MONASH University

Information Technology

FIT5183: Mobile and Distributed Computing Systems (MDCS)

Lecture 1B
Introduction to Mobile and Distributed Computing

Today’s Lecture

- ❑ Overview of Distributed Systems
- ❑ Historical Development
- ❑ Distributed Computing Models:

  Clusters, *Grids* and *Cloud* Computing
- ❑ Distributed Software Architectural Styles
- ❑ Distributed Systems Architectures
- ❑ Challenges and Design Issues

References

- Distributed Systems: Principles and Paradigms (2nd Edition), Tanenbaum & Van Steen, 2014, Pearson
- Distributed Systems: Concepts and Design (5th Edition), Coulouris, Dollimore, Kindberg & Blair, 2012, Addison-Wesley
- Web Services: Concepts, Architectures and Applications, Alonso G., et. al., 2010, Springer

Supplementary:

- Computer Networking: A Top-Down Approach, (7th Edition), Kurose J. and Ross K., 2017, Pearson Addison-Wesley
- Business Data Communications & Networking (10th Edition), FitzGerald, J. & Dennis, A. 2009, John Wiley & Sons

Selected Figures in notes drawn from above texts for educational purposes.

Distributed Computing Systems Definitions and Concepts

Definition of a Distributed System

- *“A distributed system is a collection of independent computers that appears to its users as a single coherent system”* (Tanenbaum & Van Steen, 2014, pg 2)

- *A system “in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages”* (Coulouris, Dollimore, Kindberg & Blair, 2012, pg 2)

Distributed Computing

- A computing paradigm where a number of autonomous machines (not necessarily homogeneous)

  which are geographically distributed/separated

  can communicate and exchange/transfer data

  through a computer network

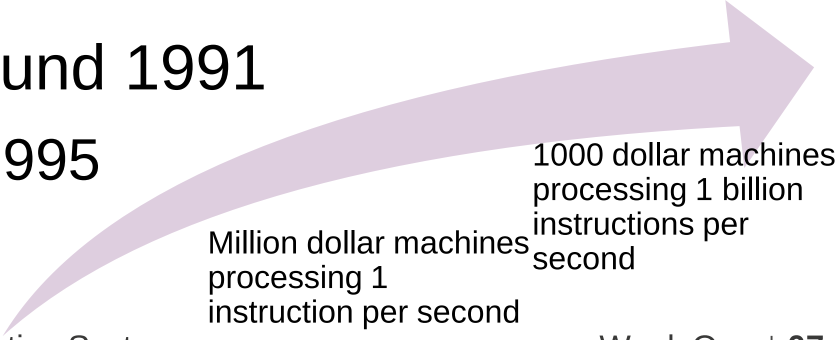
  to perform certain related tasks

- Distribution of machines physically
- Distribution of data, processing, control, business logic/rules, …

History of Distributed Systems (condensed version)

Early distributed systems (1945-1980s):

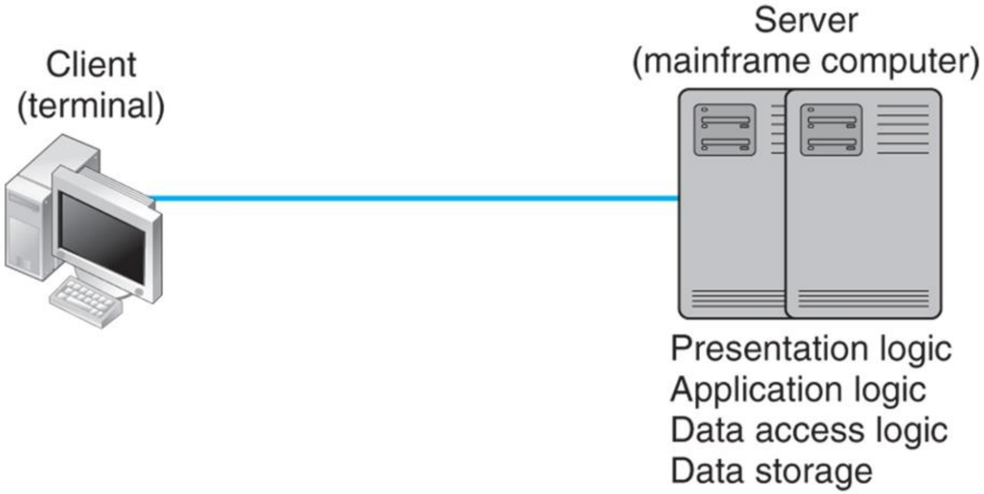
- Expensive computers, mainframes and minicomputers
- Limited distribution, dumb terminals
- The major changes from mid 1980s:
  - Emergence of microprocessors (starting with 8-bit machines)
  - LAN and WAN connecting large number of machines and enabling data transfer at high speed between the machines
- The internet was introduced around 1991
- Java language was introduced in 1995



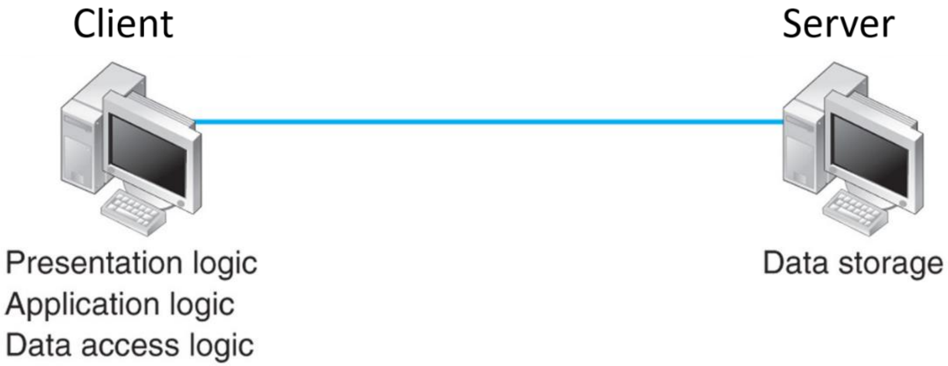
Million dollar machines processing 1 instruction per second

1000 dollar machines processing 1 billion instructions per second

Early Distributed LAN and Internet Systems



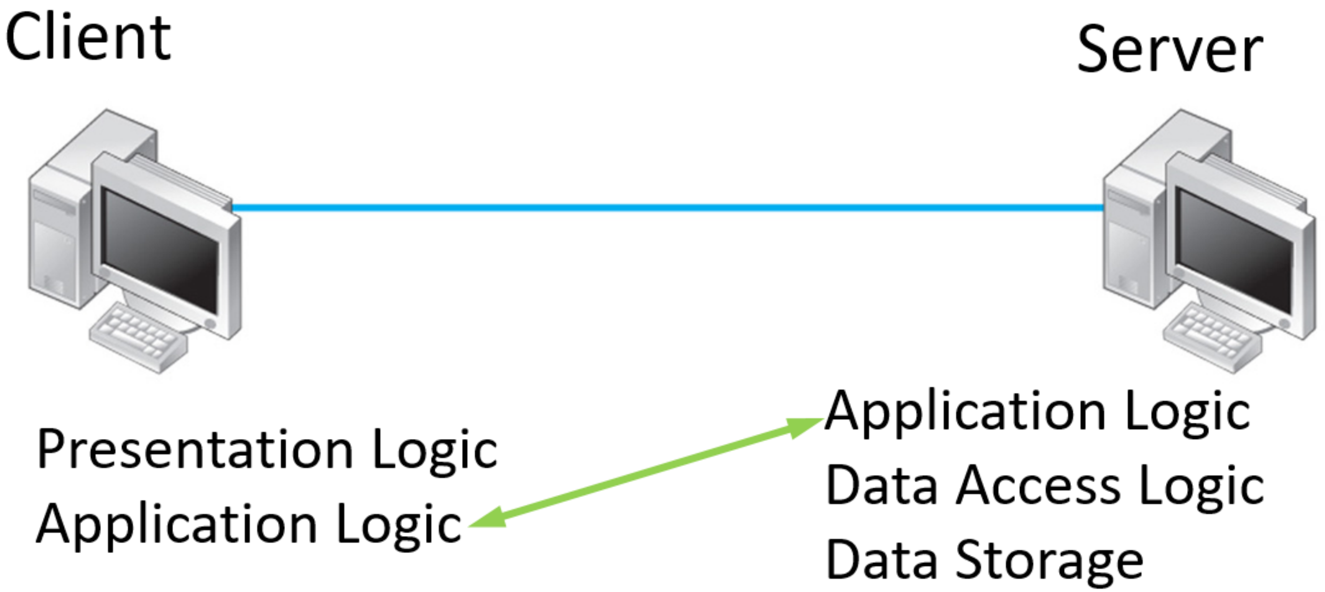
- *1960s*: Server (host mainframe) based system connected with dumb terminals. **All** processing performed on mainframe serving connected client; basically monitor + keyboard.



FitzGerald & Dennis (2009)

- *1980s*: Client (PC) based system leveraging processing power of personal computers. Server requests could require transfer of an **entire** database. Early word-processors worked this way.

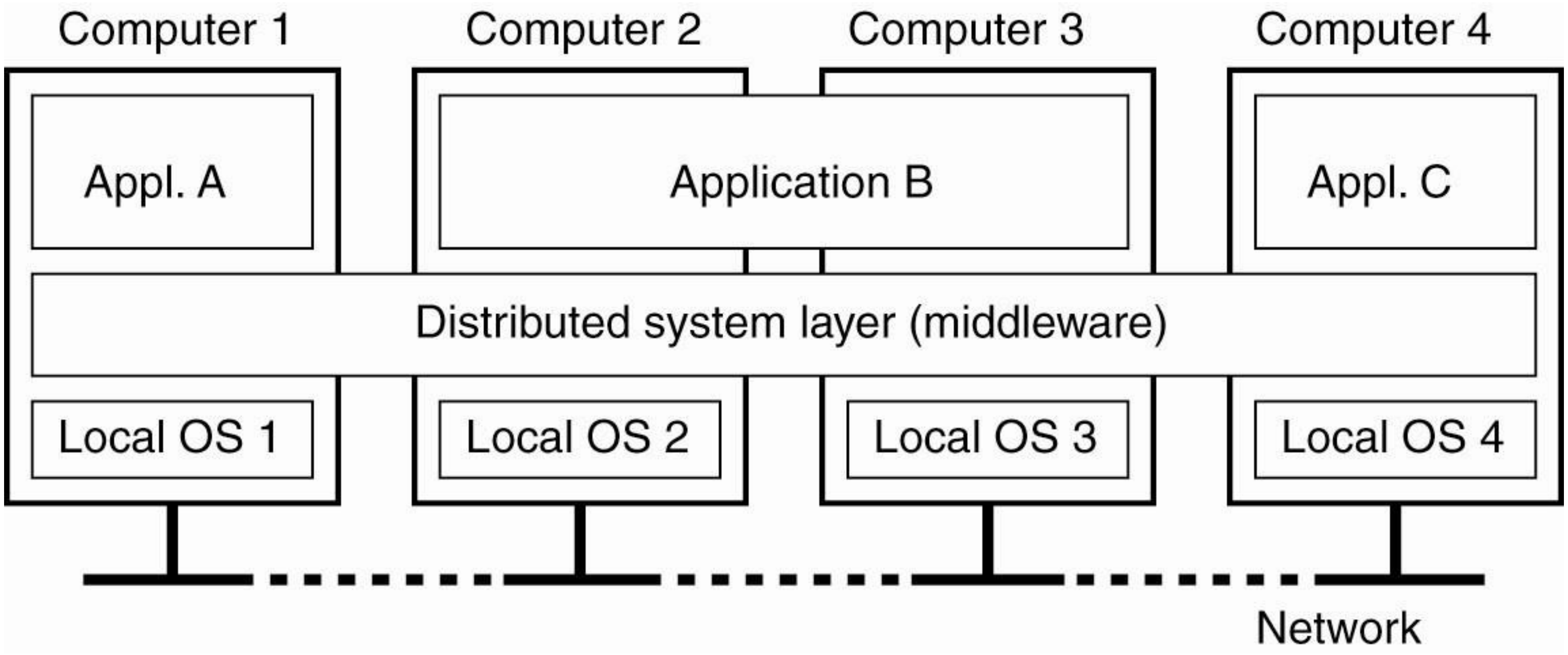
Evolution of the Client-Server Model



FitzGerald & Dennis (2009)

- *1990s*: The traditional “2-tier” Client-Server models attempt to **balance processing load** across Client and Server computers.
- Many variations of this highly successful model with multiple clients, multiple servers and introduction of intermediate middleware tiers to provide **abstract interfaces**, resource **virtualisation** and **indirection**.

