Welcome!

MDS F2014 BSWU/SDT

Meta

You must provide feedback.

Form

- Conjoined KSDT/BSDT course
- ~ 10hr/week workload
- 10 weeks lectures + exercises
- 6 weeks groupwise mini-projects
- All communications via learnit+learnit forums.
  NB! Link:
  https://learnit.itu.dk/course/view.php?id=3003980
- Read your ITU email.

Exercises

- TAs: Holger Stadel Borum, Frederik Madsen
- Time and place: wed 10-12, fri 10-12 @ 3A52
- Exercises for lecture in exercise class
  the *following* week.
- … but really, up to you and TAs.

Exam

- D2G—Group hand-in, individual oral exam
- Hand-in during the course:
  - 3 mini-projects
  - 3 *designated* exercise sets
  - … and also in a collection at semester's end.

Prerequisites

- In order to participate at this course you must be able to use all major elements of the Java language for implementing and testing medium sized programs, including threads, inheritance, packages, I/O, streams and serialisation.

ILOs

- describe and use fundamental architectures, principles and models used in designing and constructing mobile and distributed systems.

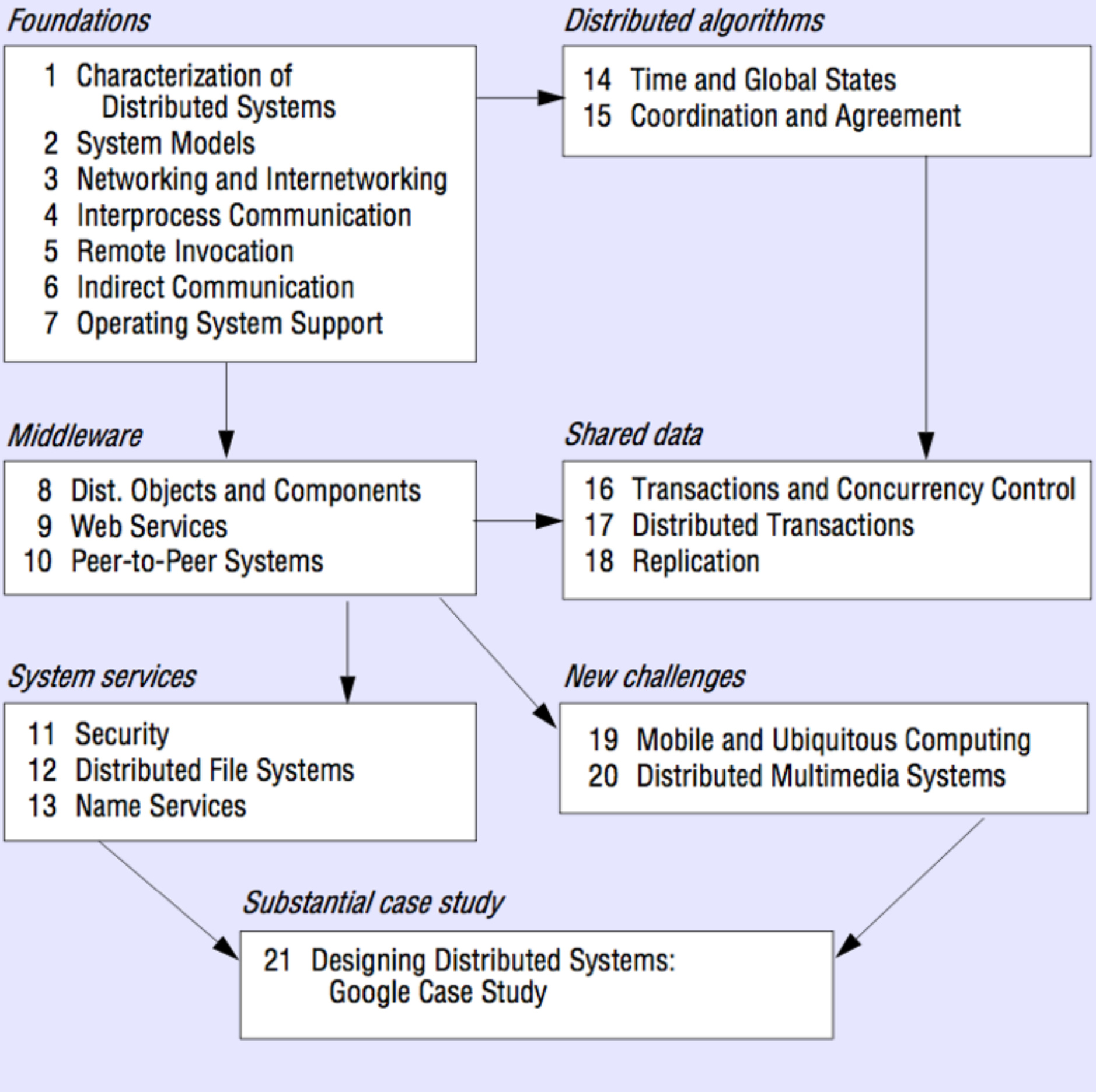
- describe and use basic concepts in mobile and distributed systems for evaluation and designing of solutions to problems in the field.

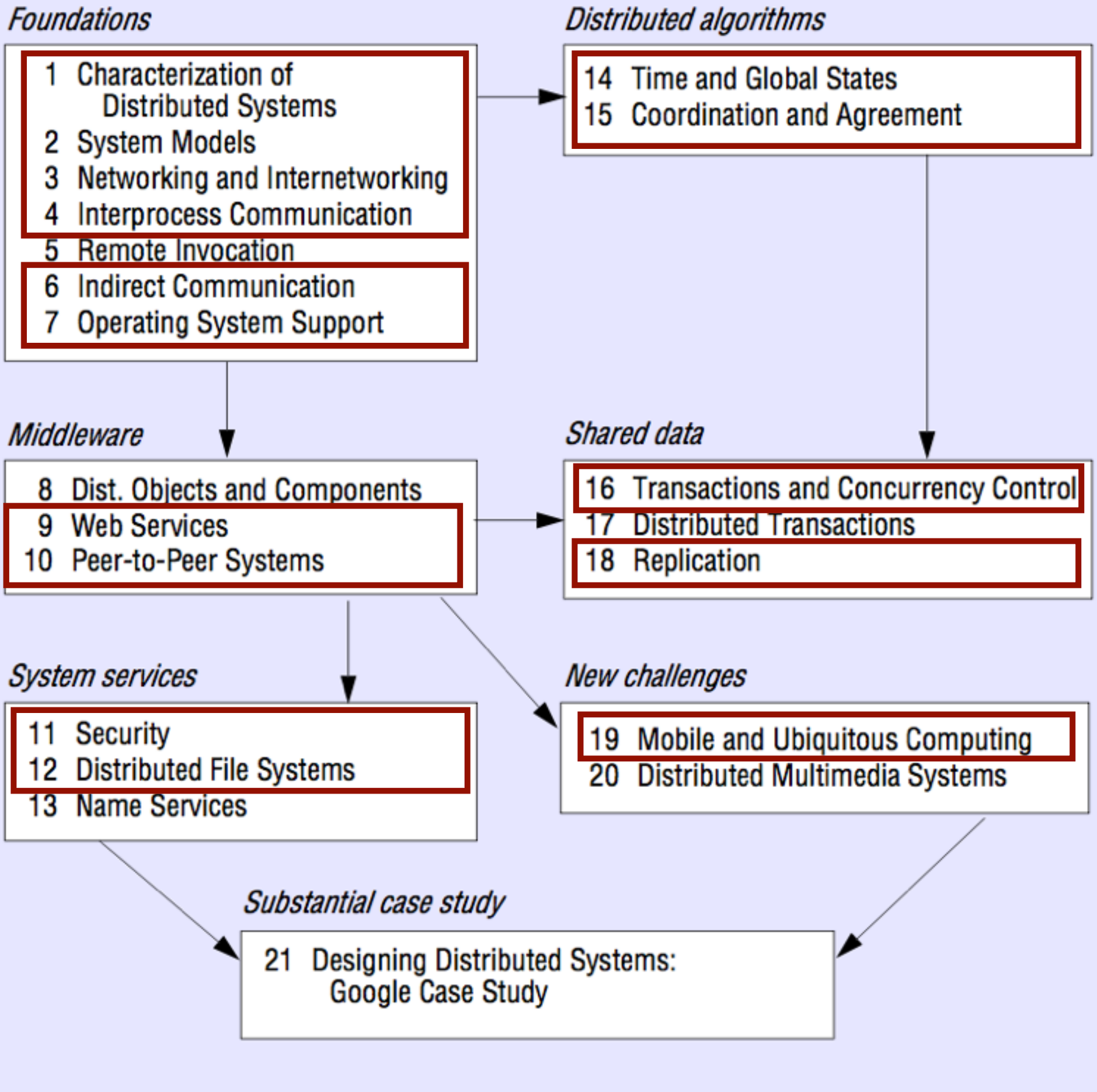
- implement distributed, and mobile systems in practice with the help of the above techniques

ILOs

- describe and use fundamental architectures, principles and models used in designing and constructing mobile and distributed systems.
- describe and use basic concepts in mobile and distributed systems for evaluation and designing of solutions to problems in the field.
- implement distributed, and mobile systems in practice with the help of the above techniques

Principles, practice, and code.


Foundations
1 Characterization of Distributed Systems
2 System Models
3 Networking and Internetworking
4 Interprocess Communication
5 Remote Invocation
6 Indirect Communication
7 Operating System Support
Distributed algorithms
14 Time and Global States
15 Coordination and Agreement
Middleware
8 Dist. Objects and Components
9 Web Services
10 Peer-to-Peer Systems
Shared data
16 Transactions and Concurrency Control
17 Distributed Transactions
18 Replication
System services
11 Security
12 Distributed File Systems
13 Name Services
New challenges
19 Mobile and Ubiquitous Computing
20 Distributed Multimedia Systems
Substantial case study
21 Designing Distributed Systems: Google Case Study


Foundations
1 Characterization of Distributed Systems
2 System Models
3 Networking and Internetworking
4 Interprocess Communication
5 Remote Invocation
6 Indirect Communication
7 Operating System Support
Distributed algorithms
14 Time and Global States
15 Coordination and Agreement
Middleware
8 Dist. Objects and Components
9 Web Services
10 Peer-to-Peer Systems
Shared data
16 Transactions and Concurrency Control
17 Distributed Transactions
18 Replication
System services
11 Security
12 Distributed File Systems
13 Name Services
New challenges
19 Mobile and Ubiquitous Computing
20 Distributed Multimedia Systems
Substantial case study
21 Designing Distributed Systems: Google Case Study

Tips

- Do the exercises.
- Read up-front. Lectures don’t cover all of book.
    - Requests for specific coverage no later than friday 9am welcome.
- Form comparable-experience groups.

One must learn by doing the thing;
for though you think you know it,
you have no certainty, until you try.

Sophocles, 496-405 BC.

I cant teach you anything.

Questions?

Distributed Systems

Introduction

Coulouris chapter 1

“A distributed system is one in which components located at networked computers communicate and coordinates their actions only by passing messages.”

–p.17

"A distributed system is one in which components located at networked computers communicate and coordinates their actions only by passing messages."

