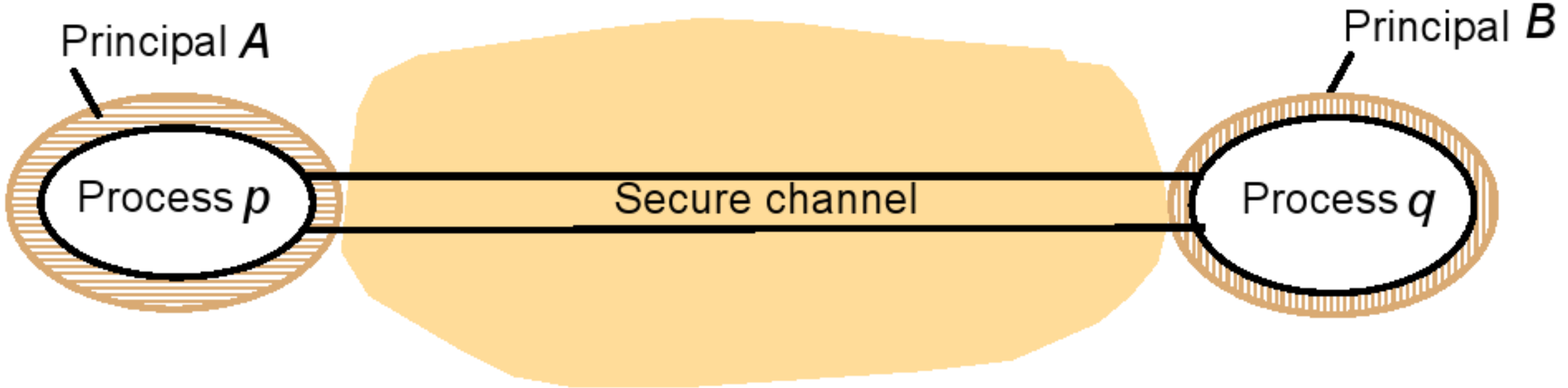
- Eavesdropping
  - obtaining copies of messages without authority
- Masquerading
  - sending/receiving messages using the identity of another principal without their authority
- Message tampering
  - intercepting and altering messages
- Replaying
  - intercepting, storing and replaying messages
- Denial 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