An Example of a Distributed System



- A distributed system organised as middleware. The middleware layer extends over multiple machines, and offers each application the same interface.

Distributed Computing Models

Distributed Computing Models

Evolution and combination of different computing models and concepts:

- **Parallel processing** in high performance computer systems
- Computer **clusters**, **Grid** computing, **Cloud** computing, …
- **Virtualization (虚拟化)** to manage lots of computing and storage resources
- **Utility computing** for packaging computing and storage resources as services
- High-level communication of applications through **web services**
- **Mobile** and **pervasive** distributed computing systems (more about these later)

Grid Computing

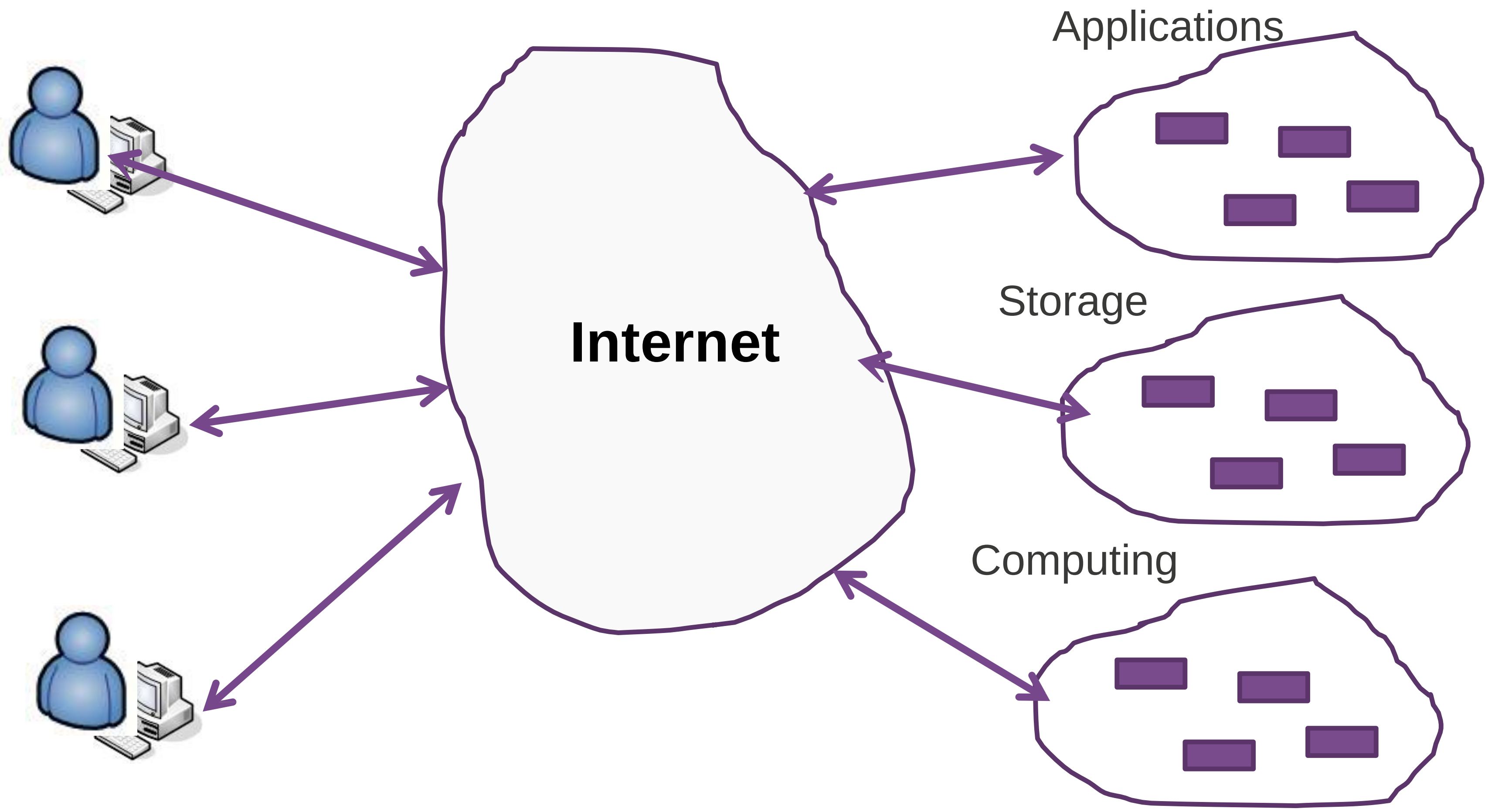
- Several computers connected together to solve large problems (e.g. processing scientific data)
- Heterogeneous (H/W and O/S) and autonomous
- Physically, widely distributed over a LAN, MAN, WAN or Internet and at a larger scale compared to cluster computing
- Can use any spare computer or node's computing power in an opportunistic way when available
- Some challenges:
  - resource allocation
  - load balancing
  - task scheduling

Examples of Grid-based Distributed Computing

- SETI (Search for extra-terrestrial Intelligence) @Home project to detect intelligent life outside Earth
  http://setiathome.berkeley.edu/

- Folding@home for disease research that simulates protein folding
  http://folding.stanford.edu/

Cloud Computing



Cloud Computing Definitions

“Cloud computing is a type of **Internet-based computing** that provides **shared** computer processing **resources and data** to computers and other devices **on demand**.” (Wikipedia)

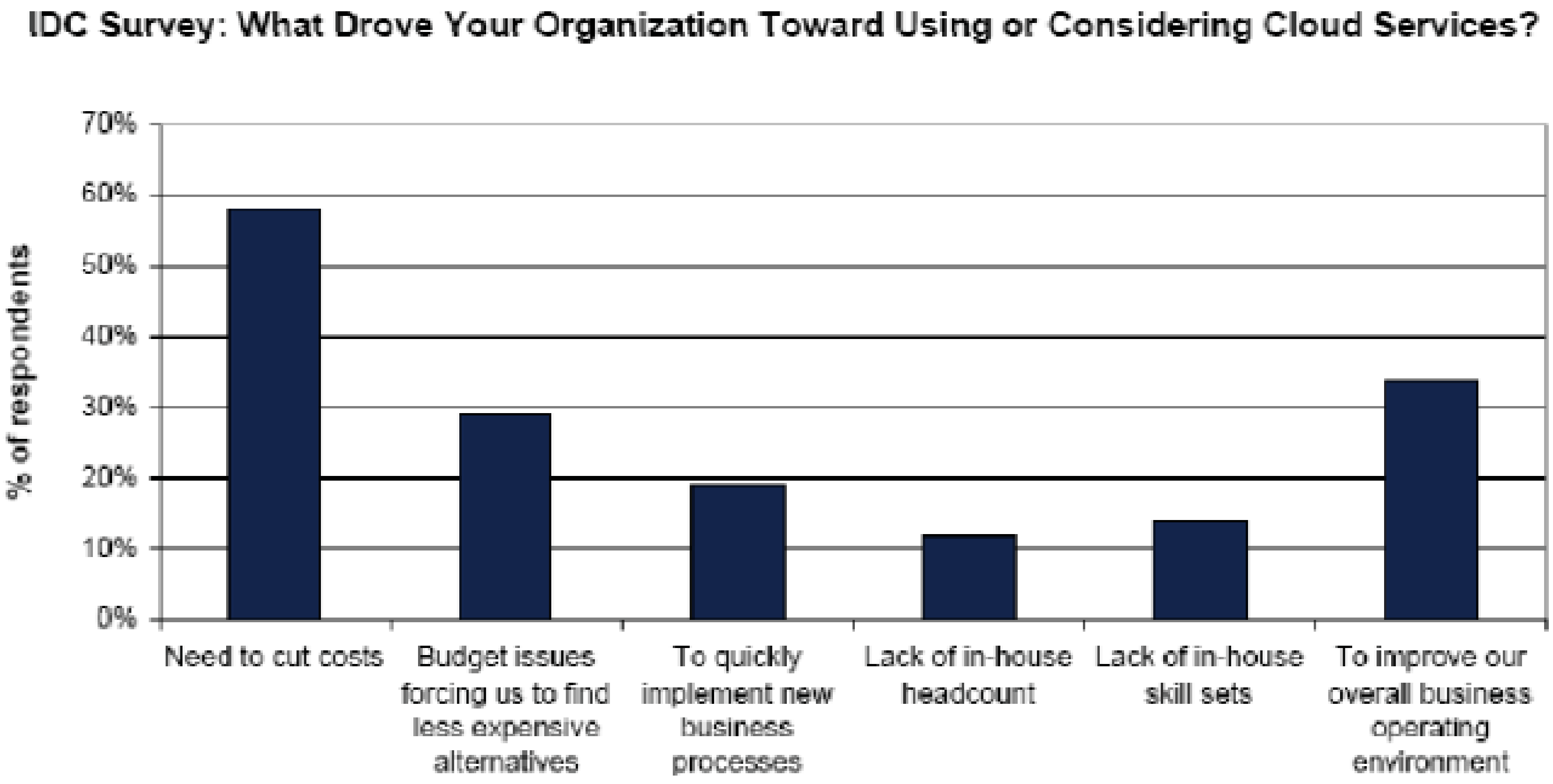
“Cloud computing means **storing and accessing data** and programs **over the Internet** instead of your computer's hard drive.” (E. Griffith, PC Magazine, 2016)

“Cloud computing, often referred to as simply “the Cloud,” is the delivery of **on-demand computing resources**—everything from applications to **data centers—over the Internet** on a **pay-for-use basis**.“ (IBM)

Cloud Computing Features

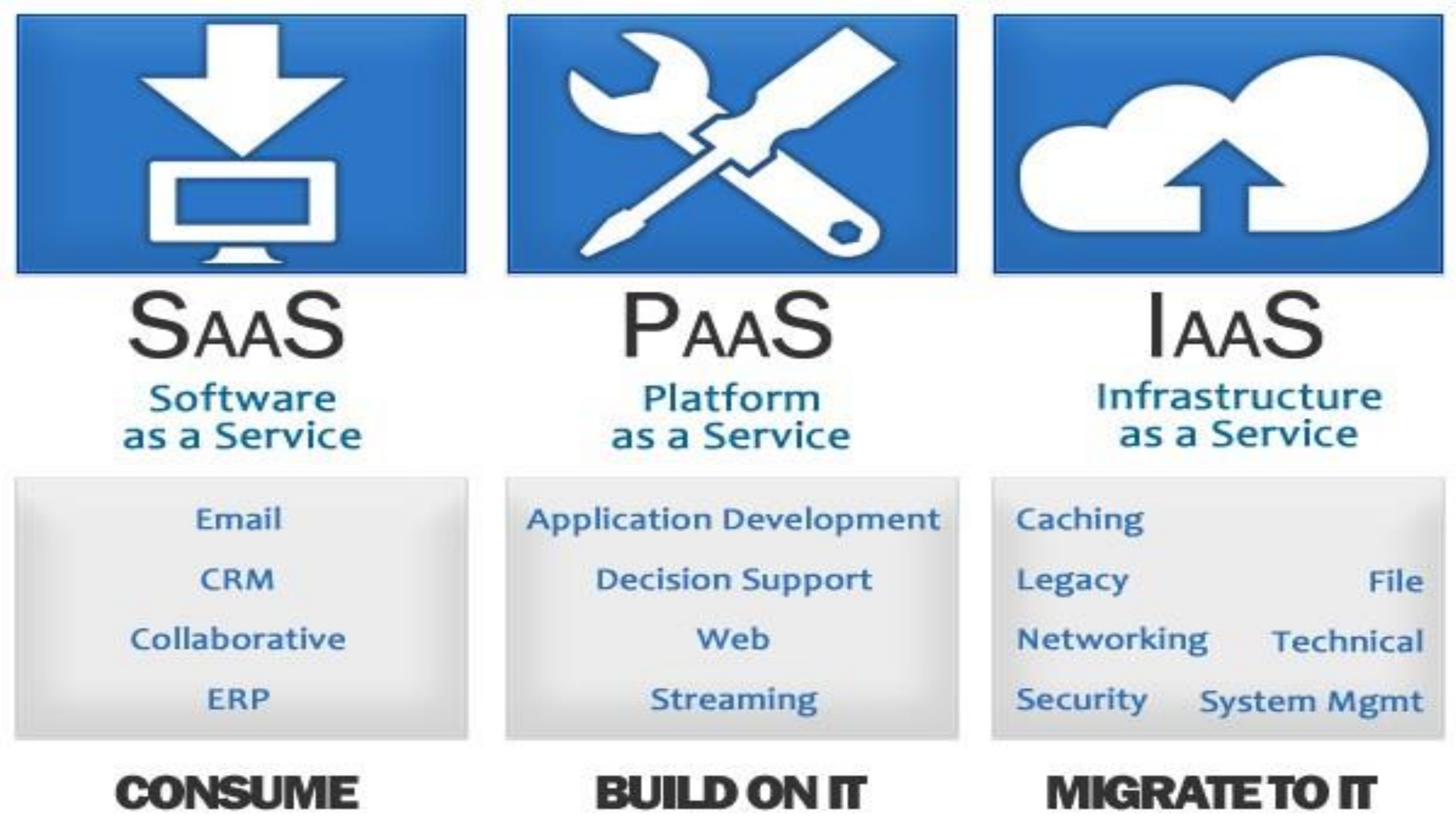
- A vision of **on demand**, **reliable** services provided over the Internet (i.e. cloud)
- A large network of computer, servers, data centres. Provides an easy access to **virtually infinite computing**, **storage** and **networking** resources
- **Pay-per-use** (utility computing)
- **Elasticity** (scaling up and down as needed)
- **Infrastructure**, **platform**, and **application** software as services
- **Low cost**: save on hardware costs, maintenance costs, software licenses, etc.