- Concurrency is the norm
- No global clock
- Independent failures

Challenges

- Heterogeneity
- Openness
- Security
- Scalability
- Availability and Failure handling
- Concurrency
- Transparency
- Quality of Service

Failure handling

- Detect failures
- Mask failures
- Tolerate/avoid failures
- Recover from failures

Security

- Access control

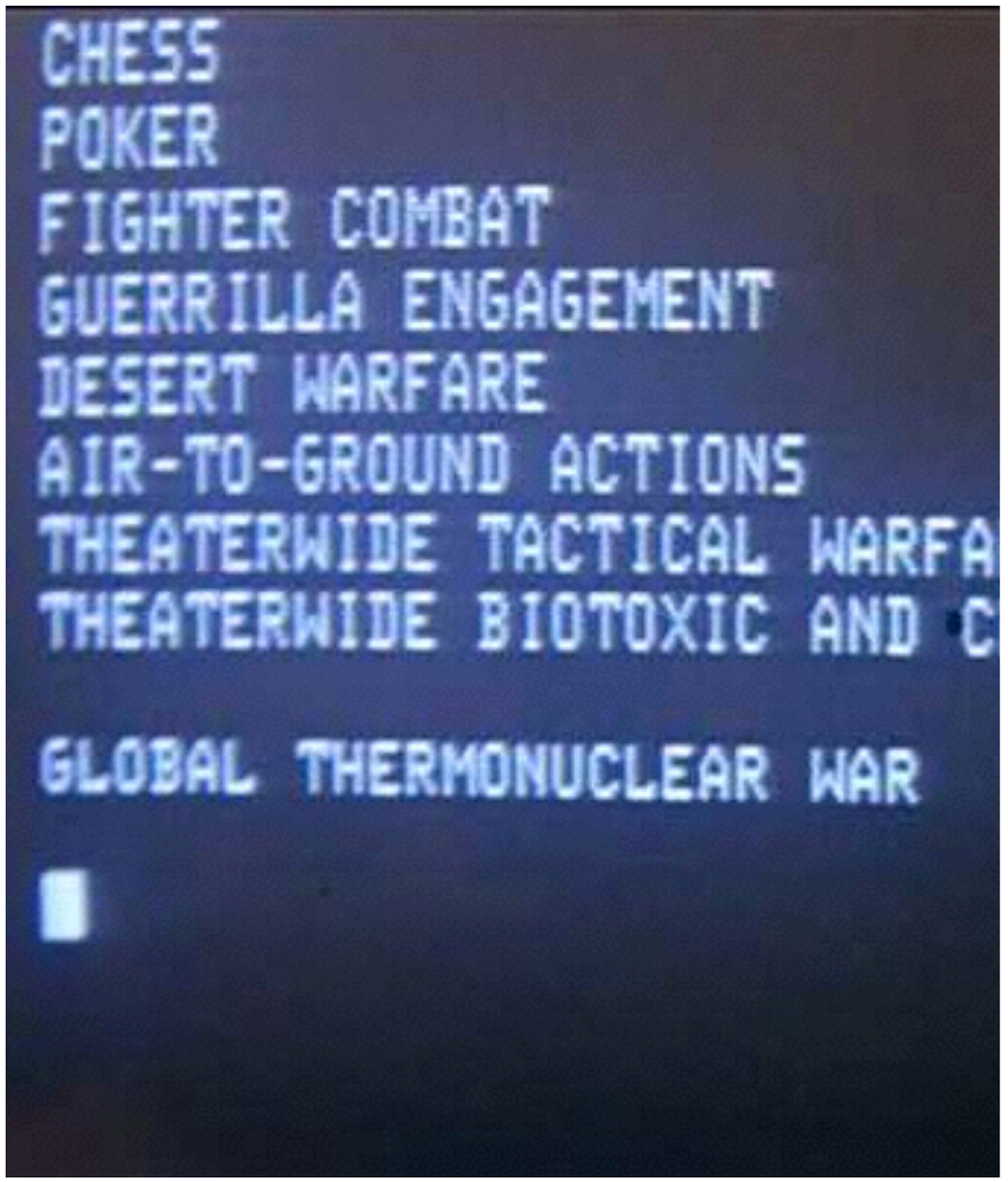
  (authentication, identification)

- Security

- “Denial of Service attacks”

- updates, delegation, mobile code

- Social engineering...



Concurrency

- resource sharing
- interference
- deadlock
- fairness
- concurrency control

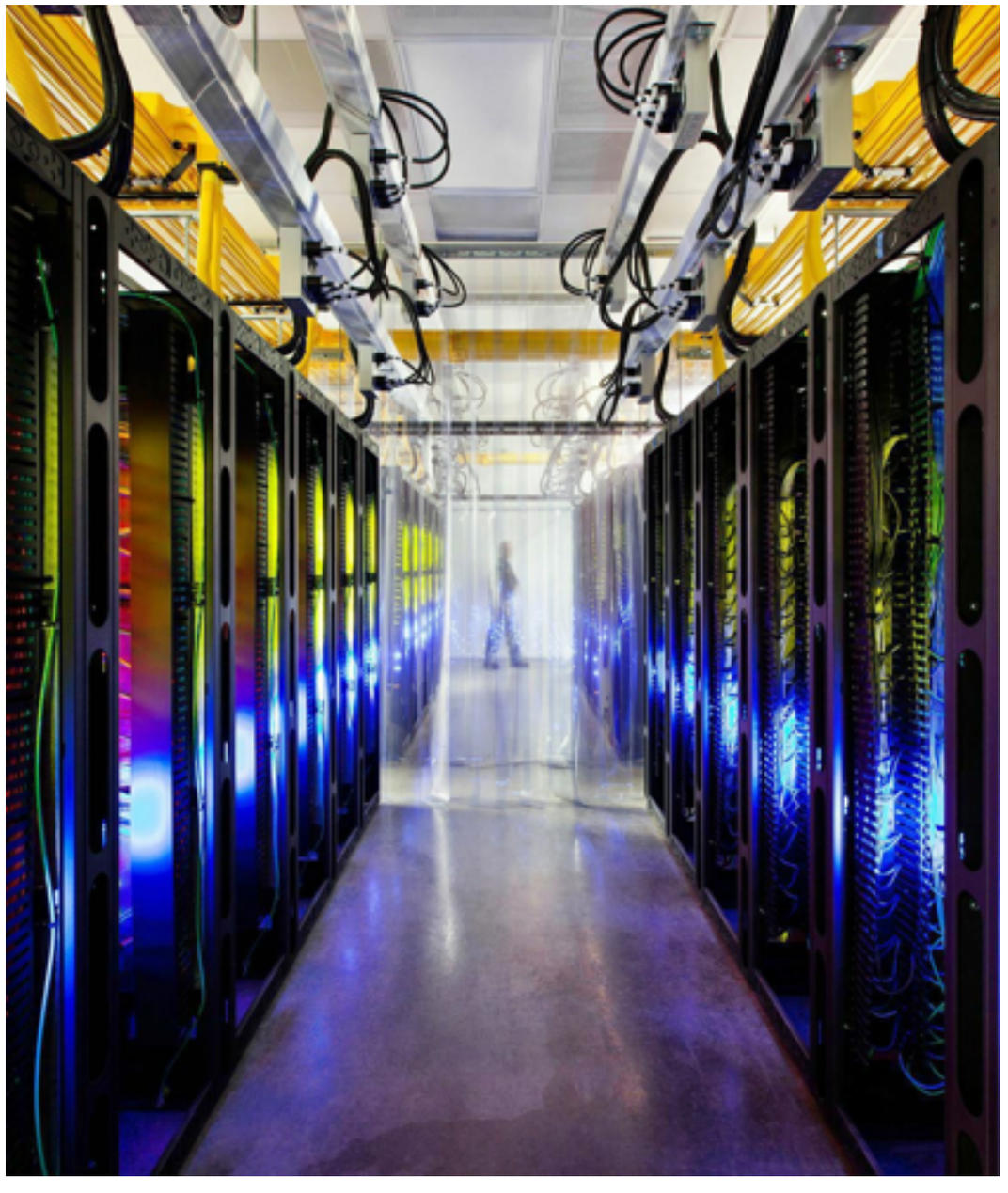
Scalability

- Physical resources: O(n)
- Performance loss:

  hierarchical data structures: O(log n)

  avoid bottlenecks

- Software resources: E.g. 32 bit IP addresses

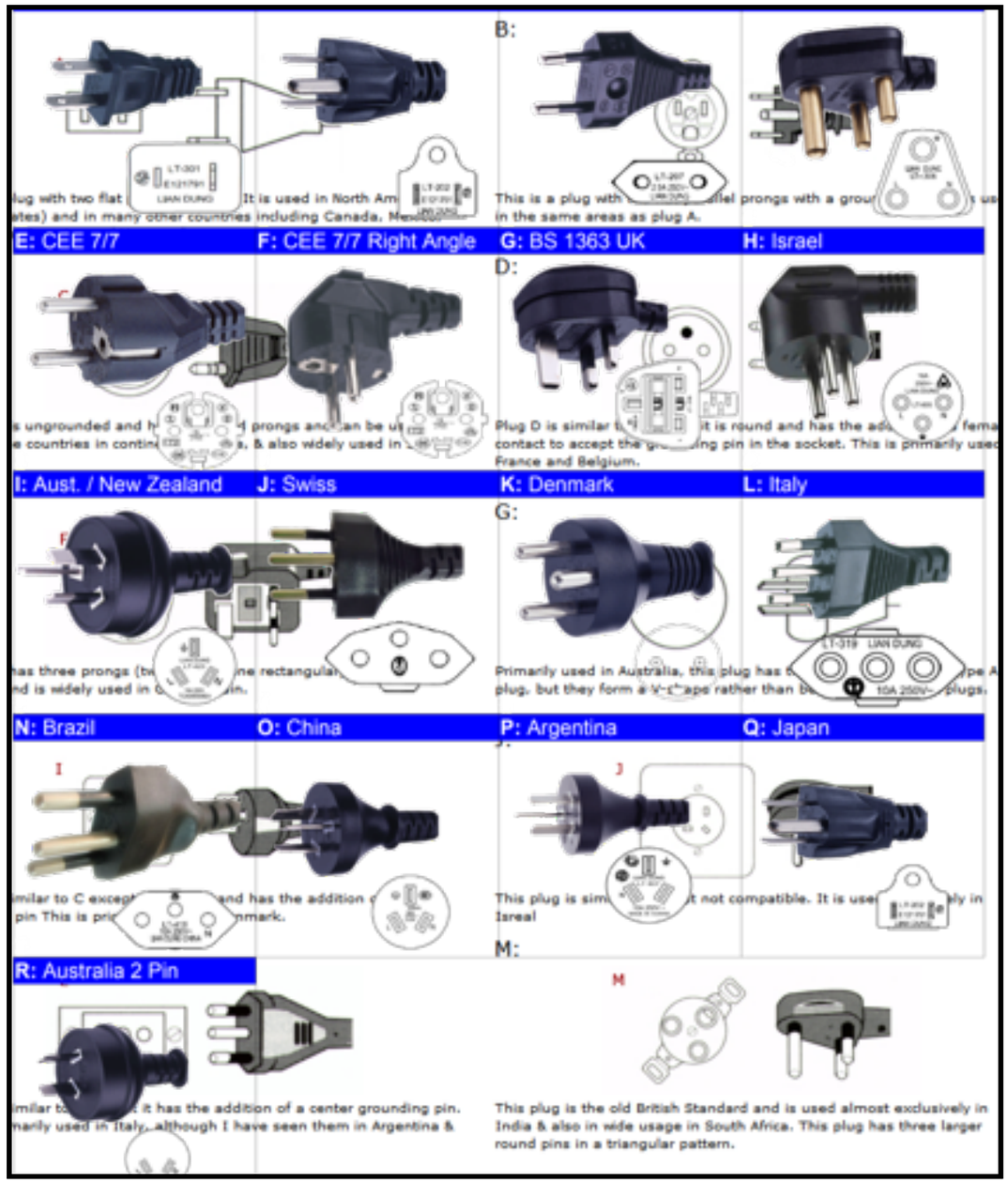


Transparency

- local/remote access
- physical location
- concurrency
- replication
- failure
- mobility
- performance
- scaling

Heterogeneity

- network
- computer hardware
- operating systems (windows, linux, unix, ..)
- programming languages
- software developers



Openness

- Support of extensions and changes
- Requires standardized public interfaces
- www is a good example

Quality of Service

- Reliability
- performance
- security

Summary

- “[A distributed system is] one in which components located at networked computers communicate and coordinate their actions only by passing messages.”