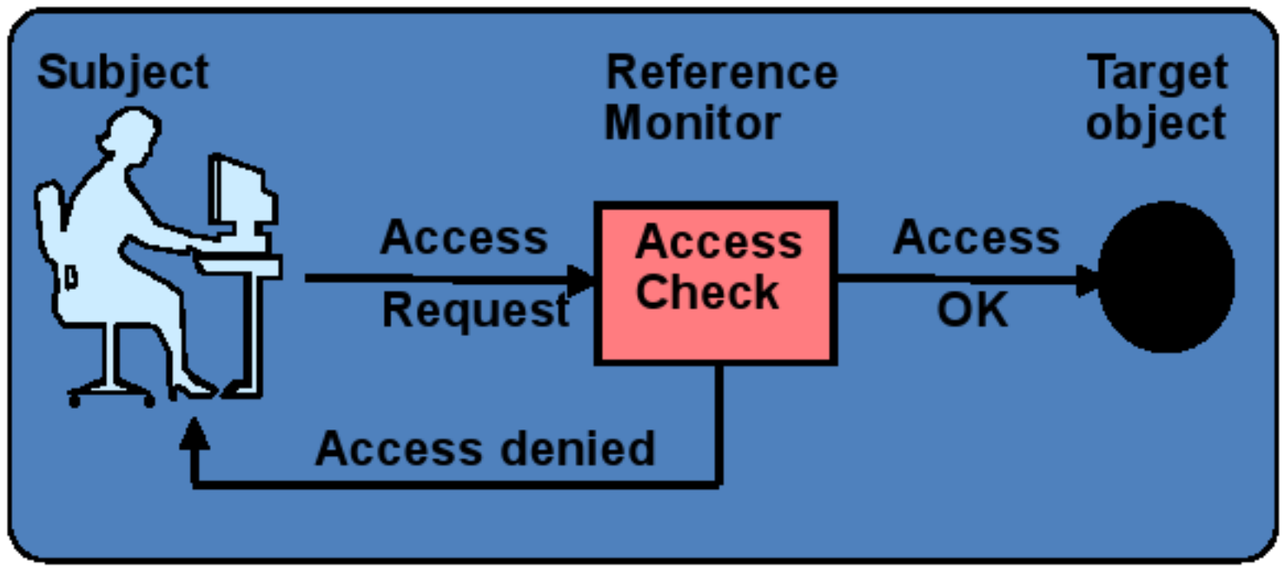
Main security techniques

- 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

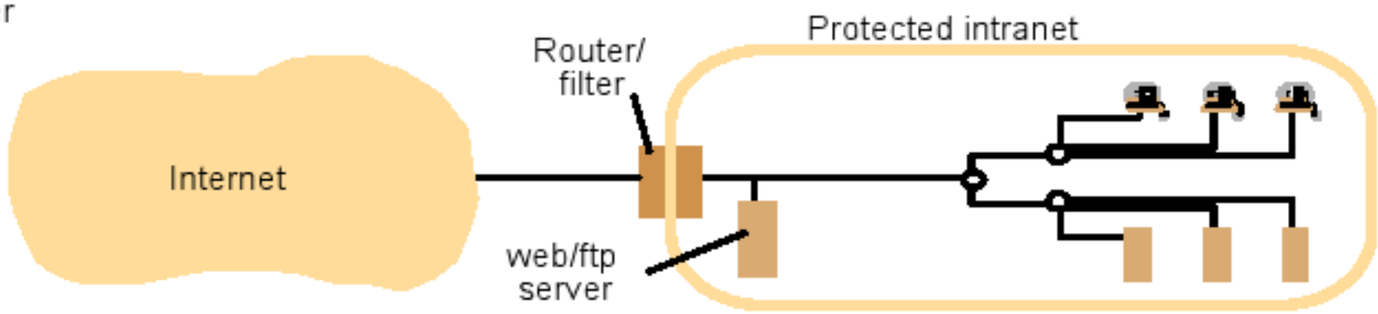
- Monitor and control all communication into and out of an intranet.
- Service control:
  - filter requests for services on internal hosts
  - e.g. reject HTTP request unless to official webserver
- Behaviour control
  - prevent illegal or anti-social behaviour
  - e.g. filter ‘spam’ messages
- User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

How does it work...

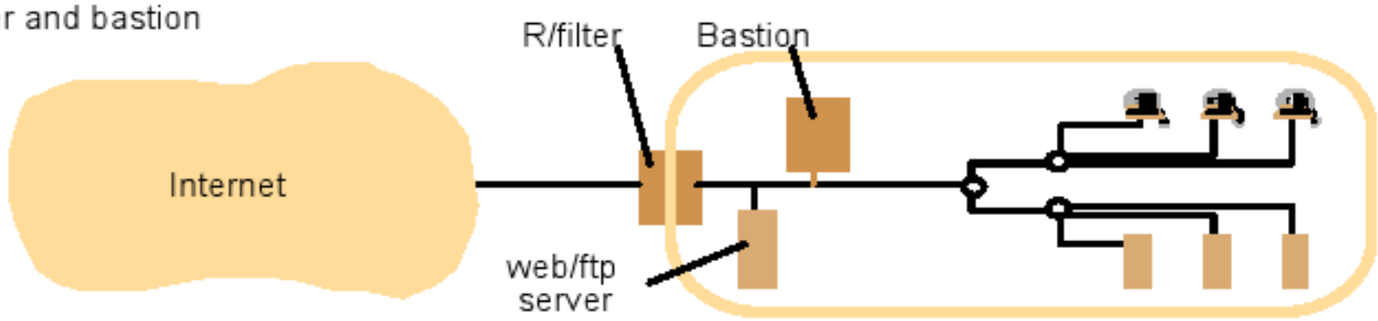
- A set of processes, at different protocol levels:
- IP packet filtering
  - screening of source & destination, only 'clean' packets proceed
  - performed in OS kernel of router
- TCP gateway
  - monitors TCP connection requests
- Application-level gateway
  - runs proxy for an application on TCP gateway,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