Motivation for Cloud Computing



Alonso, G. Cloud Computing Presentation, JISBD (2010)

Cloud Layers



SAAS

Software as a Service

Email

CRM

Collaborative

ERP

CONSUME

PAAS

Platform as a Service

Application Development

Decision Support

Web

Streaming

BUILD ON IT

IAAS

Infrastructure as a Service

Caching

Legacy

File

Networking

Technical

Security

System Mgmt

MIGRATE TO IT

SaaS – Software as a Service

- Applications run on the cloud and **users can connect from any computer to access them using a web browser**
- No need to buy, install, update, and maintain.
- Highest margin for Providers; Also highest “vendor lock-in”
- Examples:
  - Salesforce.com
  - Google apps (such as Gmail and Google docs)
- It is beneficial:
  - When there is a short term need
  - When there is a regular rise and fall in demand
  - When there is a need for web and mobile access

PaaS – Platform as a Service

- Provides the computing platform for building and hosting applications on the web
- **Virtualised Hardware with Basic Software**
- Suitable for easier and faster creation of web applications
- Suitable for collaborative development
- Suitable for testing and deployment
- Examples:
  - Google App Engine: It allows you to host your web application on Google servers
  - Microsoft Azure

IaaS – Infrastructure as a Service

- **Provides the infrastructure including networking**, **computers** (virtualized grids or dedicated hardware) **and storage**
- Highest level of flexibility and control over IT resources
  - Allows the users to control the underlying cloud infrastructure including network, servers, operating systems, or storage
- Lowest margin for Providers. Also least “vendor lock-in”
- Suitable for new business as no need to buy hardware, servers, storage
- Also suitable for growing businesses, trials or temporary business needs
- Examples: Amazon Web Services (AWS)
- Sometimes there is overlap between IaaS and PaaS

Different Types of Cloud

- ❑ Public cloud

  All the hardware, software, or supporting infrastructure, are owned and managed by providers and are accessed over the Internet

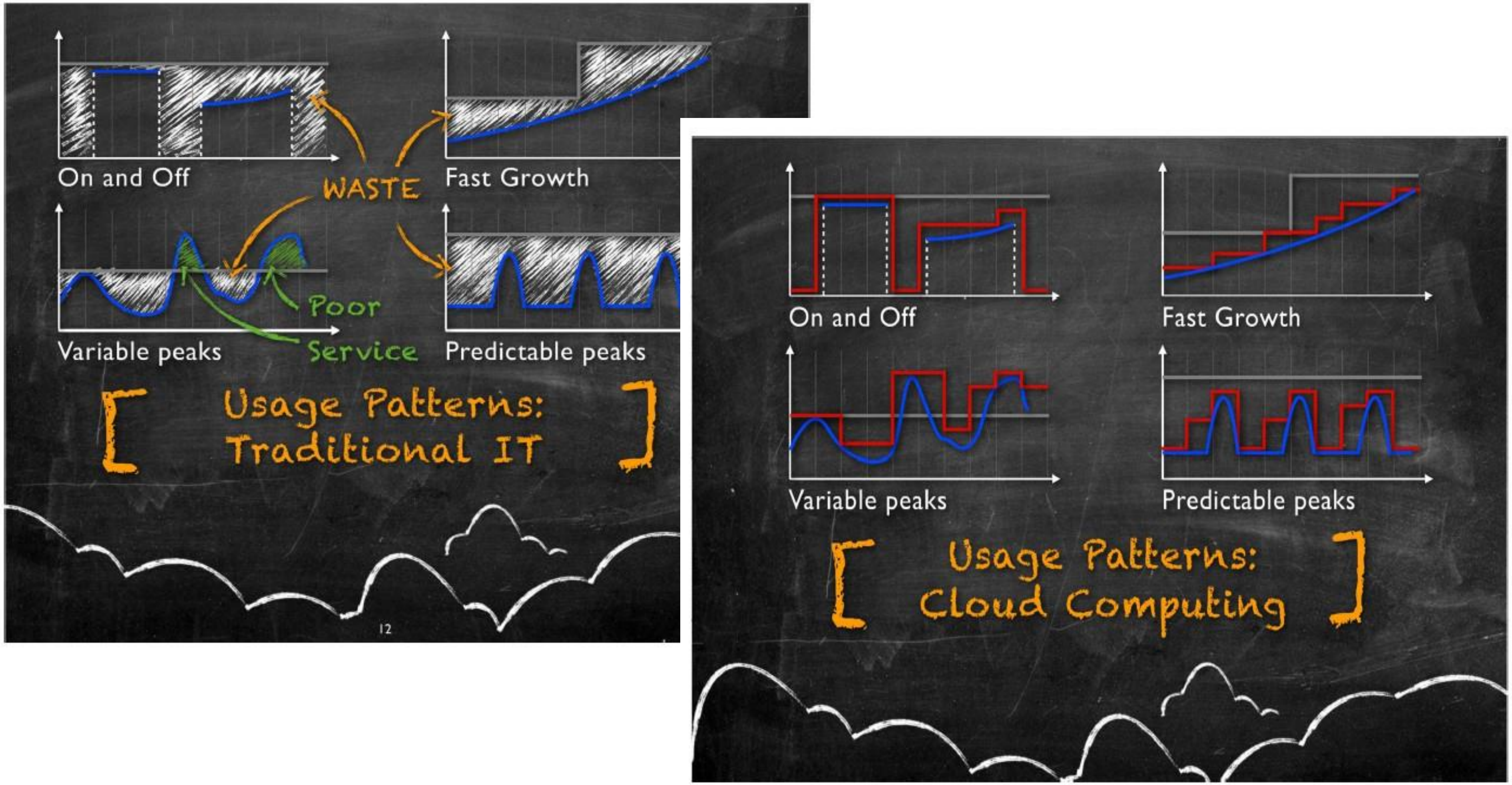
- ❑ Private cloud

  It is operated on a private network by an organisation (not shared with any other organisations), and hosted on-premise internally or externally by a third party

- ❑ Hybrid cloud

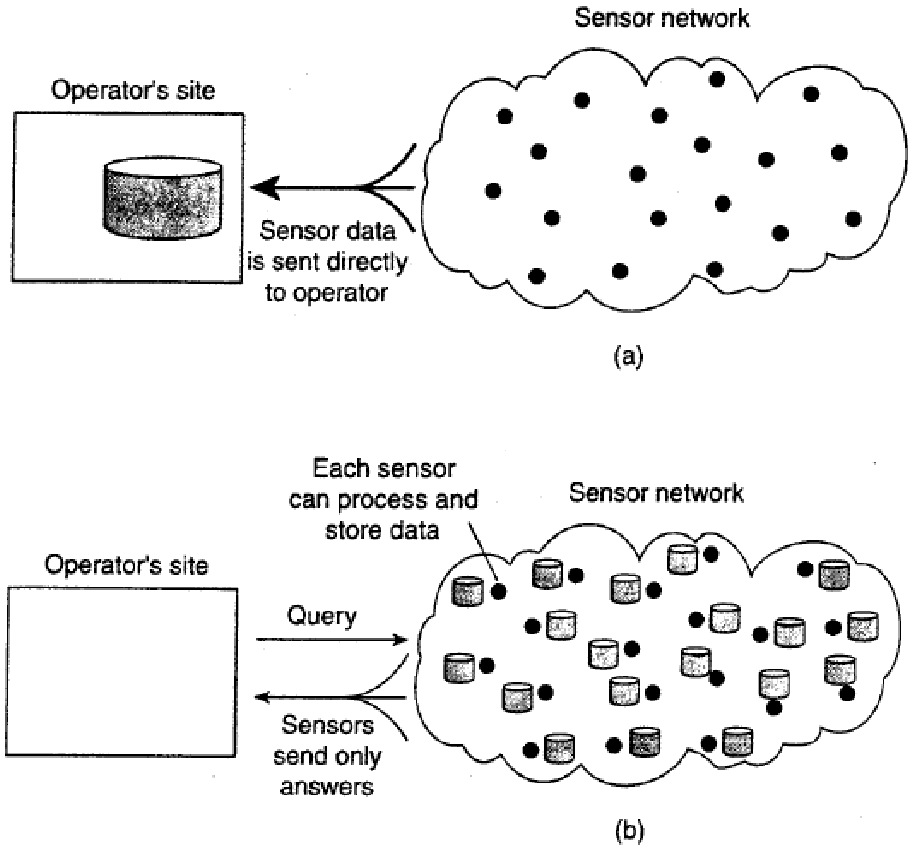
  A combination of public and private cloud networks

Cloud Usage Patterns and Economics



By Simone Brunozzi (AWS Technology Evangelist, APAC)

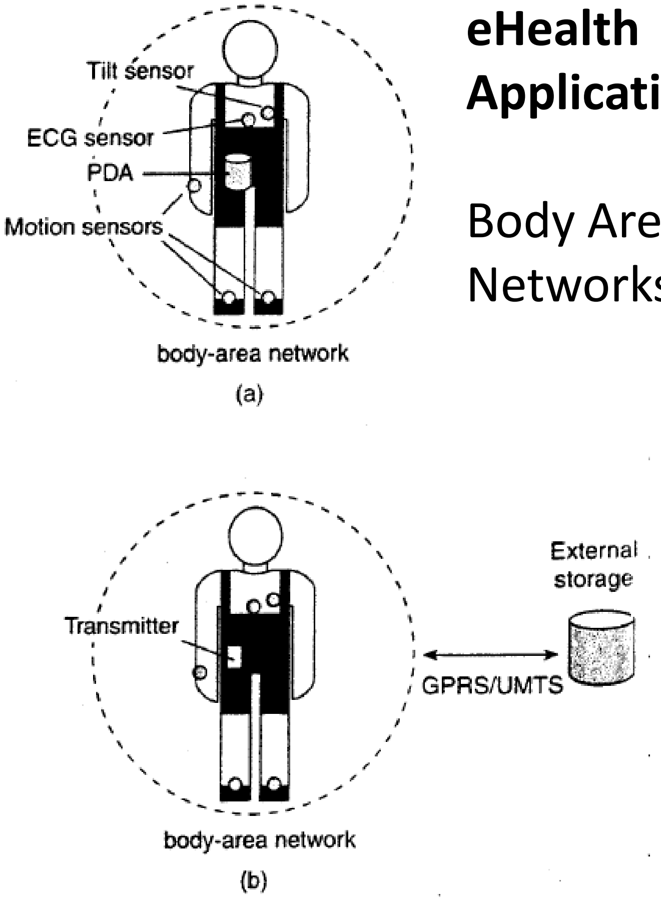
Pervasive Distributed Systems (more later..)