- Challenges: Heterogeneity, Openness, **Security**, Scalability, Availability, **Failure handling**, **Concurrency**, Transparency, Quality of Service.

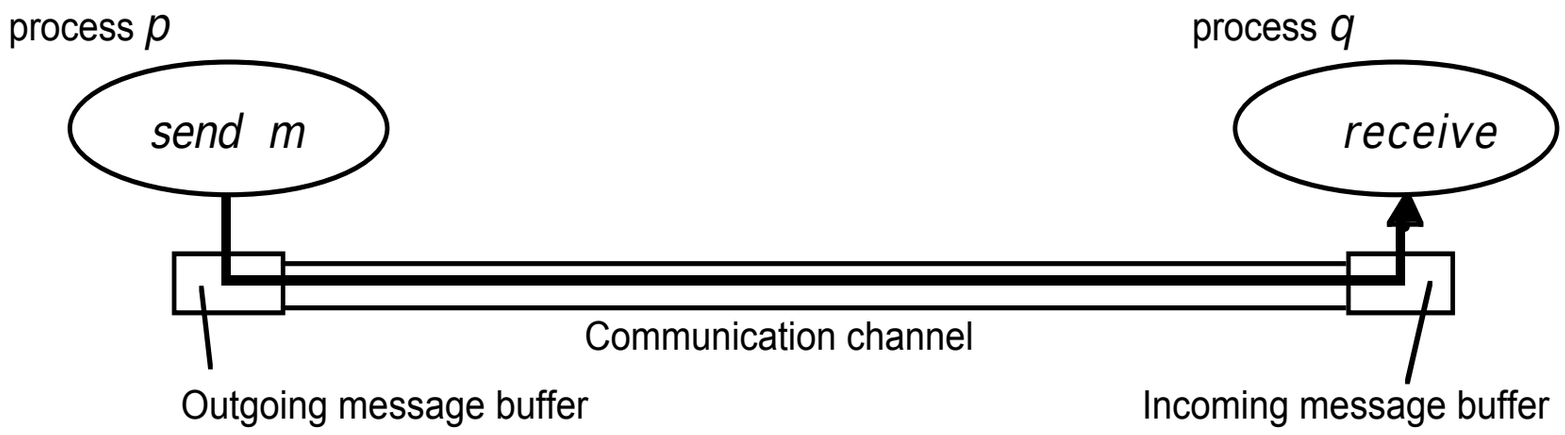
Break.

Foundations of Networking

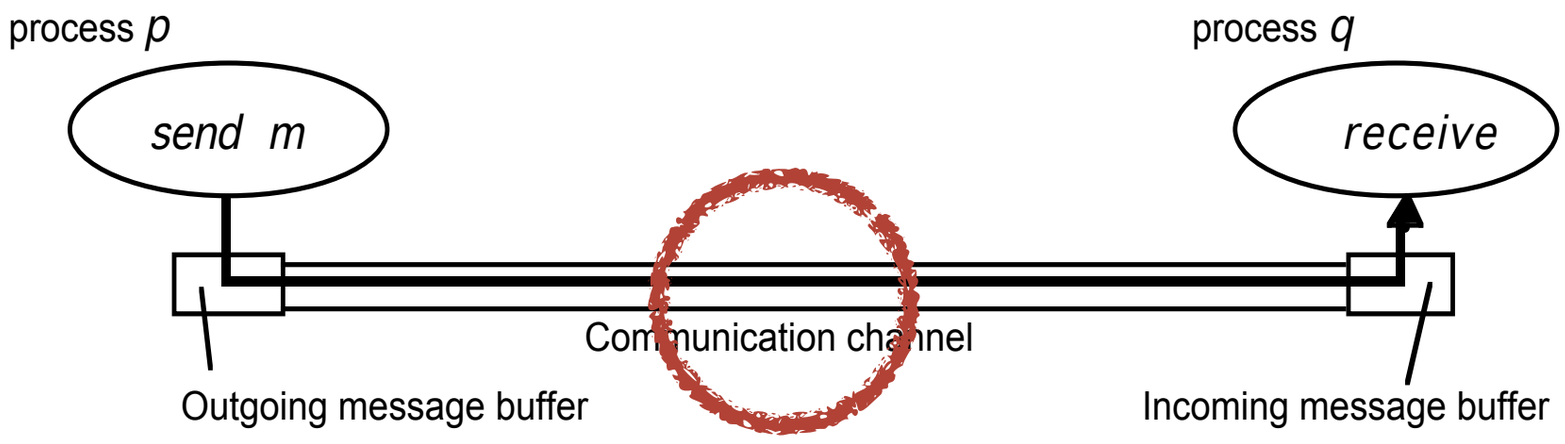
Terminology (i)

- *Processes* run on *hosts*.
- Processes send and receive *messages* on *communication channels.*
- Processes and communication channels are both subject to *failures.*
- Channel failures may yield *dropped, lost, re-ordered* or *duplicated* messages.

Basic model


process p
process q
send m
receive
Communication channel
Outgoing message buffer
Incoming message buffer

Basic model: Failures



Channel may exhibit omission & byzantine failures

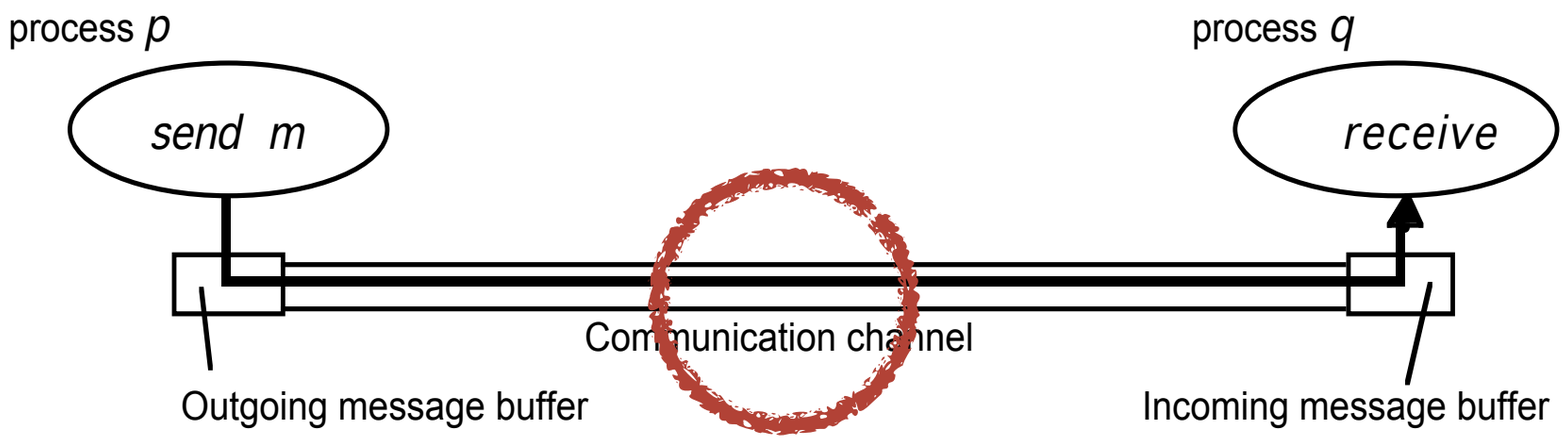
Channel failure handling

- Checksums
  (vs. message corruption)

- Acknowledgments, timeouts, re-transmissions
  (vs. lost messages)

- Sequence numbers
  (vs. duplicated/re-ordered messages)

Basic model: Security



Adversary may
intercept, copy, transform, insert, and delete
messages

Terminology (ii)

- Reliable vs unreliable channels/protocols
- Connection-oriented vs connection-less/protocols
- Datagram vs streaming

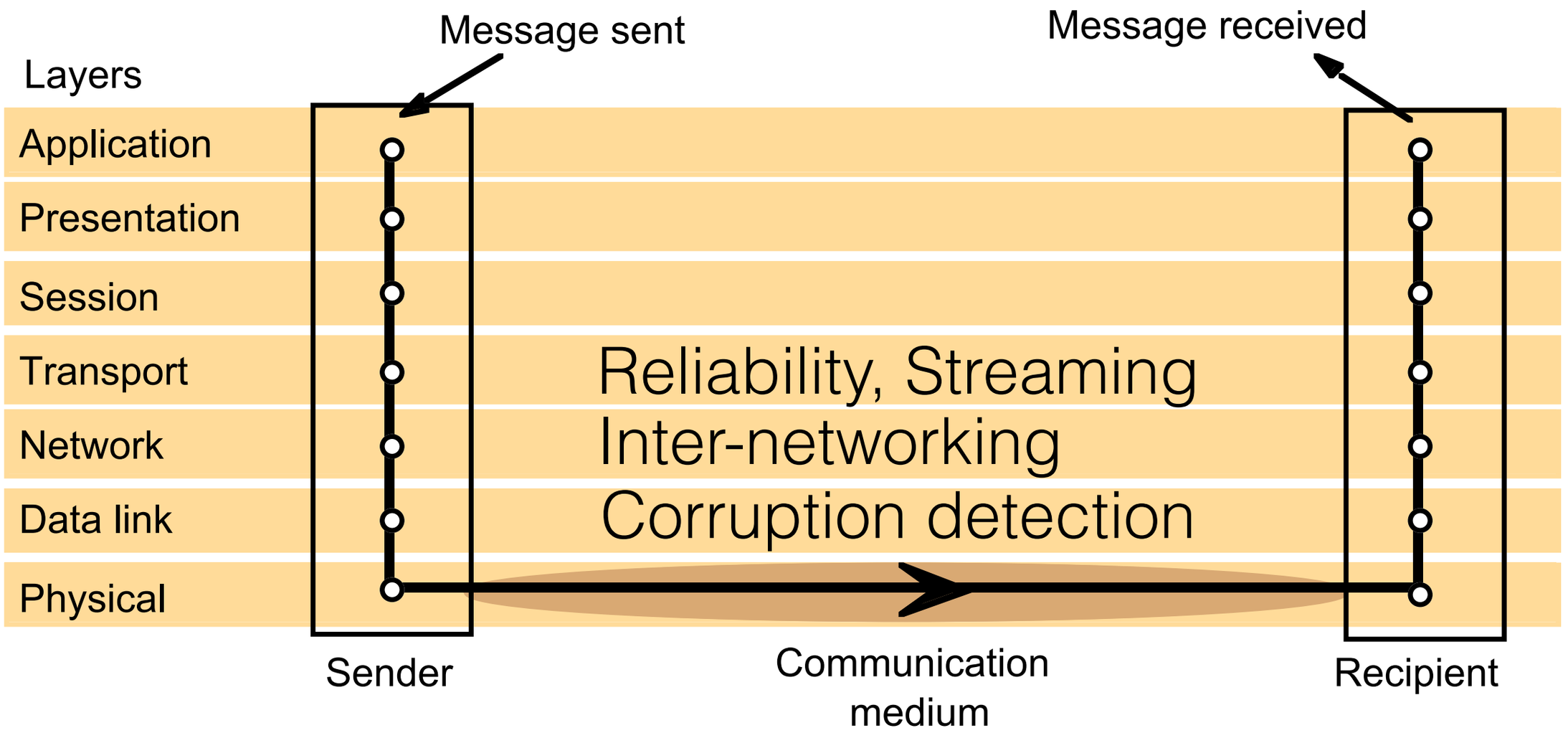
Terminology (iii)

- Processes adhere to *protocols*: specified sequences of message exchanges and data formats.