Firewall configurations

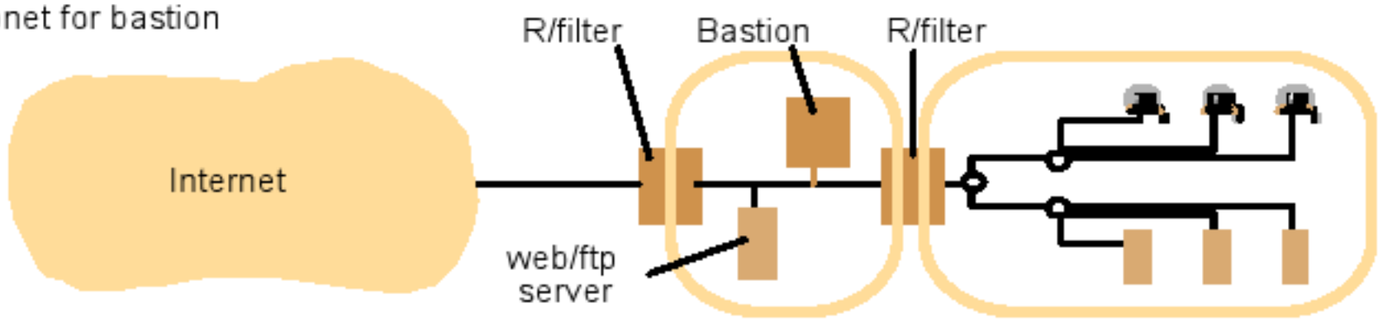
a) Filtering router



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)

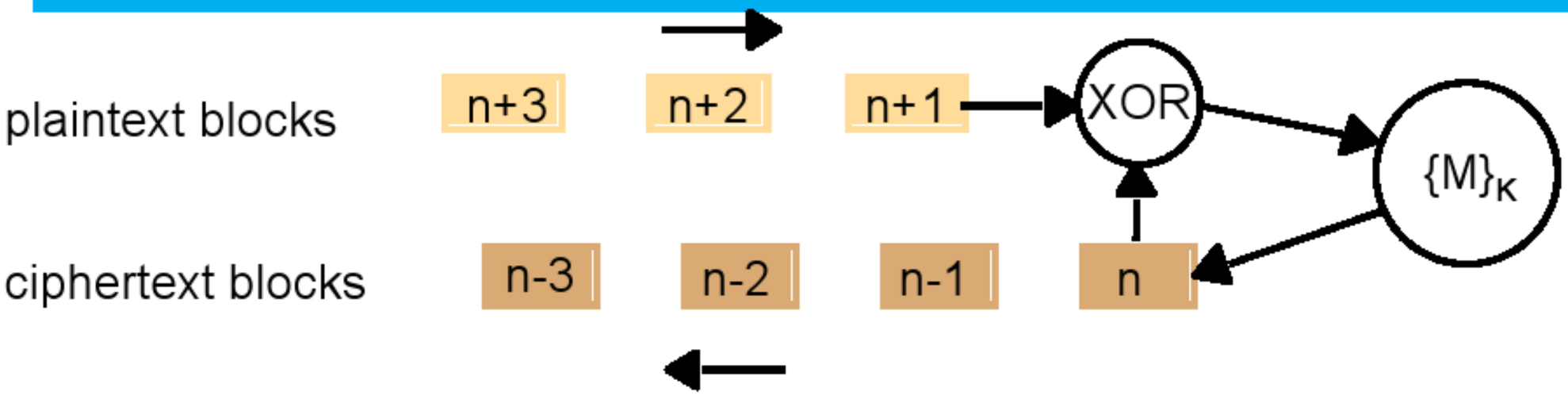
Symmetric cryptography

- 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

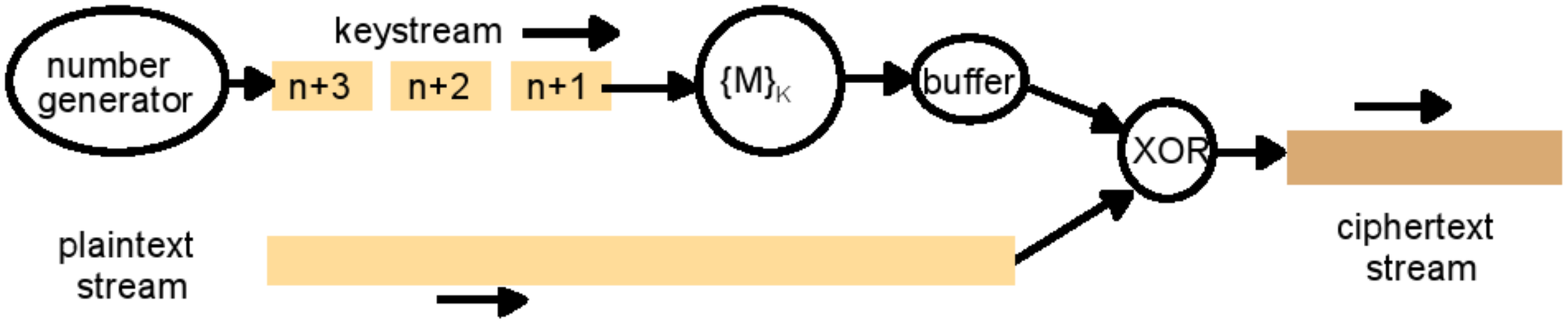
Block ciphers



- 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) × (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)