Sensor network
Operator's site
Sensor data is sent directly to operator
(a)
Each sensor can process and store data
Sensor network
Operator's site
Query
Sensors send only answers
(b)

Distributed Monitoring: Wireless Sensor Networks

eHealth Applications:

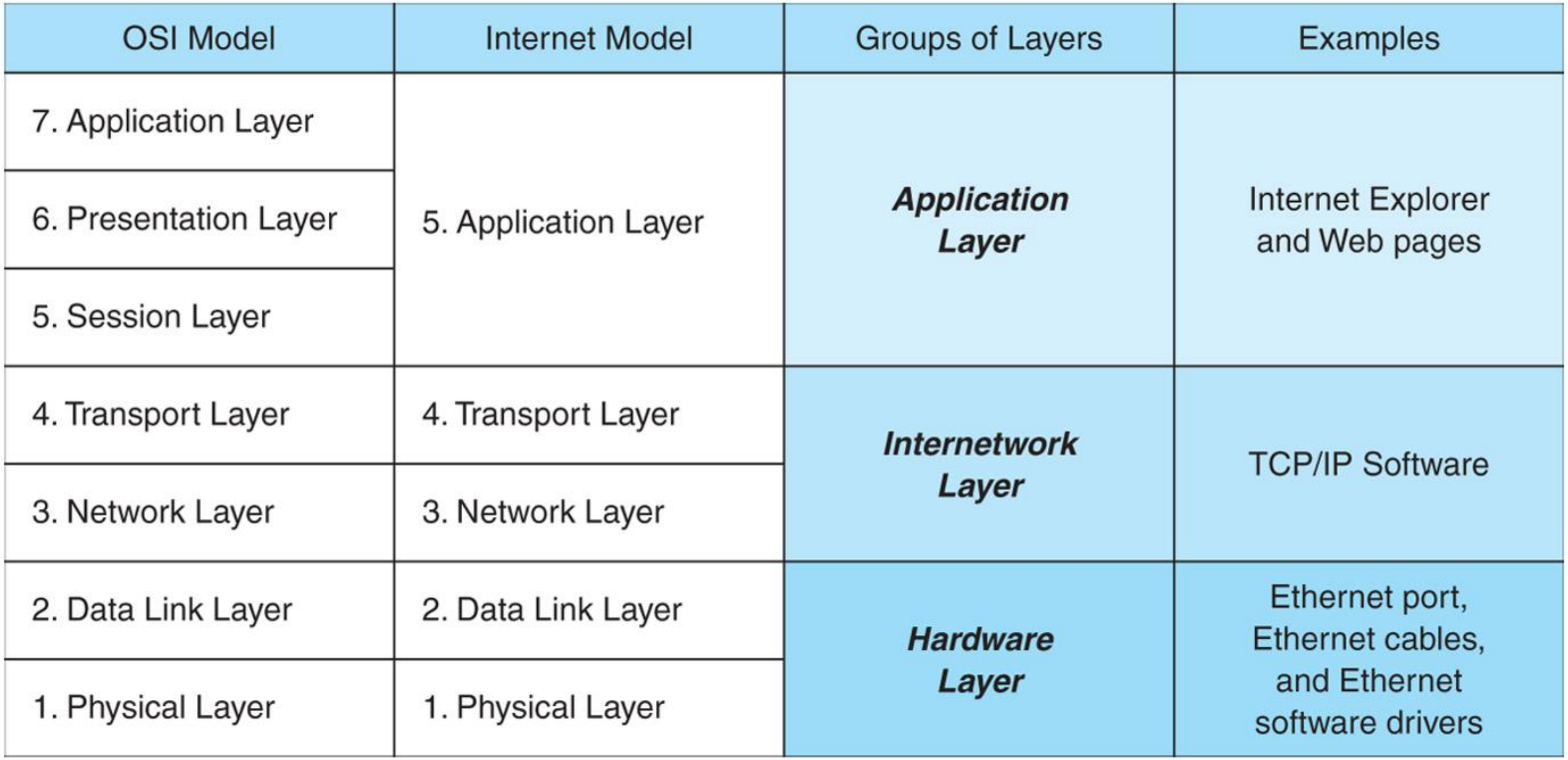
Body Area Networks



Tanenbaum & Van Steen (2014)

Communications Protocols and Standards in Distributed Systems – *Brief Introduction*

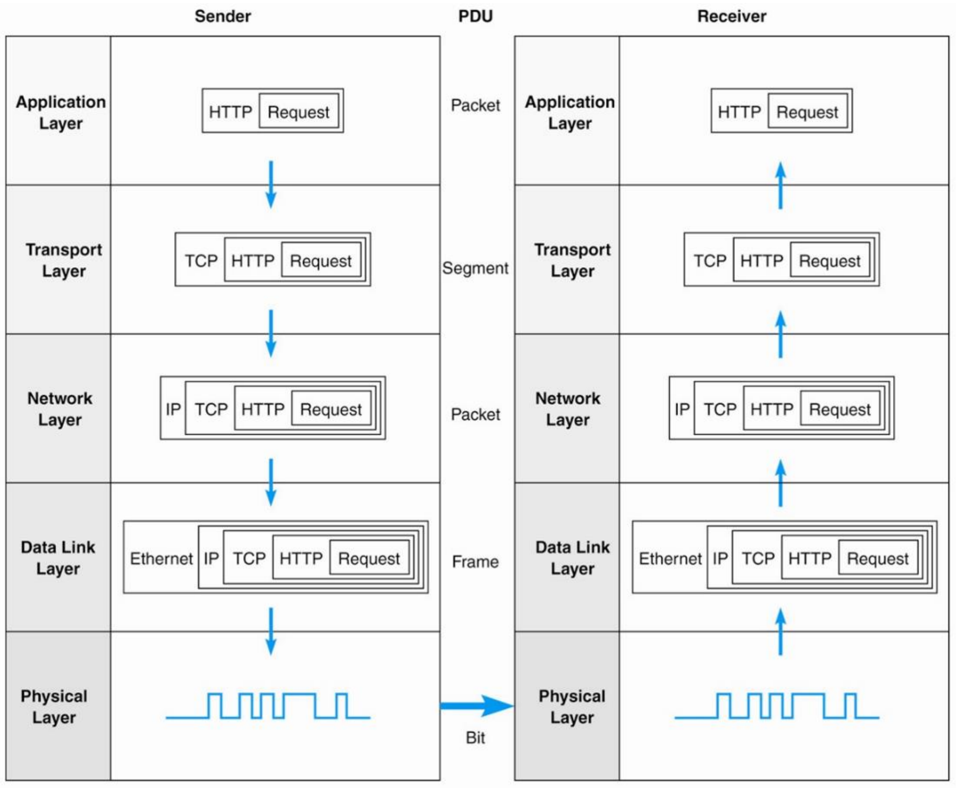
Summary of Communication Layer Models

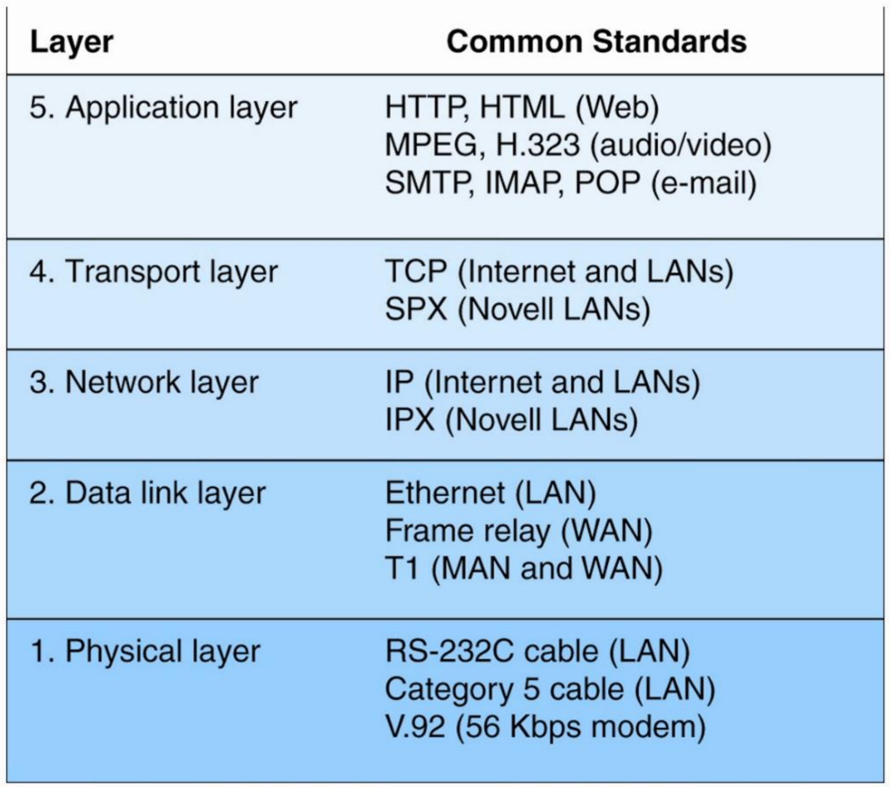


OSI Model	Internet Model	Groups of Layers	Examples
7. Application Layer	5. Application Layer	***Application Layer***	Internet Explorer and Web pages
6. Presentation Layer			
5. Session Layer			
4. Transport Layer	4. Transport Layer	***Internetwork Layer***	TCP/IP Software
3. Network Layer	3. Network Layer		
2. Data Link Layer	2. Data Link Layer	***Hardware Layer***	Ethernet port, Ethernet cables, and Ethernet software drivers
1. Physical Layer	1. Physical Layer		

FitzGerald & Dennis (2009)