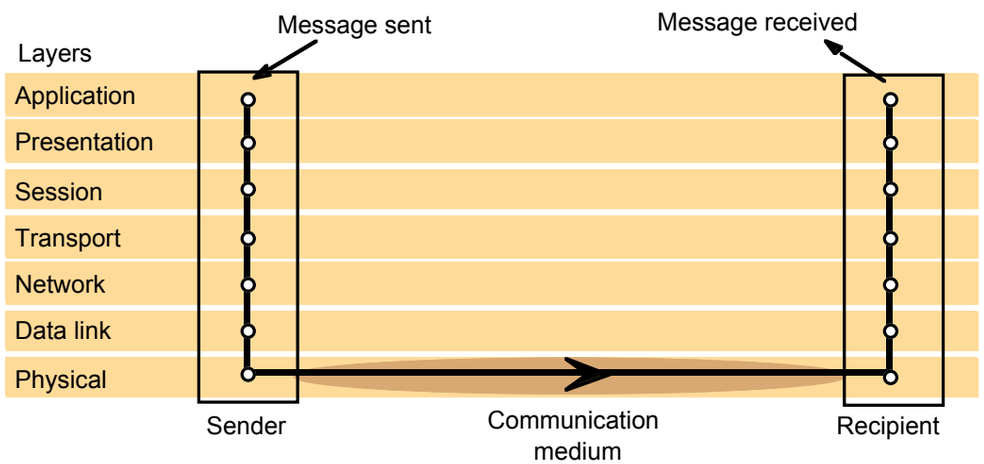
- E.g., TCP, UDP, SMTP, HTTP, …

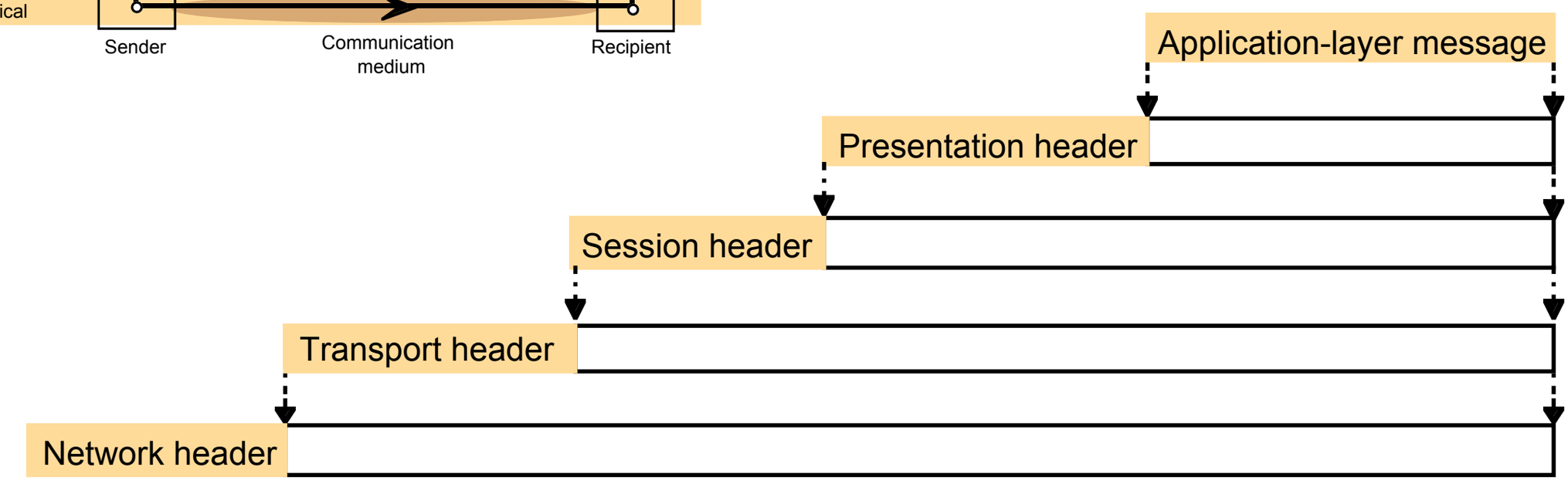
- Protocols are typically combined in *protocol stacks*, each layer protocol adding functionality.

Protocol layers



Protocol layers headers





Routing (distance vector)

Bellman's shortest path algorithm, 1957 - Distributed algorithm by Ford & Fulkerson 1962

Routings from A

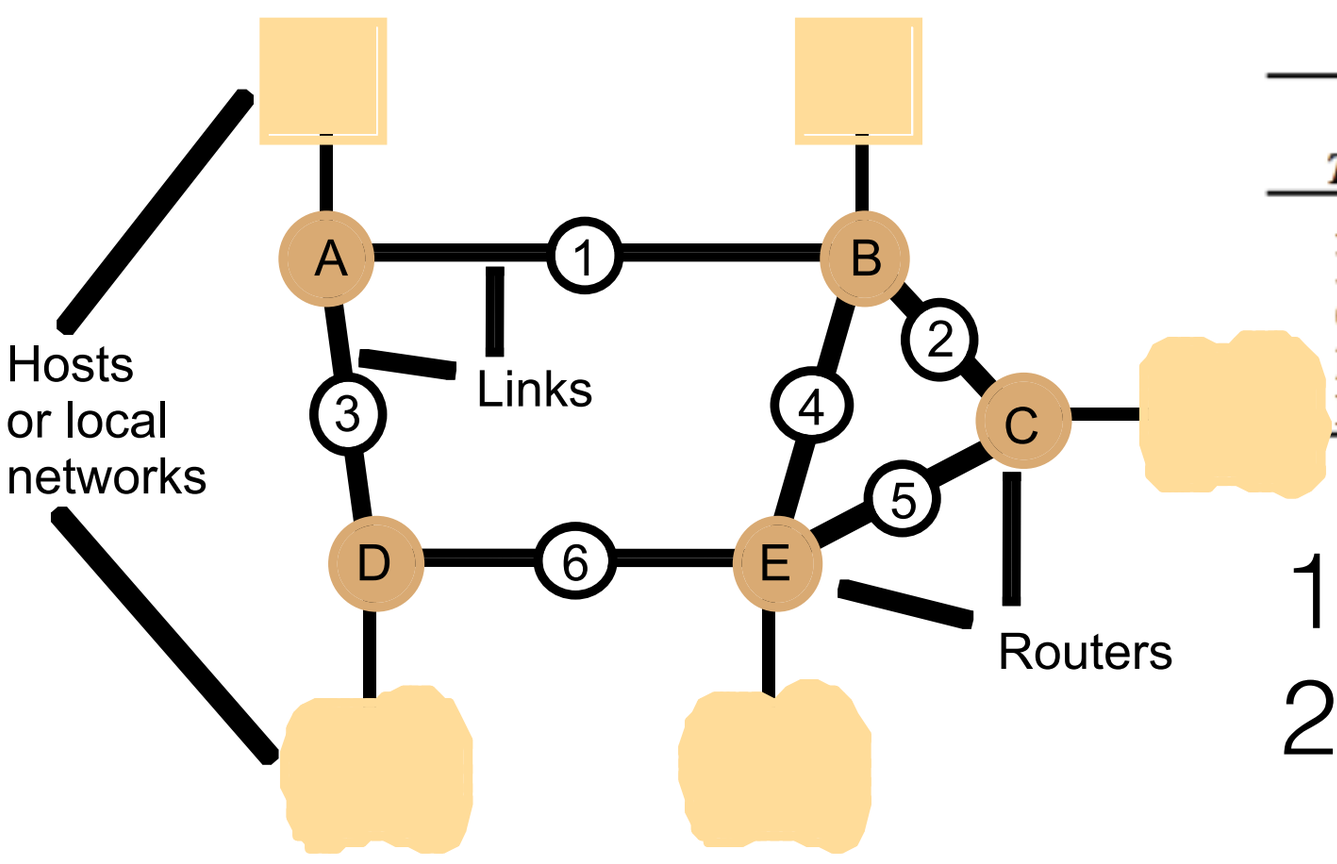
To	*Link*	*Cost*
A	local	0
B	1	1
C	1	2
D	3	1
E	1	2

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

1) Determine route
2) Update knowledge

Updating knowledge?

Bellman's shortest path algorithm, 1957 - Distributed algorithm by Ford & Fulkerson 1962

Routings from A

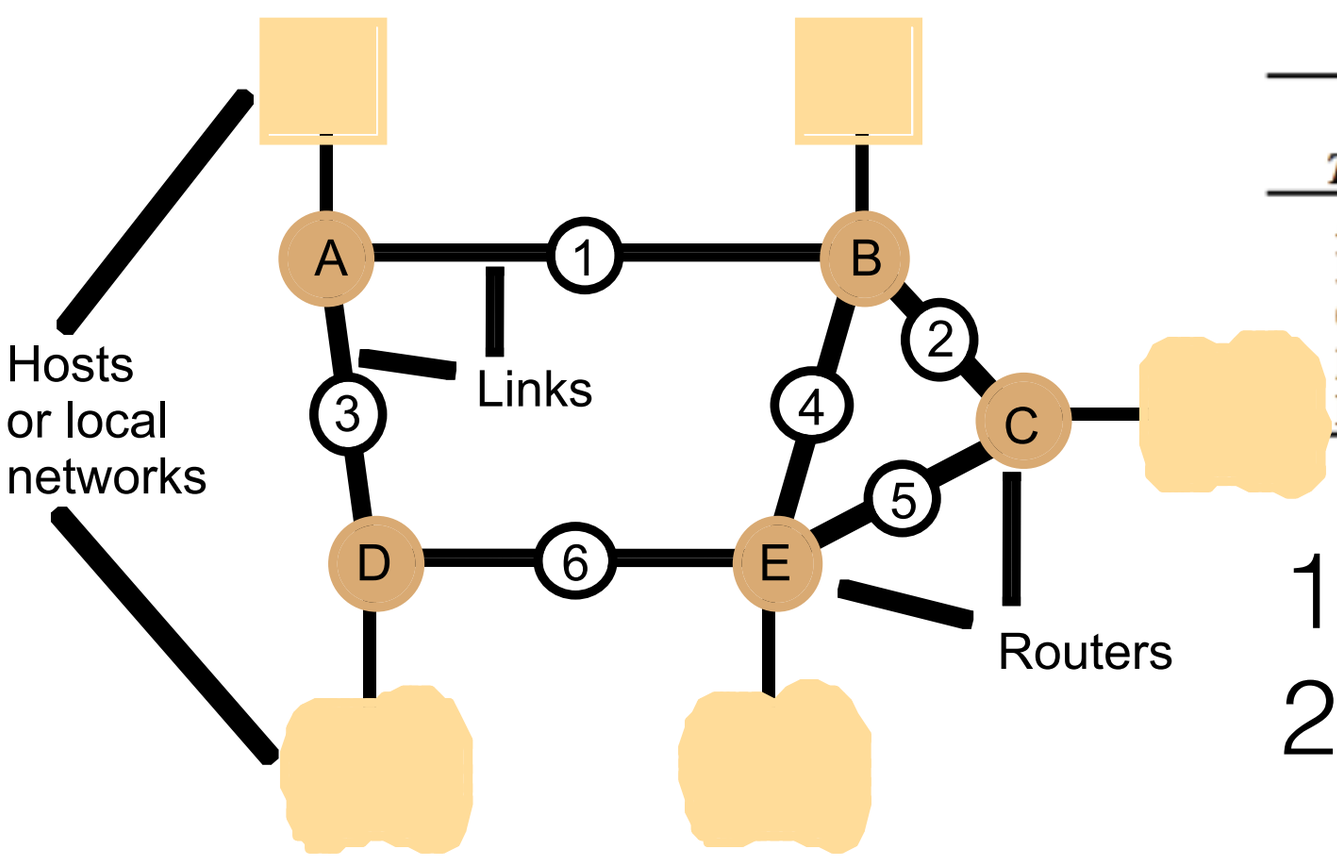
To	*Link*	*Cost*
A	local	0
B	1	1
C	1	2
D	3	1
E	1	2