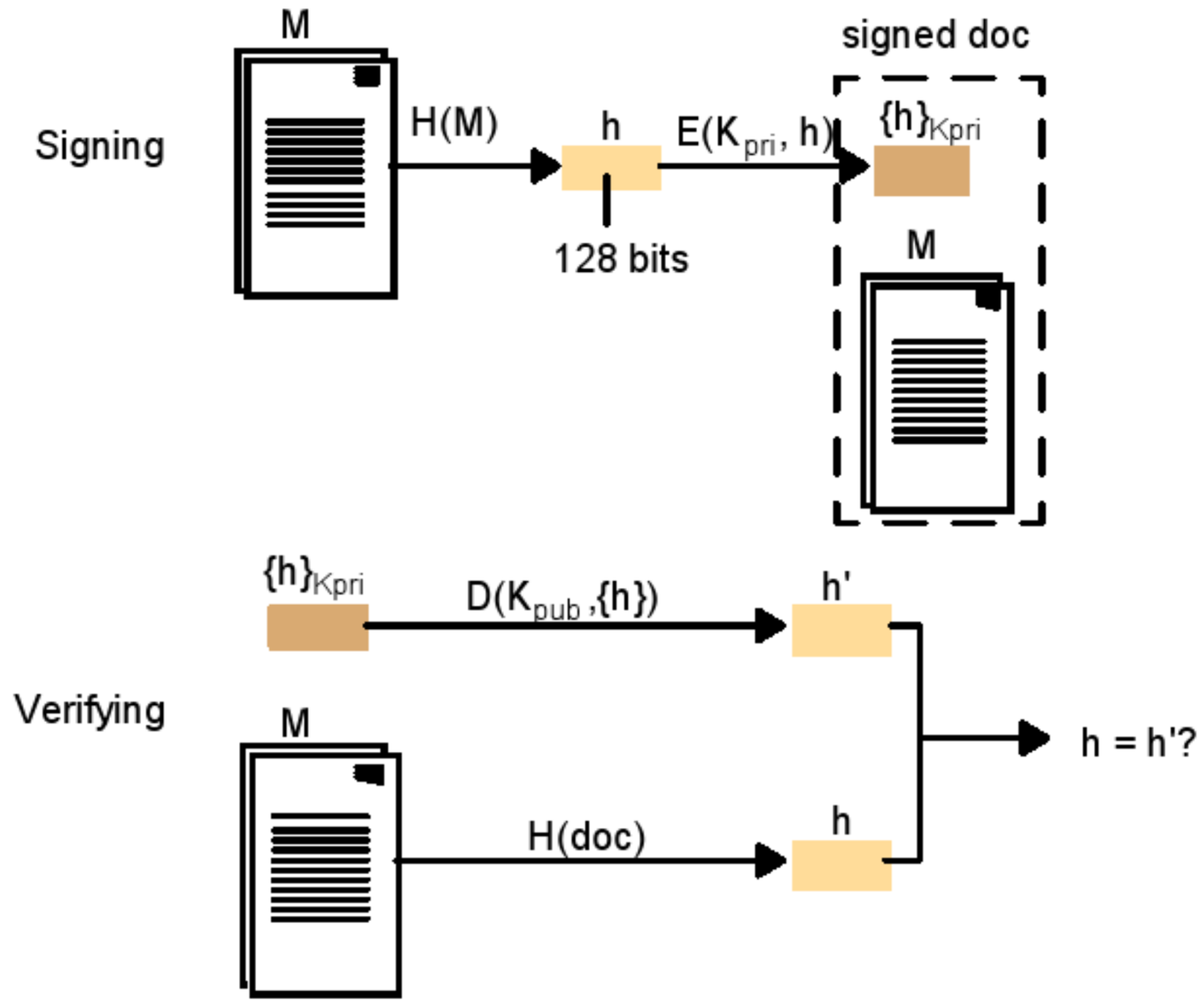
Digital signatures

- 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 $\{K_{AB}, A\}_{K_B}$ (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

    $\{K_{AB}, A, t\}_{K_{Bpub}}$
  - 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