Layered Data Communication Packets, Envelopes and Common Standards

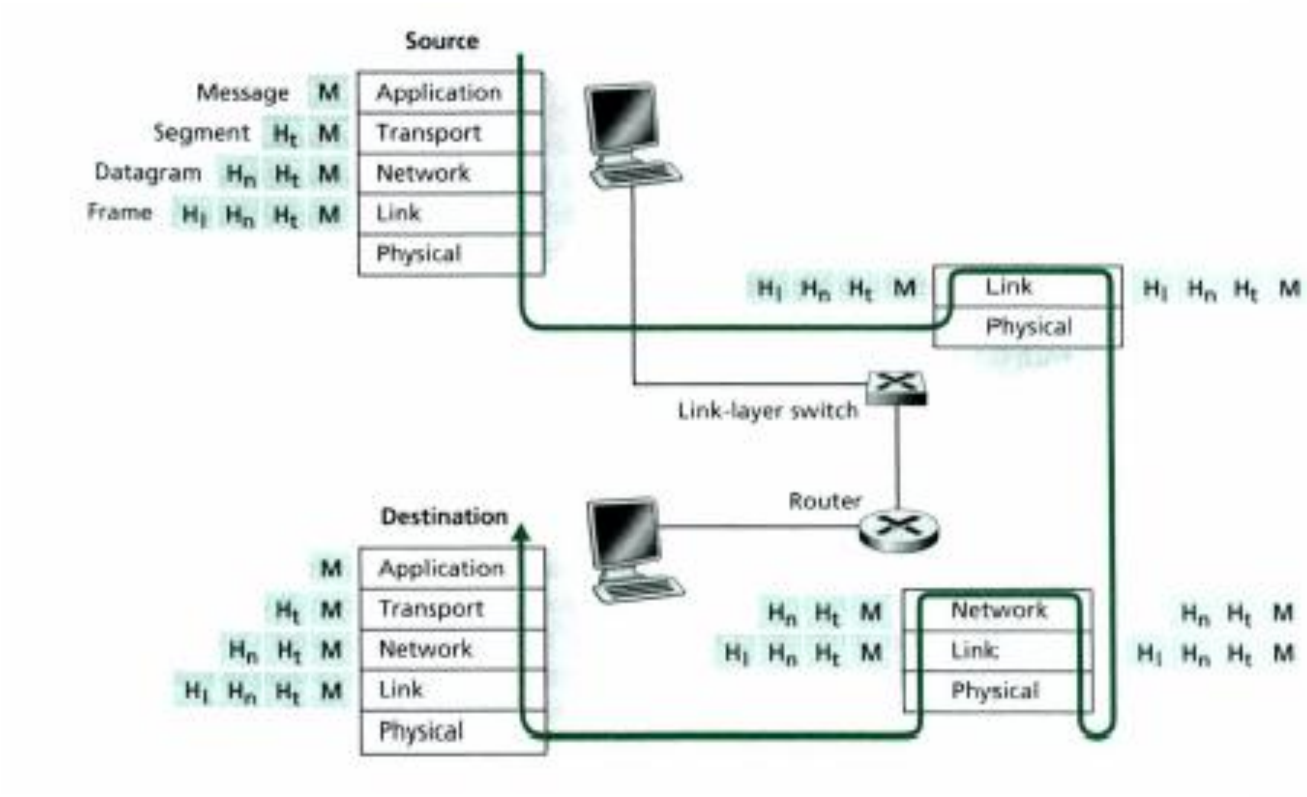


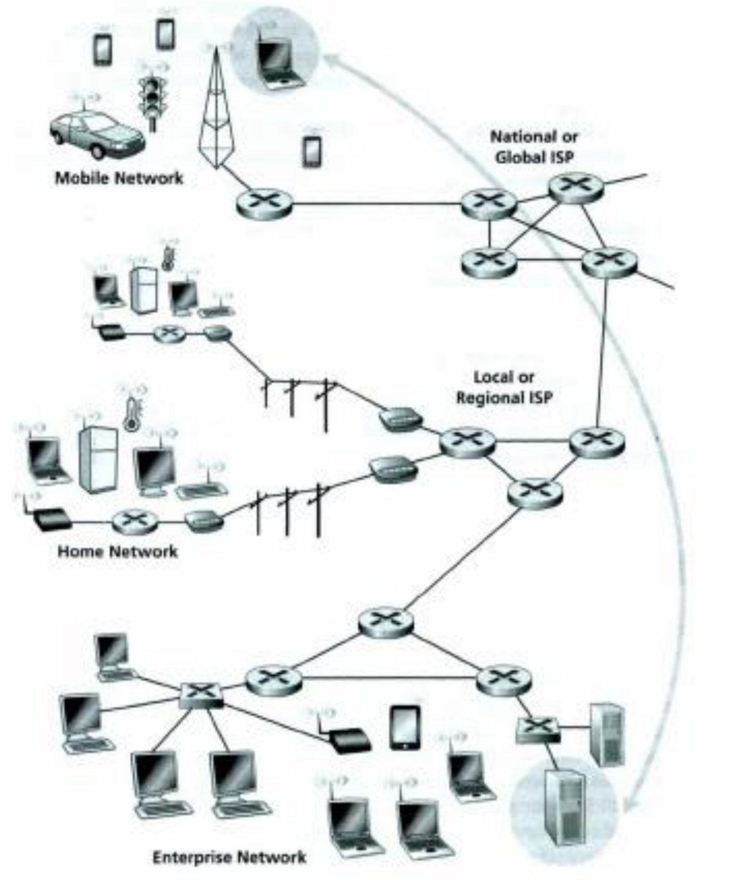


Layer	Common Standards
5. Application layer	HTTP, HTML (Web) MPEG, H.323 (audio/video) SMTP, IMAP, POP (e-mail)
4. Transport layer	TCP (Internet and LANs) SPX (Novell LANs)
3. Network layer	IP (Internet and LANs) IPX (Novell LANs)
2. Data link layer	Ethernet (LAN) Frame relay (WAN) T1 (MAN and WAN)
1. Physical layer	RS-232C cable (LAN) Category 5 cable (LAN) V.92 (56 Kbps modem)

FitzGerald & Dennis (2009)

End-2-End Internet Communications

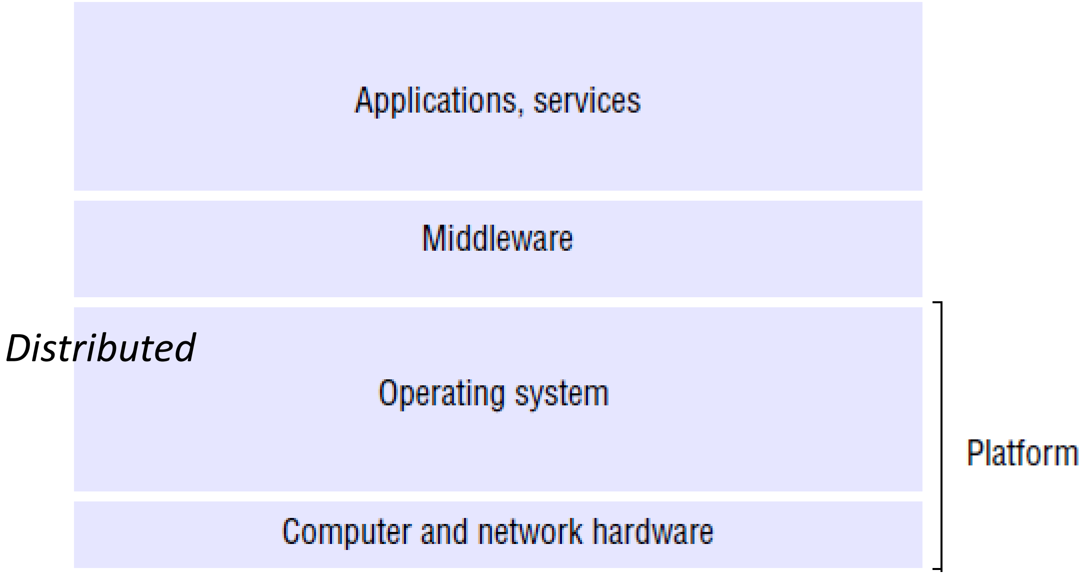

Source
Message M Application
Segment H_t M Transport
Datagram H_n H_t M Network
Frame H_l H_n H_t M Link
Physical
H_l H_n H_t M Link H_l H_n H_t M
Physical
Link-layer switch
Router
Destination
M Application
H_t M Transport
H_n H_t M Network
H_l H_n H_t M Link
Physical
H_n H_t M Network H_n H_t M
H_l H_n H_t M Link H_l H_n H_t M
Physical

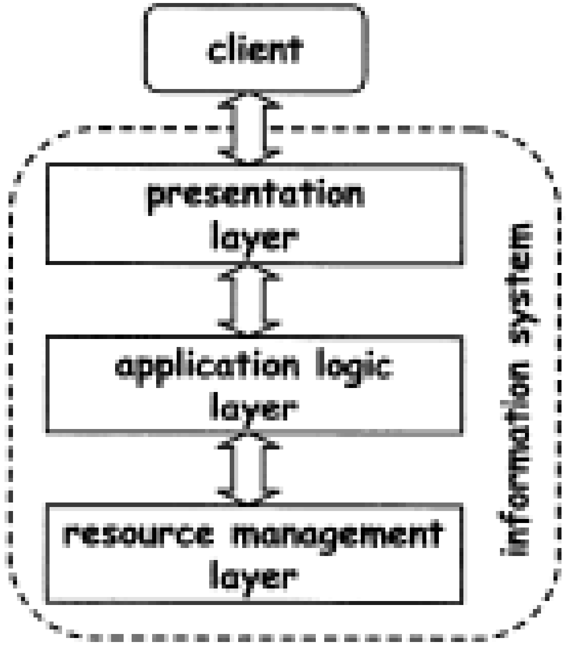


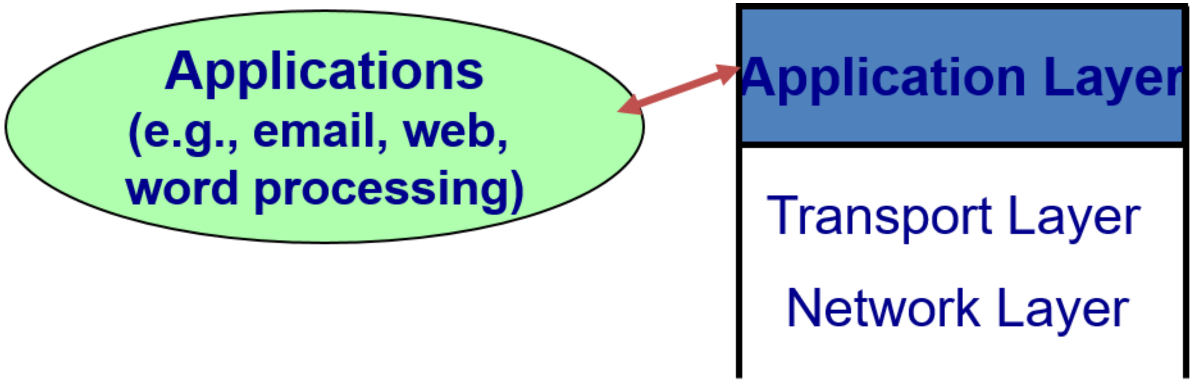
Couroulis et. al. (2012)

Network layers and logical layers of DCS

- Distributed Information Systems implemented in Application Layer
- Middleware sits above OS and Transport Layer







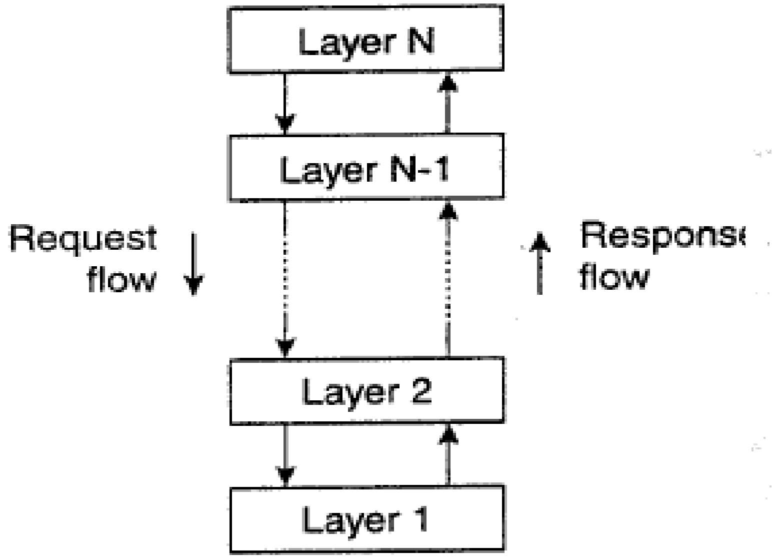
Architectural Models & Paradigms

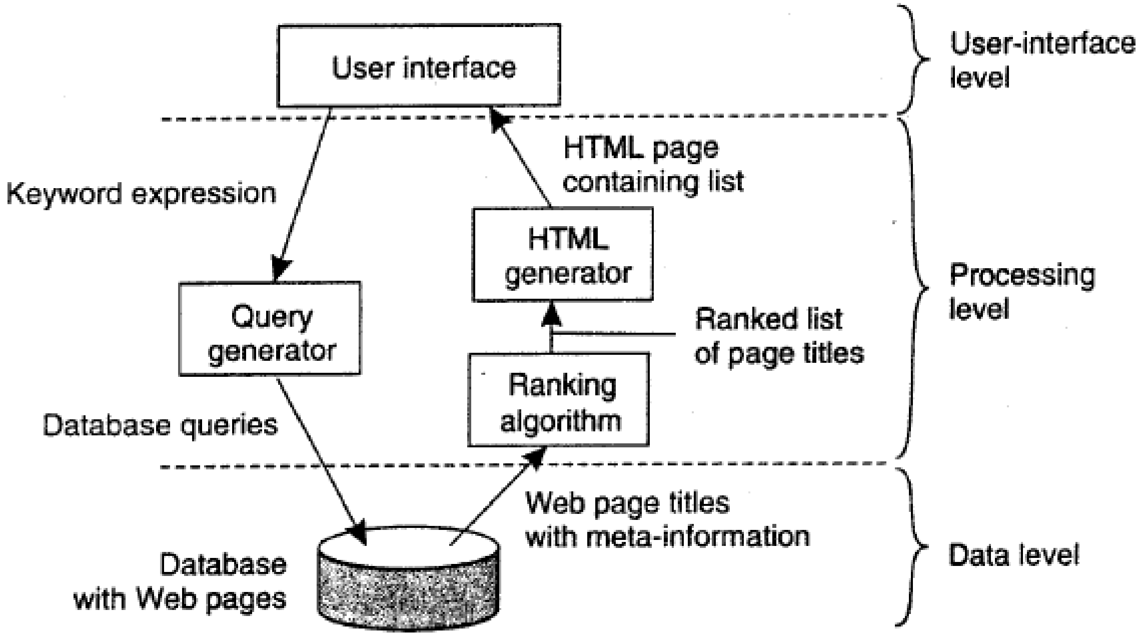
Software Architectural Styles

- The logical organisation of distributed systems into software components (how they are connected and exchange data)
- It includes:
  - **Components**: modular, replaceable parts with no dependencies
  - **Connectors**: Mechanisms for communication and cooperation, through interfaces, sending and receiving messages or calling methods on other objects
- Architectural styles of distributed systems:
  - Layered architecture
  - Object-based architecture
  - Data-centred architecture
  - Event-based architecture

Layered Architectures

- □ Grouping related functionalities into layers vertically
- □ Separation of concerns, flexibility, easy to maintain

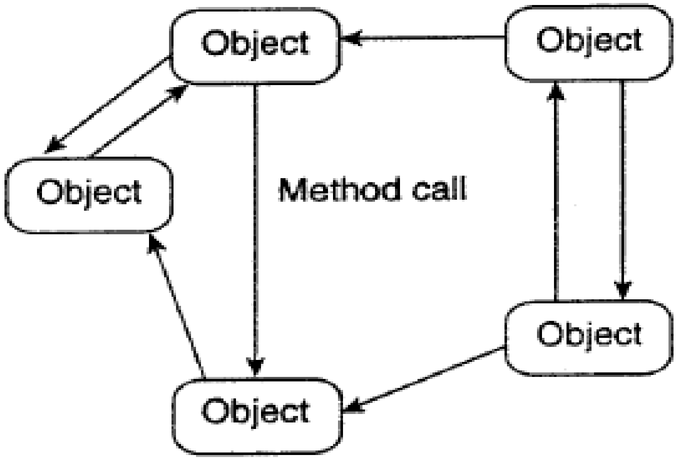


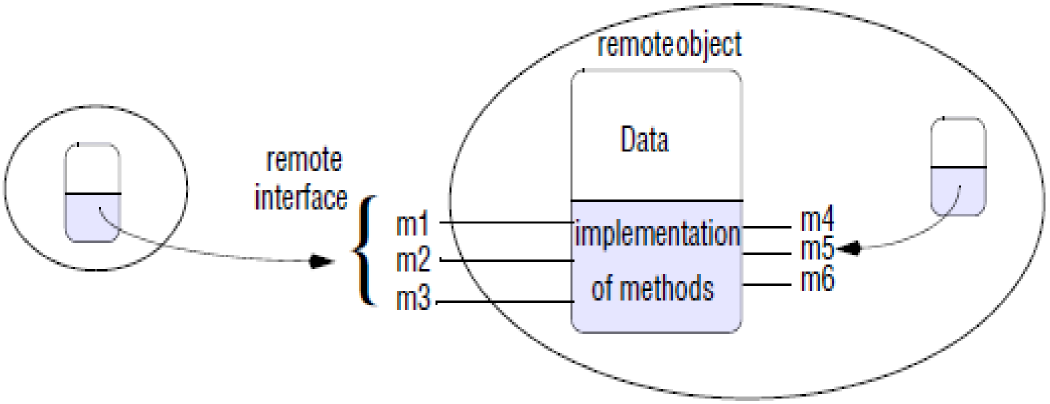


Tanenbaum & Van Steen (2014)

Object-based Architectures

- □ It consists of reusable, loosely coupled/independent objects responsible for different application functionalities
- □ Object-oriented concepts (abstraction, encapsulation, inheritance, etc.)