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

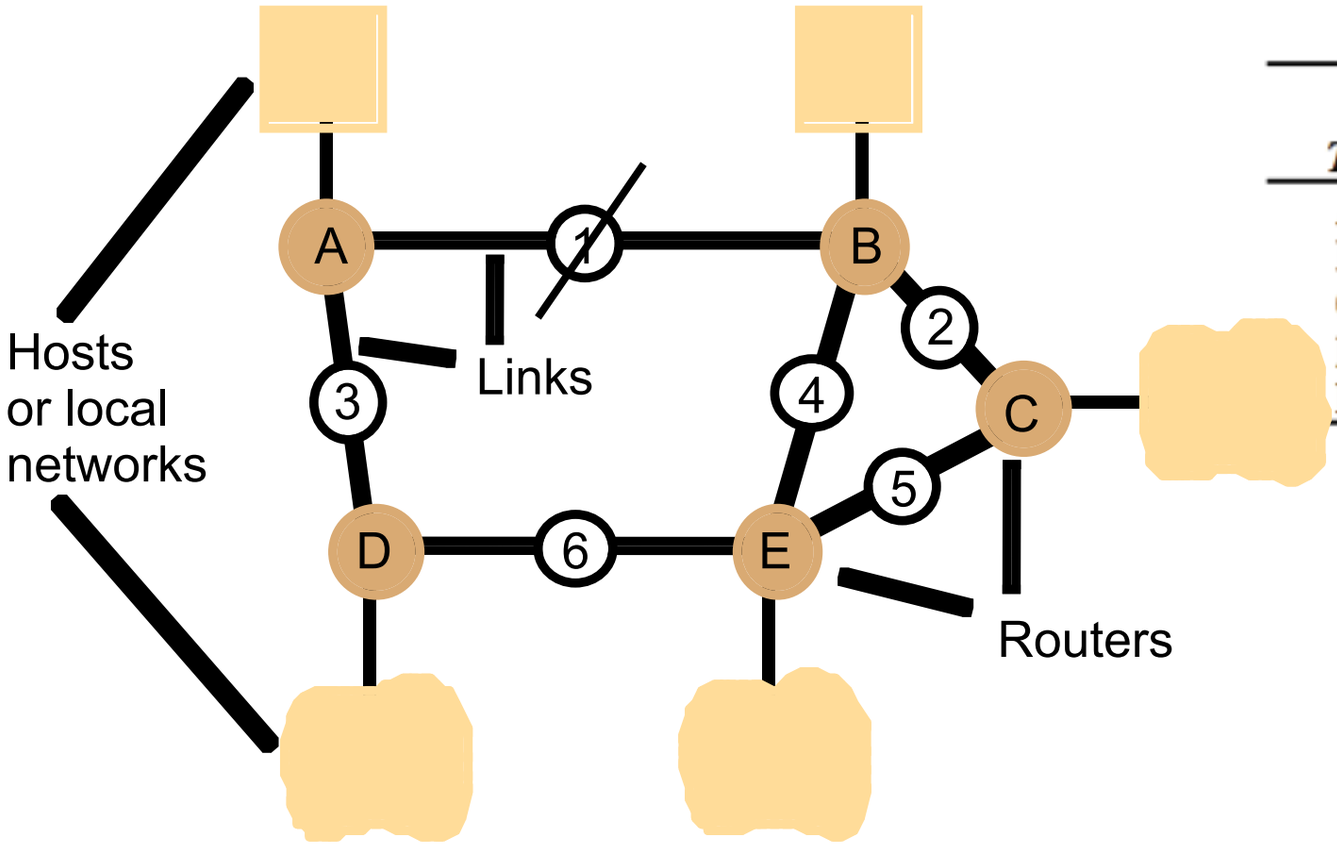
1) Determine route
2) Update knowledge

Routing

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

Routings from B		
To	*Link*	*Cost*
A	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

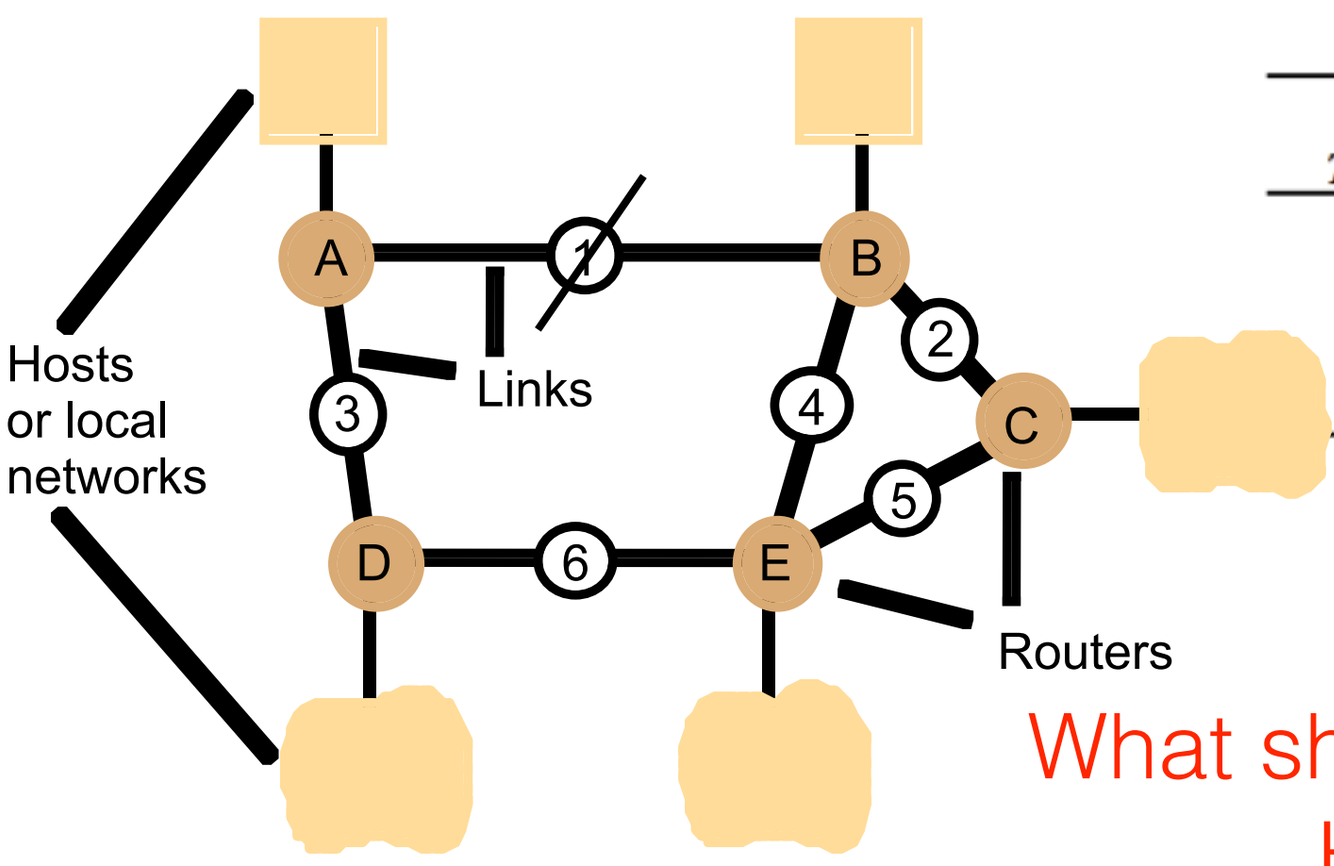
… routers then exchange updated tables

Routing

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

Routings from B		
To	*Link*	*Cost*
A	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



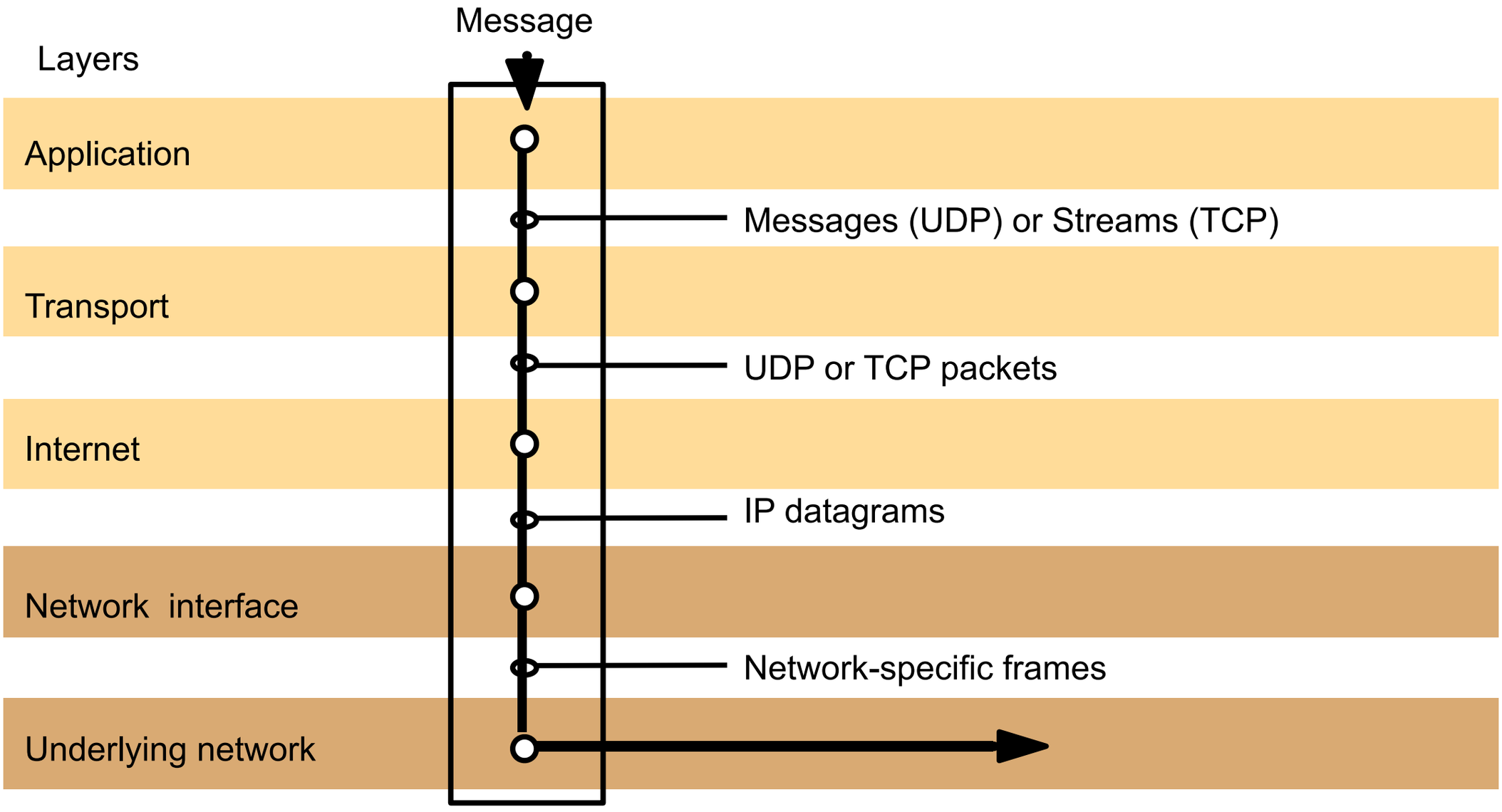
Routings from D		
To	*Link*	*Cost*
A	3	1
B	3	2
C	6	2
D	local	0
E	6	1