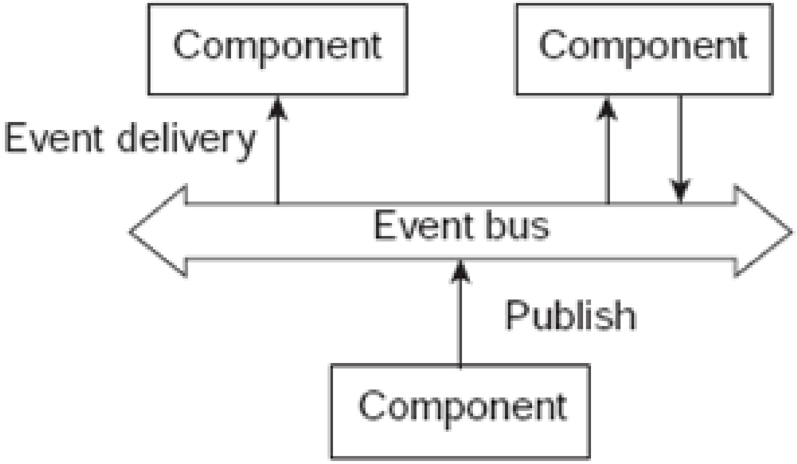


Tanenbaum & Van Steen (2014)

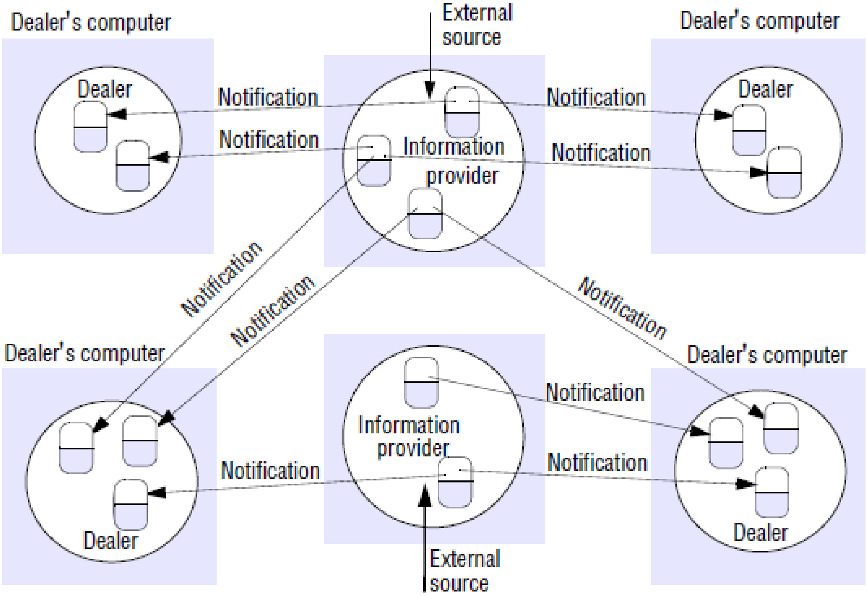
Couroulis et. al. (2012)

Event-based Architecture

- Components communicate based on events (significant changes in the state)
- It typically involves publish/subscribe messaging
- Popular distributed asynchronous architecture model (loosely coupled)



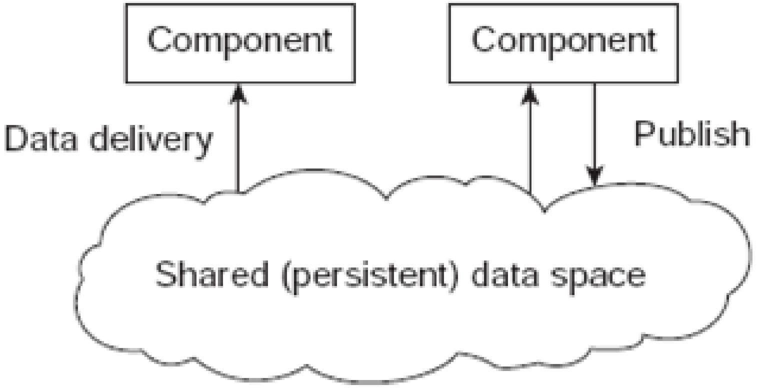
Tanenbaum & Van Steen (2014)



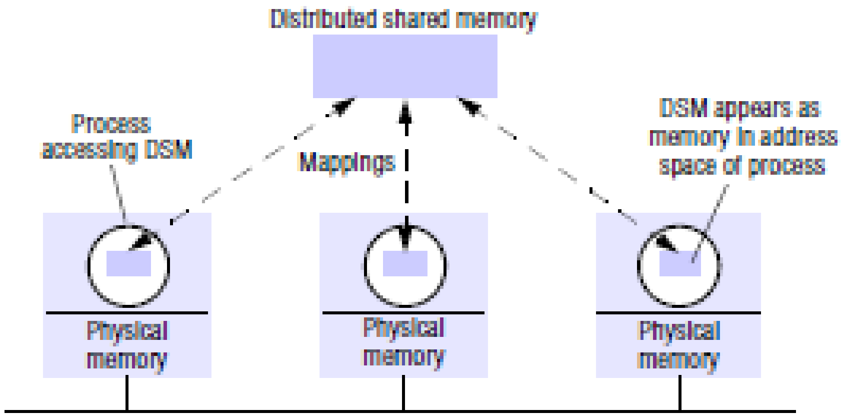
Couroulis et. al. (2012)

Data-Centred Architecture

- Components regularly access a common data repository and interact through the shared data store by reading or updating data
- Mainly consisting of a data store/repository and data accessors
- It can be integrated with event-based architectures



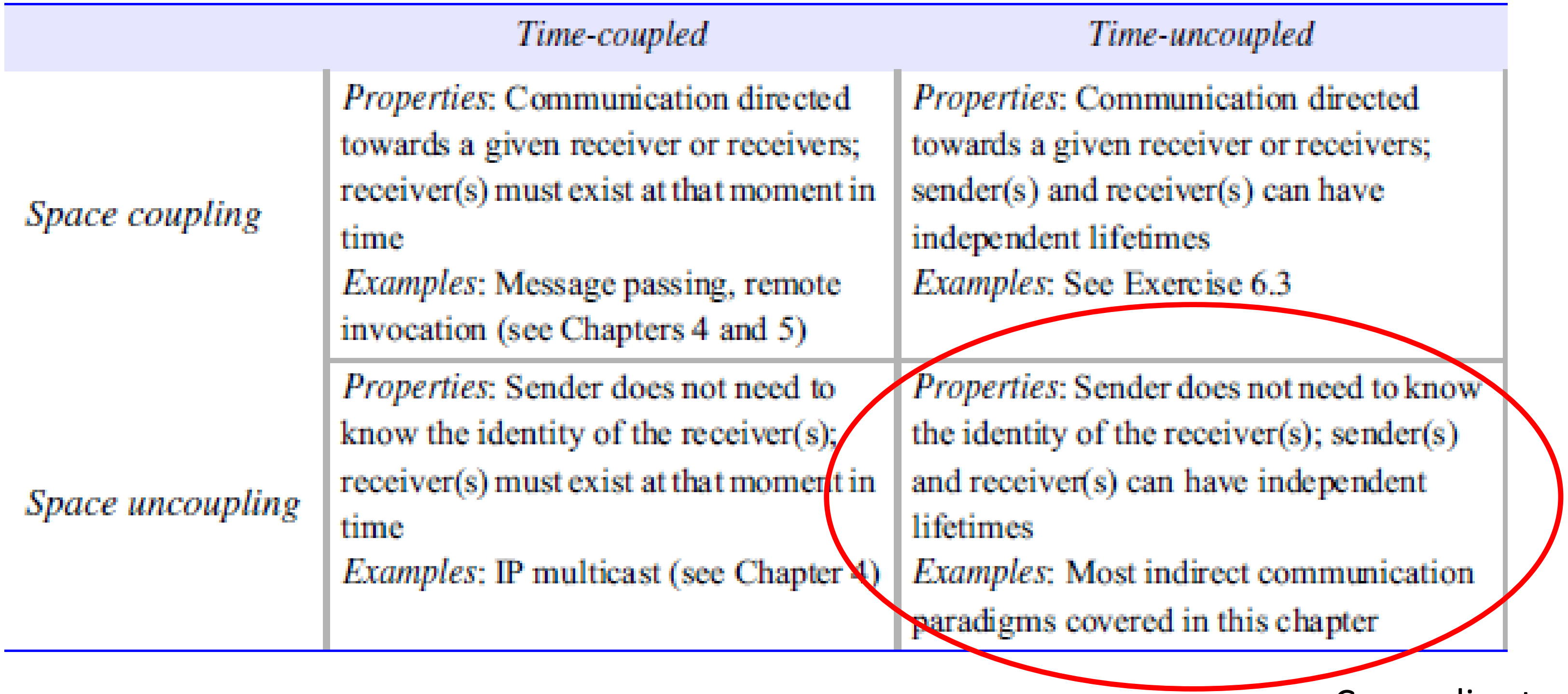
Tanenbaum & Van Steen (2014)


Distributed shared memory
Process accessing DSM
Mappings
DSM appears as memory in address space of process
Physical memory
Physical memory
Physical memory

Couroulis et. al. (2012)

Space and Time Uncoupling between Distributed Processes

- Event-based and Data-centred architectures using **Indirect Communication** assume that processes are **loosely-coupled** or **uncoupled** in space and time.



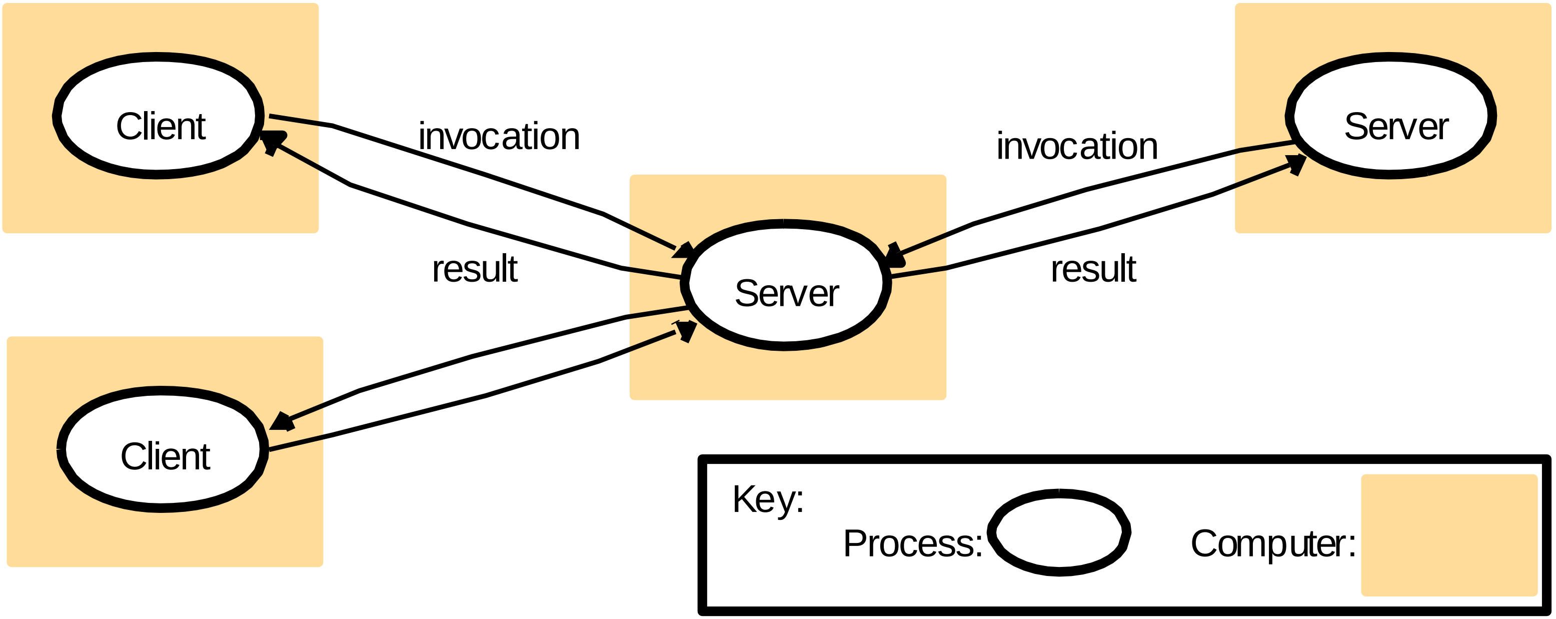
	Time-coupled	*Time-uncoupled*
Space coupling	*Properties*: Communication directed towards a given receiver or receivers; receiver(s) must exist at that moment in time *Examples*: Message passing, remote invocation (see Chapters 4 and 5)	*Properties*: Communication directed towards a given receiver or receivers; sender(s) and receiver(s) can have independent lifetimes *Examples*: See Exercise 6.3
Space uncoupling	*Properties*: Sender does not need to know the identity of the receiver(s); receiver(s) must exist at that moment in time *Examples*: IP multicast (see Chapter 4)	*Properties*: Sender does not need to know the identity of the receiver(s); sender(s) and receiver(s) can have independent lifetimes *Examples*: Most indirect communication paradigms covered in this chapter

Couroulis et. al. (2012)

Distributed Computing System Architectures

- **System architecture**: Based on placement of software components and their interactions.
- Data flows between the *processes* (not host machines/client PCs)
- *Vertical distribution*: placing logically different layers/components on different machines
- *Horizontal distribution*: a single logical layer is distributed across multiple machines (e.g. distributed databases)
- The **2-tiered** *client/server model*
- **3-tiered** and **multi-tiered** extensions to client/server model
- **Peer-to-peer**
  - Each process logically equal to each other (as client & server)
- Other system architectures (Master-slave, hierarchical tiers, …)

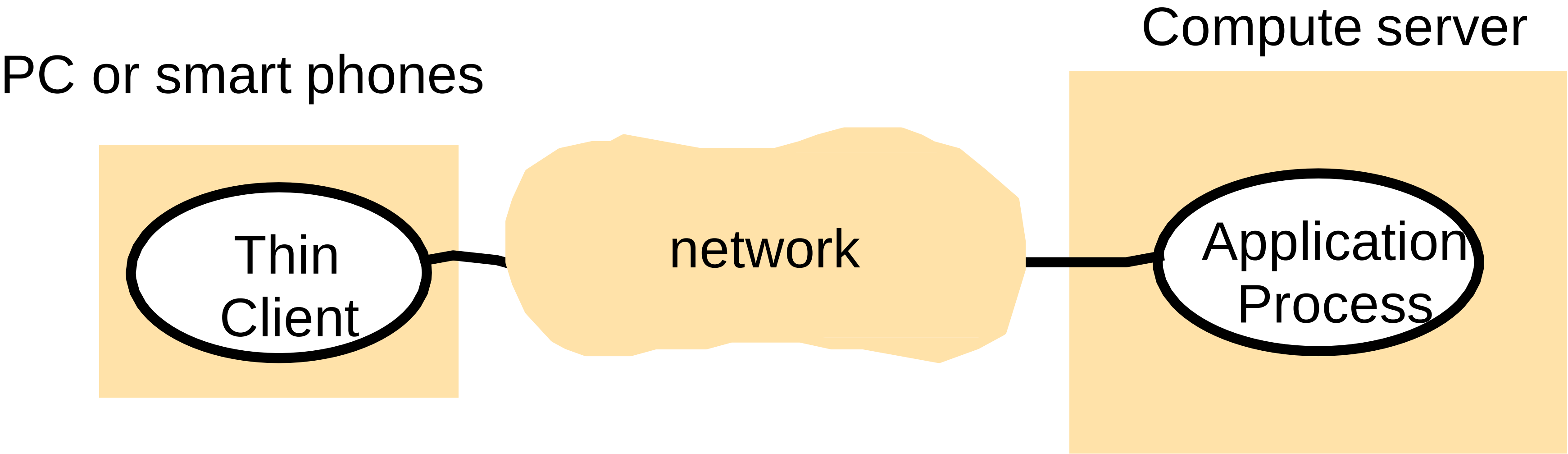
Client-Server



The most widely used architecture:

- Clients invoke individual servers
- Client processes interact with the server processes to access resources
- Servers can be clients of other servers – e.g. a web server can be a client of a file server

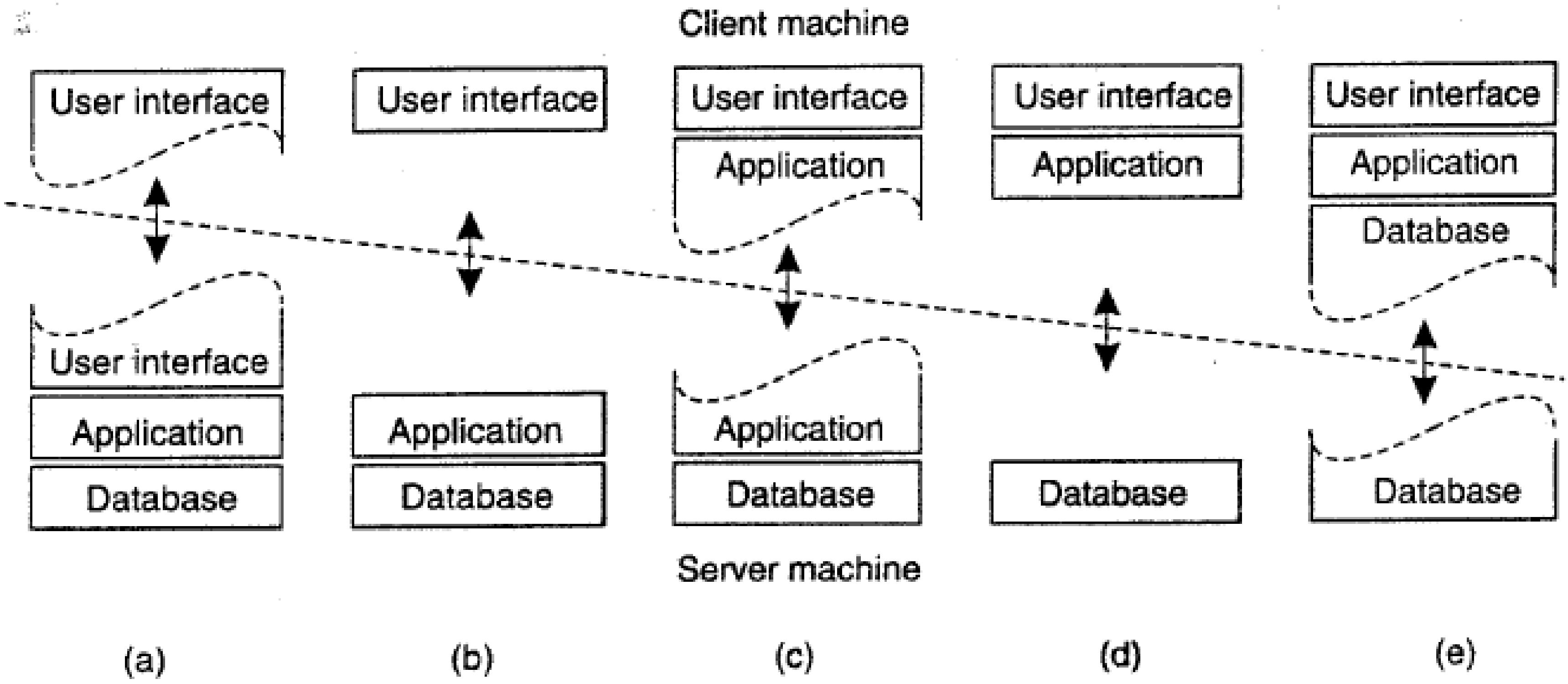
Thin clients and compute servers



- The advantage is that potentially simple local devices like a smart phone can be significantly enhanced with a set of networked services and capabilities
- The thin client model enables the access to sophisticated networked resources with few demands on the client device

Alternative Client-Server Organisations:

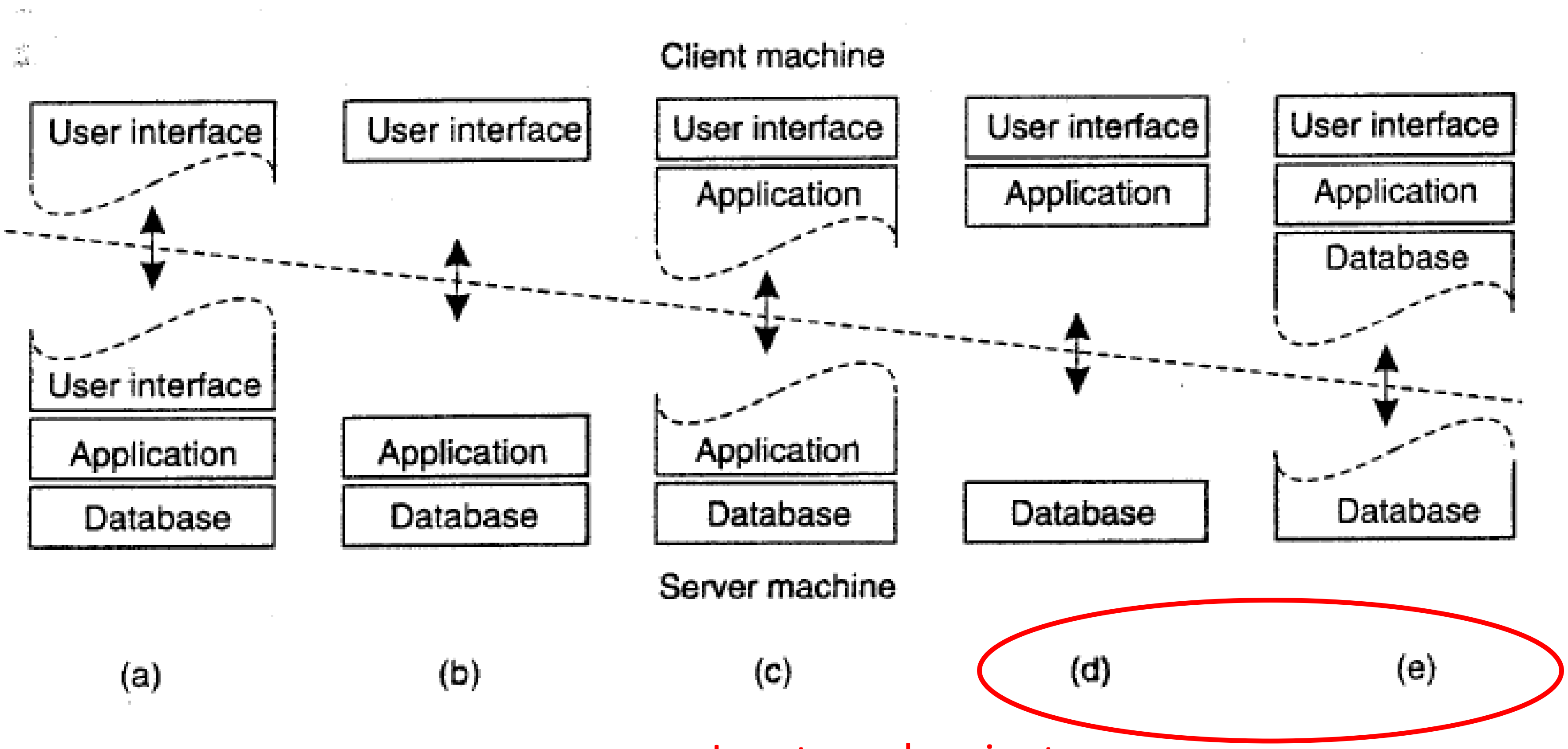
Thin and Thick Clients



Tanenbaum & Van Steen (2014)