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

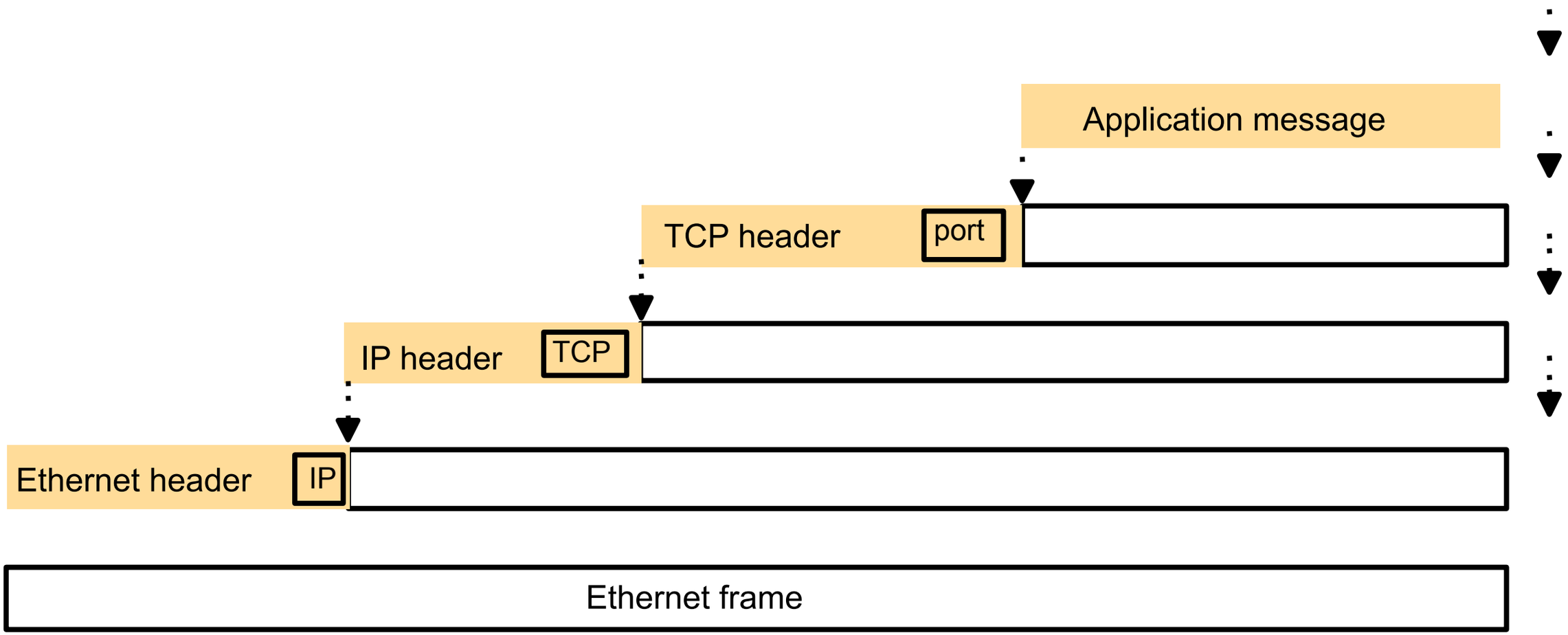
What should be the entry for E to A?
How can it learn that?

IP

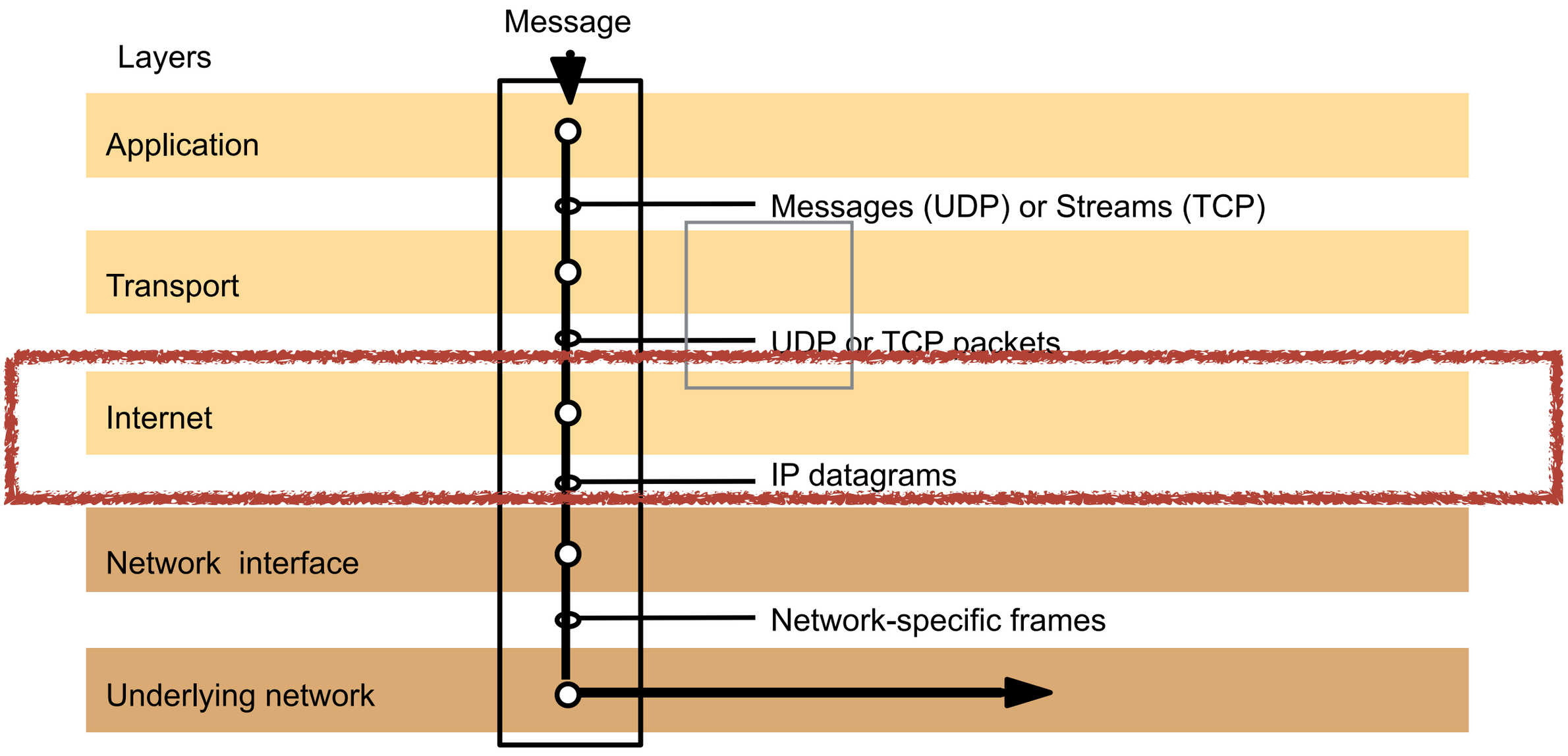
TCP/IP stack


Message
Layers
Application
Transport
Internet
Network interface
Underlying network
Messages (UDP) or Streams (TCP)
UDP or TCP packets
IP datagrams
Network-specific frames

TCP/IP stack headers



TCP/IP stack


Message
Layers
Application
Transport
Internet
Network interface
Underlying network
Messages (UDP) or Streams (TCP)
UDP or TCP packets
IP datagrams
Network-specific frames

IPv4

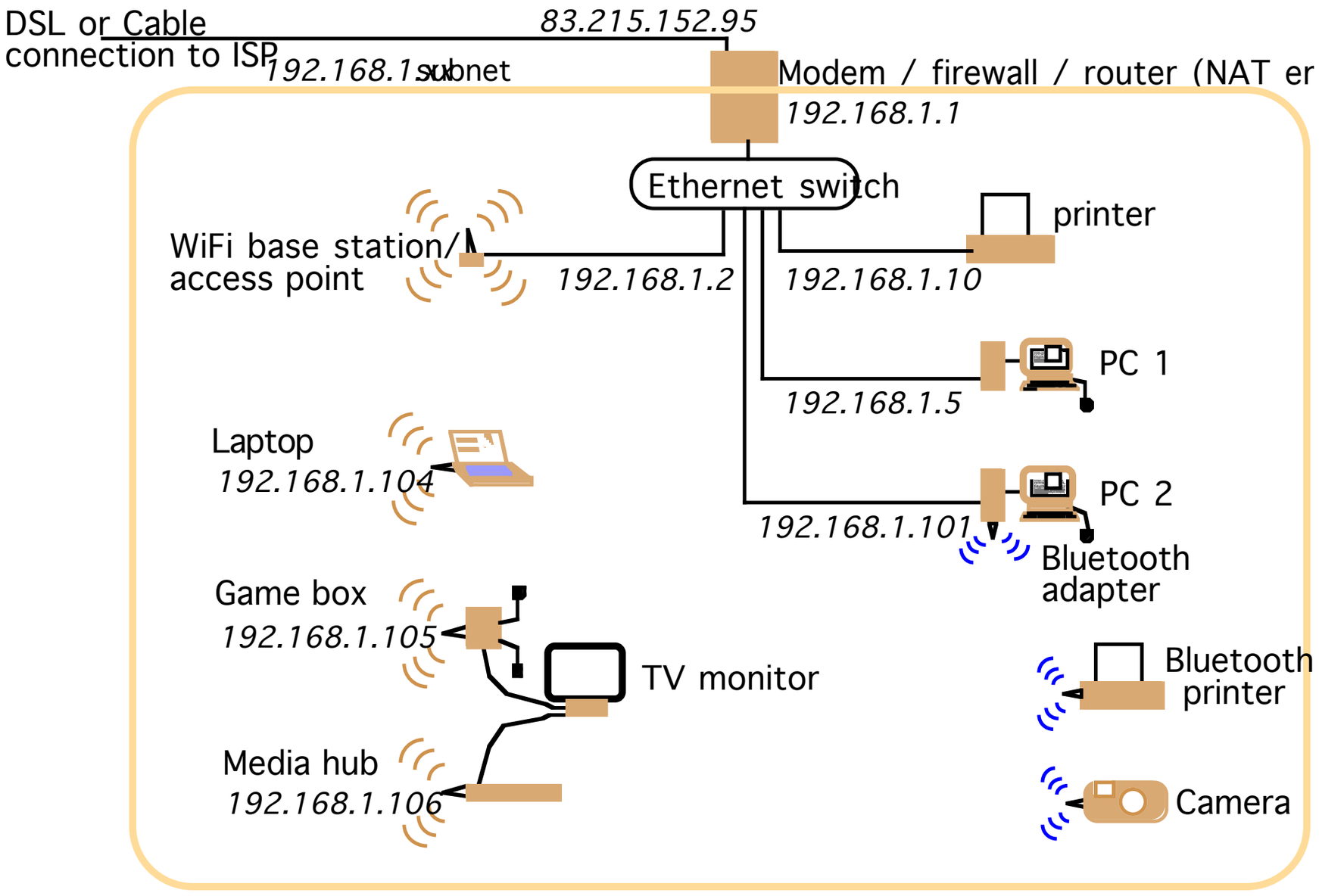
- Connection-less, unreliable, datagram protocol.
- Key functionality: Routing across distinct connected physical networks.
- IP-address: 4 byte logical address. E.g., 130.226.133.47
- CIDR: identifying IP ranges by IP-address + number of relevant bits in prefix: 130.226.132.0/23

IPv4 addresses

	octet 1	octet 2	octet 3		Range of addresses
	Network ID		Host ID		
Class A:	1 to 127	0 to 255	0 to 255	0 to 255	1.0.0.0 to 127.255.255.255
	Network ID		Host ID		
Class B:	128 to 191	0 to 255	0 to 255	0 to 255	128.0.0.0 to 191.255.255.255
	Network ID			Host ID	
Class C:	192 to 223	0 to 255	0 to 255	1 to 254	192.0.0.0 to 223.255.255.255
	Multicast address				
Class D (multicast):	224 to 239	0 to 255	0 to 255	1 to 254	224.0.0.0 to 239.255.255.255
Class E (reserved):	240 to 255	0 to 255	0 to 255	1 to 254	240.0.0.0 to 255.255.255.255

- 192.168.0.0/16, 10.0.0.0/8: private networks
- 127.0.0.0/8: loopback device
- 0.0.0.0/8: this host on this network
- 255.255.255.255/32: limited broadcast

Example network, NAT



MTU

- Maximum Transmission Unit
- Different physical links have different upper bounds on payload size
- IPv4 fragments packets on the fly

IPv4 Header

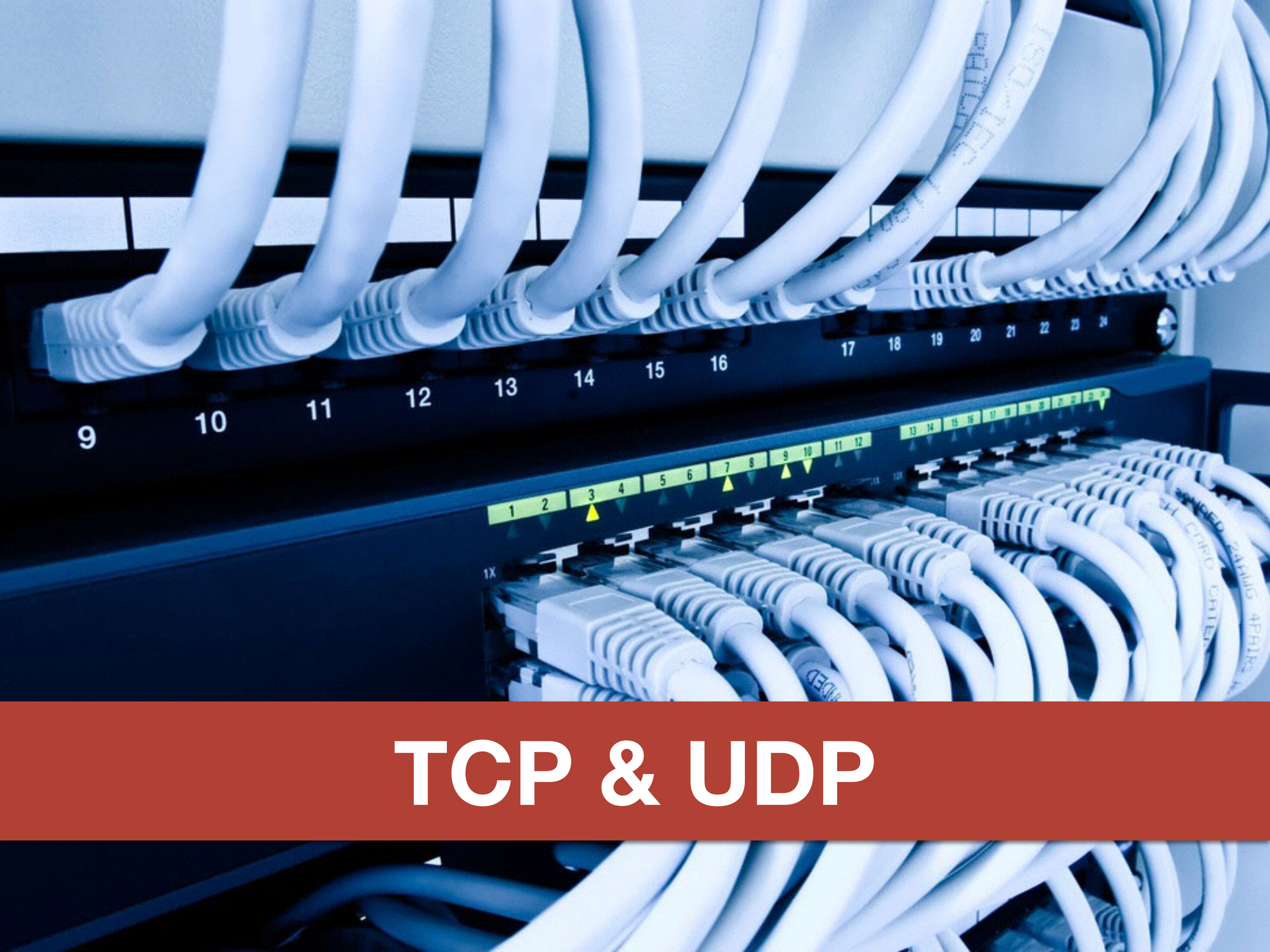
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags				Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

Practice

- Applications: TCP, UDP, SNMP, ICMP…
- Nobody likes numbers.
- Domain Names, e.g., “www.itu.dk” are translated to IP addresses by the “Domain Name System”.

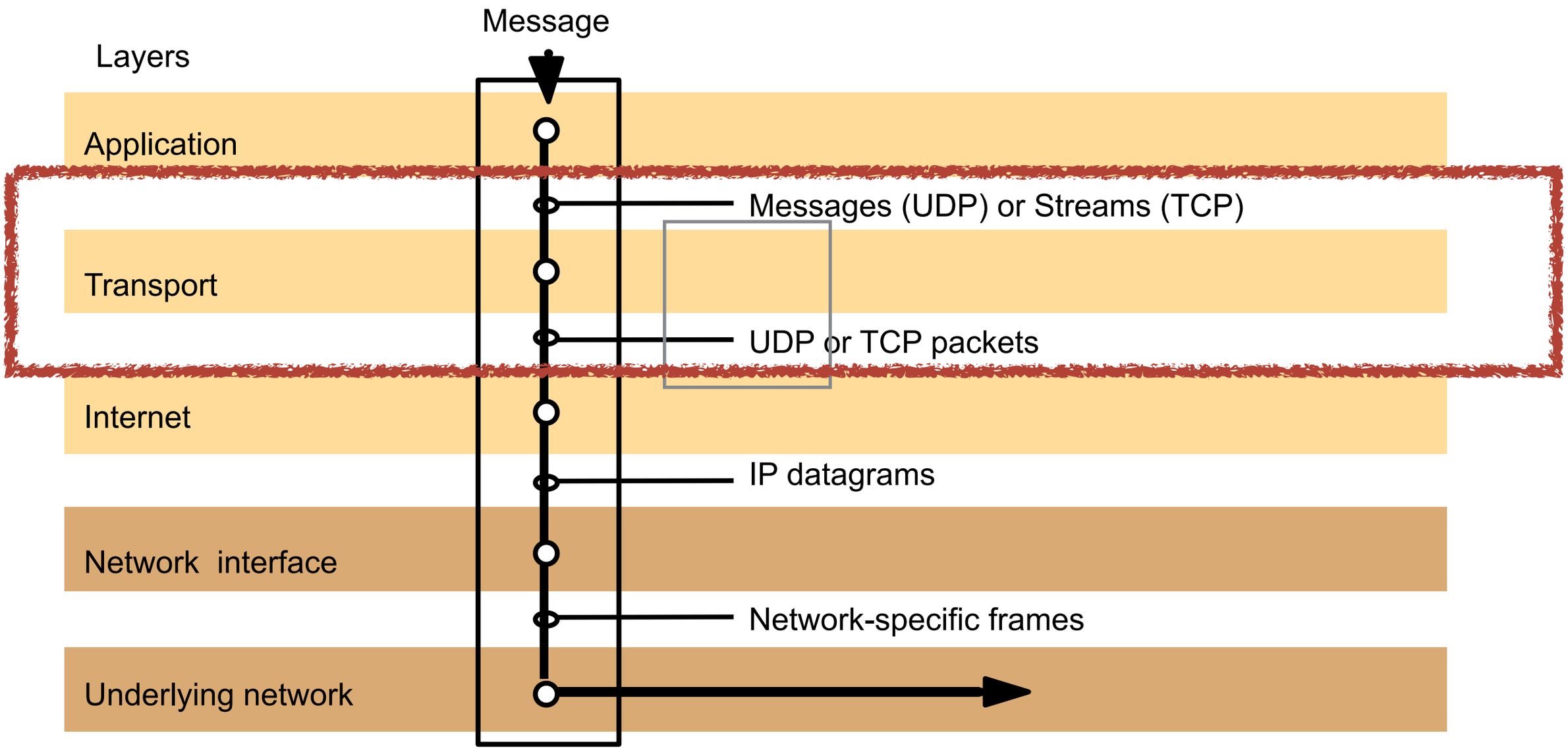
Practice

- Observe latency, connectivity using “ping”.
- Observe routes using “traceroute”.



TCP & UDP

TCP/IP stack

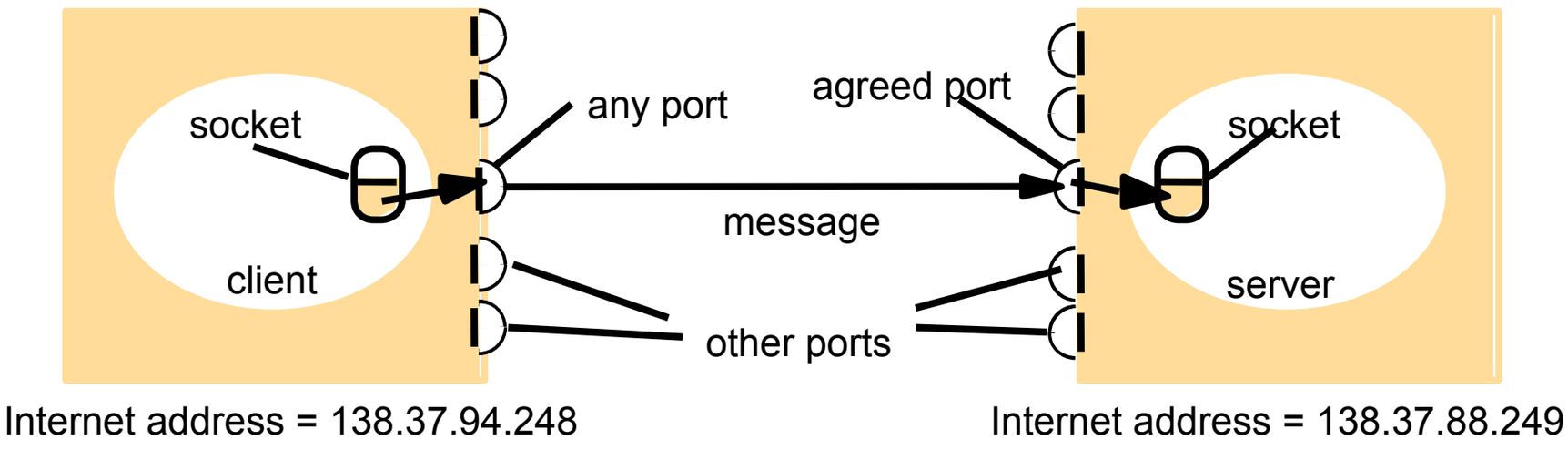

Message
Layers
Application
Transport
Internet
Network interface
Underlying network
Messages (UDP) or Streams (TCP)
UDP or TCP packets
IP datagrams
Network-specific frames

Ports in TCP/UDP

- Processes on a host are addressed by *ports.* A process is *bound* to a port.

- TCP/UDP protocols are used through a programming abstraction called a *socket*.

- Binding to ports 1-1023 require root privileges.



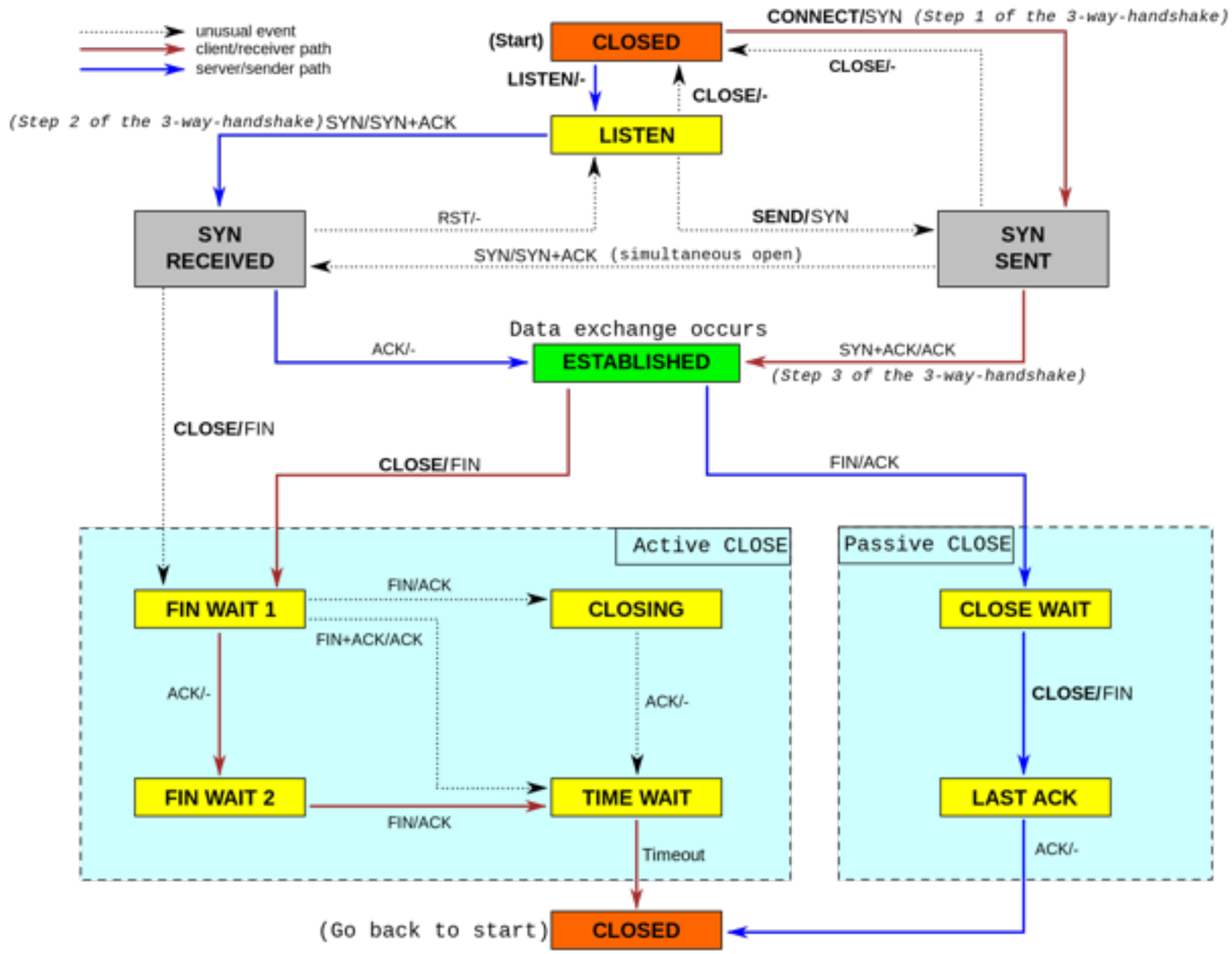
UDP

- Connection-less, unreliable, datagram-protocol.

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

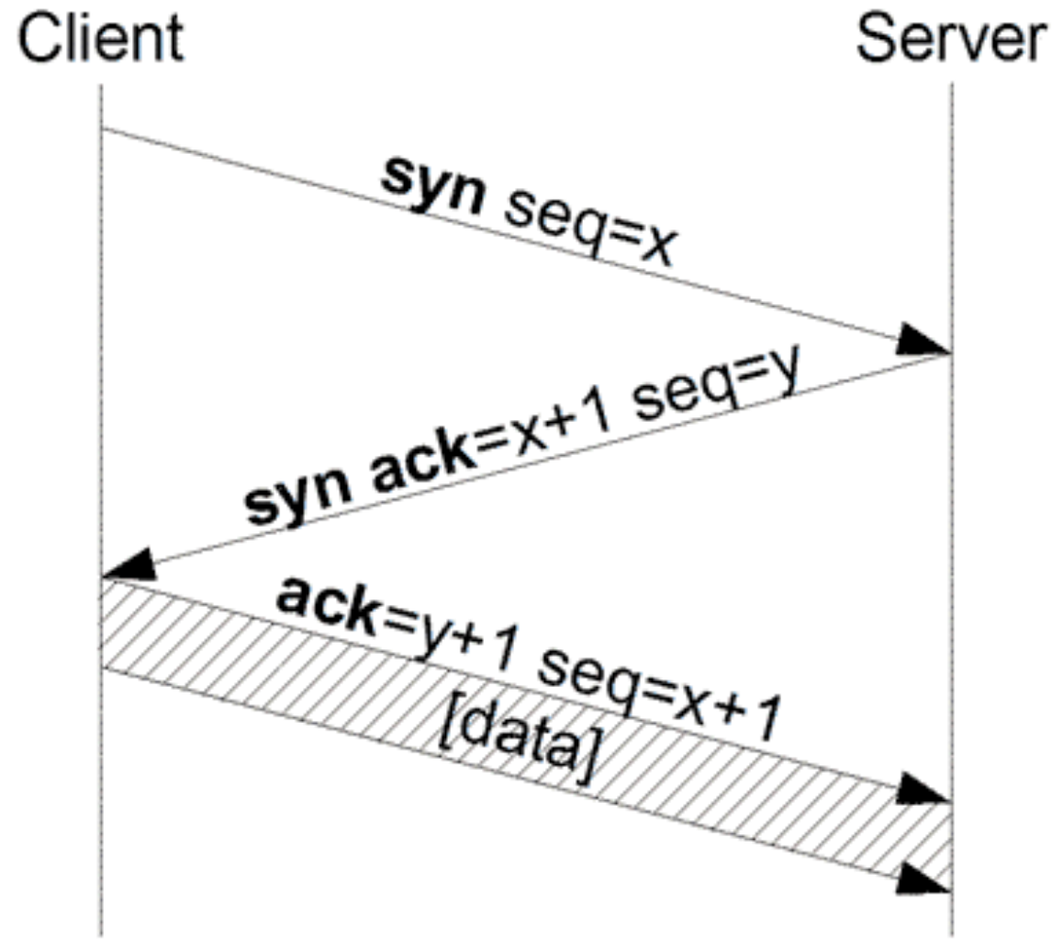
TCP

- Connection-oriented, reliable, streaming protocol.
- Specified as a fairly complex state machine:



Connection-oriented

- 3-way handshake
- A TCP connection is identified by all of (ip1, port1, ip2, port2).
- … so, a web-server at 130.226.133.47:80 can have multiple connections at that single port.



Reliable/streaming

- Sequence numbers, timeouts, acknowledgments, re-transmissions.

- Sliding window protocol, piggy-backing

TCP

- ACK flag: acknowledgments.
- SYN flag: synchronise sequence numbers
- RST flag: I’m not talking to you.

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																	Urgent pointer (if URG set)														
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

Practice

- Web-servers/HTTP uses TCP on port 80.
- E-mail/SMTP uses TCP on port 25.
- DNS lookups uses UDP on port 53.

Practice

- Observe traffic using “tcpdump”.
- Send/receive over UDP datagrams using “nc -u”.
- Connect/bind/communicate over TCP using “nc”.
- Portscanning, “nmap".

Marshalling

Serialize this!

```
public class Person {
   private String name;
   private String place;
   private int Year;
   // ctors, accessors omitted
}
```

Java Object Serialization

Serialized values				*Explanation*
Person	8-byte version number		h0	*class name, version number*
3	int year	java.lang.String name:	java.lang.String place:	*number, type and name of instance variables*
1934	5 Smith	6 London	h1	*values of instance variables*

The true serialized form contains additional type markers; h0 and h1 are handles

- public class Person implements Serializable {
- ObjectOutputStream og ObjectInputStream
- “Deep copy” (pointers->handles)
- Transient variables (e.g. references to files)
- Reflection

```
public class Person {
    private String name;
    private String place;
    private int Year;
    // ctors, accessors om
}
```

Serialization of general data structures

- CORBA

index in sequence of bytes	← 4 bytes →	*notes on representation*
0–3	5	*length of string*
4–7	"Smit"	*'Smith'*
8–11	"h "	
12–15	6	*length of string*
16–19	"Lond"	*'London'*
20-23	"on "	
24–27	1984	*unsigned long*

The flattened form represents a *Person* struct with value: {'Smith', 'London', 1984}

- Java Object Serialization

Serialized values				*Explanation*
Person	8-byte version number		h0	*class name, version number*
3	int year	java.lang.String name:	java.lang.String place:	*number, type and name of instance variables*
1984	5 Smith	6 London	h1	*values of instance variables*

The true serialized form contains additional type markers; h0 and h1 are handles

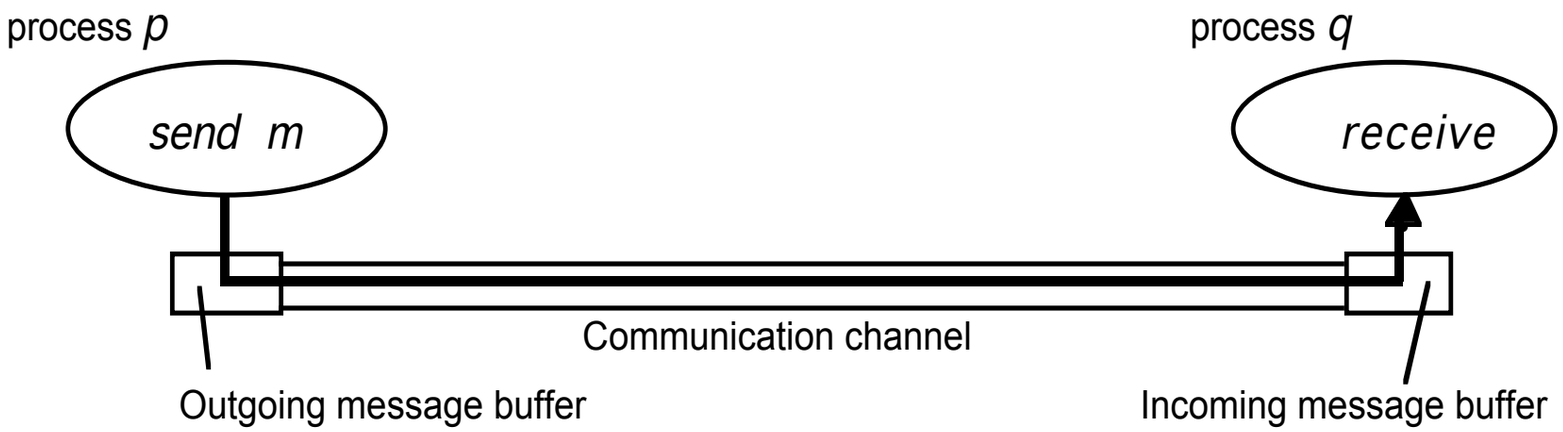
- XML

```
<person id="123456789">
        <name>Smith</name>
        <place>London</place>
        <year>1984</year>
        <!-- a comment -->
</person >
```

Summary

Foundations of Networking

- Terminology: Processes, channels, failures.



- Protocol layers, each protocol adding functionality

- Protocol layer headers.

TCP/IP

- IP: Routing, fragmentation.
  IP-address, CIDR, NAT.

- TCP/UDP: Ports, sockets.

- UDP: Datagrams. Adds checksum to IP.

- TCP: Reliability, congestion control, connections.
  3-way handshake, SYN.
  State machine.

Networking

- Different kinds of network, different characteristics
- Network protocols implemented in layers, providing different guarantees and abstraction
- IPv4 -> v6: unexpected growth and mobility
- UDP (messages) and TCP/IP (streams) transport layer protocols used for Internet communication
- External data representation: Serialization/Marshalling and De-serialization/unmarshalling

Win chocolate & the admiration of your peers!

Questions?

Mini-project 1

Mini-project 1

- Java or your choice of language
- Design as a group, program individually or in subgroups.
- When programming reveals your design defective, reconvene as a group.
- Don’t try to do it in one session.

Questions?

Parting words

Do it yourself!

- Read chapter 1, 3, 4.
- I didn't cover everything, e.g., exact distance vector routing protocol.
- Yet it is in the curriculum.
- Also, exercises!
- Form groups

One must learn by doing the thing;
for though you think you know it,
you have no certainty, until you try.

Sophocles, 496-405 BC.