Alternative Client-Server Organisations:

Thin and Thick Clients

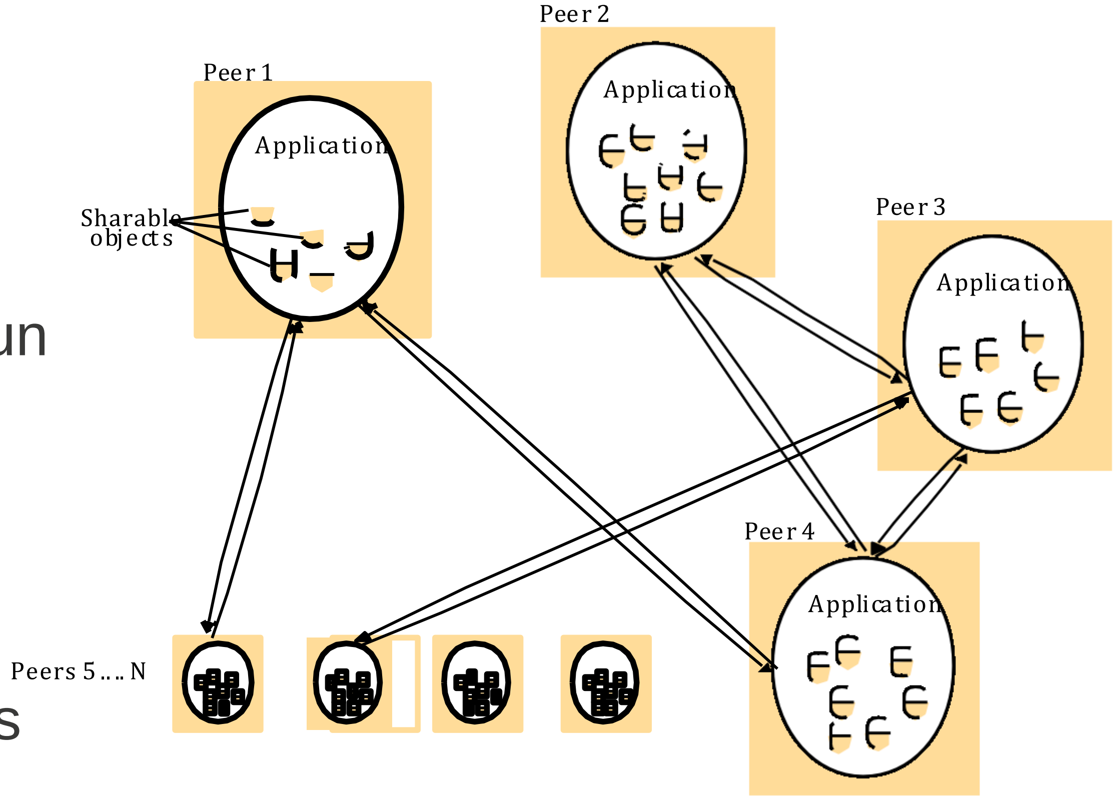


Least used variants

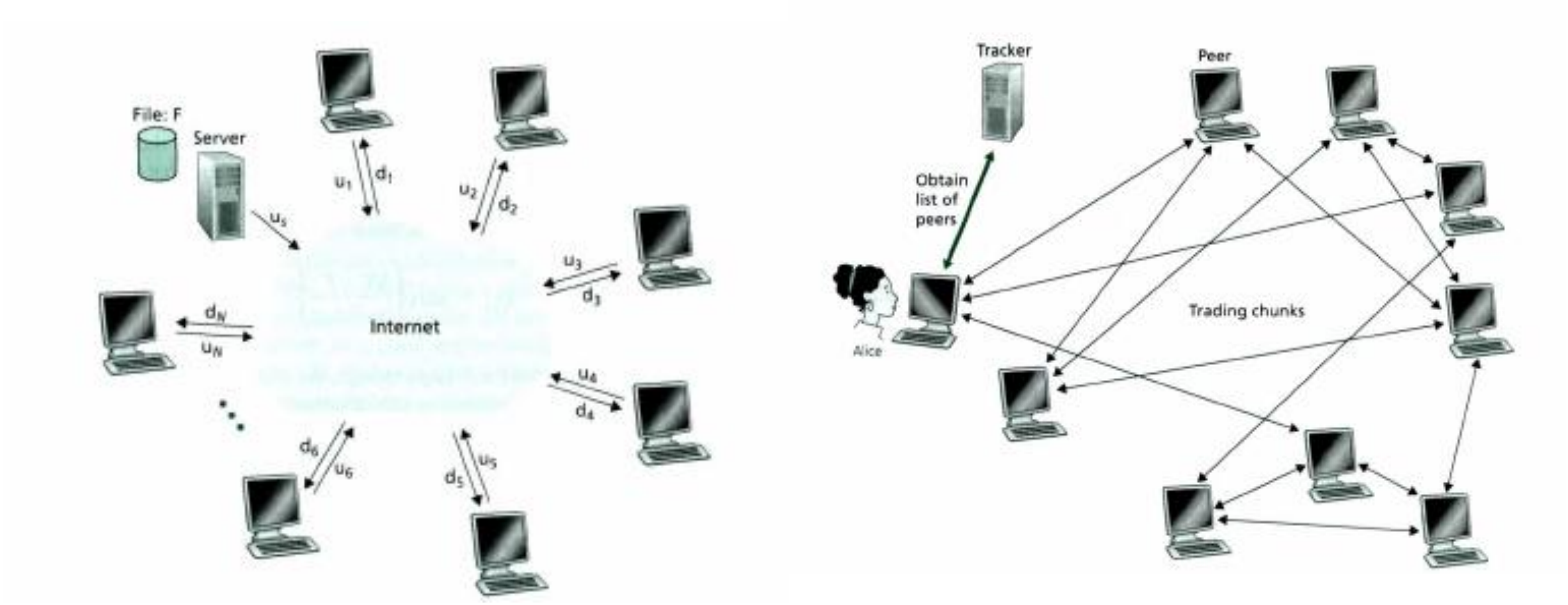
Tanenbaum & Van Steen (2014)

Peer to Peer (P2P) Model

- Applications consists of large numbers of peer processes running on separate machines
- Participating processes run same program and offer same set of interfaces to each other
- Data objects shared
- Cooperative interaction as peers
- Examples: Napster and BitTorrent (P2P file sharing)



Multi-client File Distribution and P2P
(BitTorrent example)



(a) Centralised CS architecture bottleneck

(b) Distributed file sharing in P2P network

Kurose & Ross (2017)

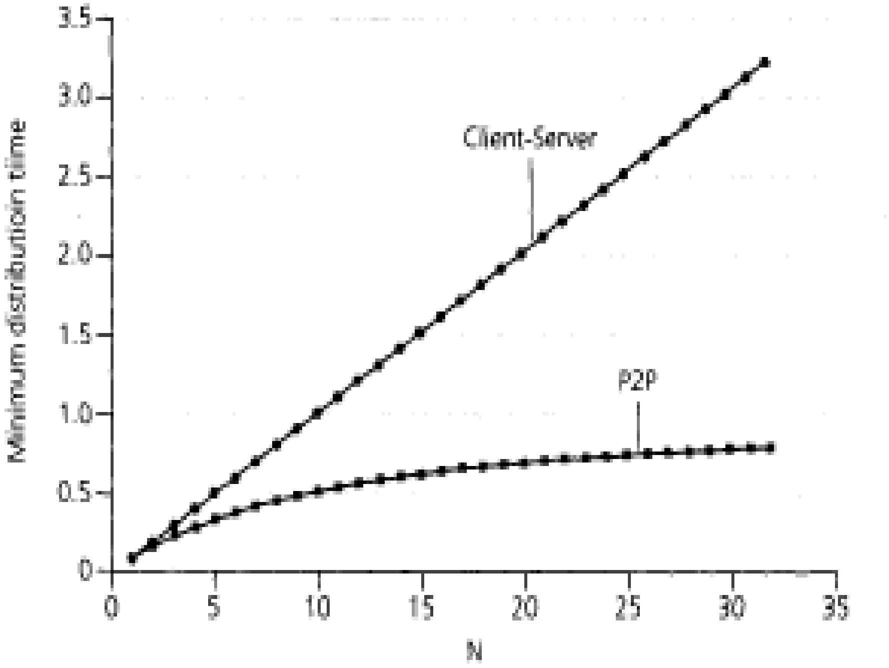
Scalability of P2P in Distributed File Systems

- Lower bound on download delay in centralised CS DFS scales by clients, *N*:

$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}.$$

- In P2P, facilitated download reduces lower bound on delay, allowing scalability with increasing *N*:

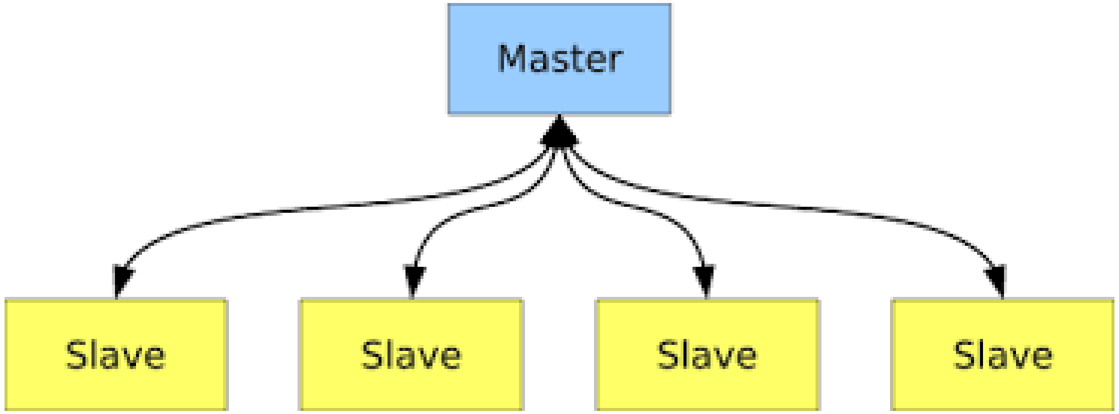
$$D_{P2P} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$



Kurose & Ross (2017)

Master and Slave Model

- One master process/program is in charge of all the other programs (slaves).
- Implemented as scheduler and worker threads
- Popular model for parallel programming and Big Data problems
- MapReduce and Hadoop



Mobile Computing and Distributed Mobile Computing – *Look ahead*

Motivation for Mobile Computing

- Mobility was the requirement of the 90’s, first in communications and then in computing
  - rapidly growing demand by users
  - many interested players

- Current technology makes mobile computing feasible

- Goal is to enable people to communicate/compute effortlessly, where they want, when they want, without “Wires”

What Mobile Users Want

- Seamless mobility
  - “connect” from any location, at any time
  - convenience of use (no extra setup, “plug and play”)
  - same computing environment, same services, consistent interfaces on the move
- Mobile users were willing to sacrifice some performance for mobility, but only *some*.

Distributed Mobile Computing

- Specialised class of distributed computing systems
- Some nodes can move in physical space
- Mobile computing is associated with mobility of hardware, users, data and software in computer applications.

Challenges & Design Issues

Pitfalls when Developing Distributed Systems

False assumptions made by first time developer:

- The network is reliable.
- The network is secure.
- The network is homogeneous.
- The topology does not change.
- Latency is zero and Bandwidth is infinite.
- Transport/transmission cost is zero.
- There is one administrator.

Pitfalls when Developing Distributed Systems

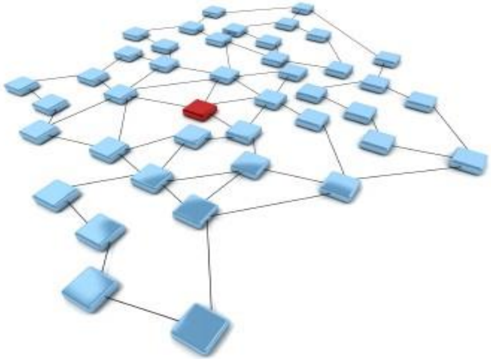
False assumptions made by first time developer:

- The network is reliable. *Reliable networks are non-existent*
- The network is secure. *Network communications are insecure*
- The network is homogeneous. *Heterogeneity (异类) is normal*
- The topology does not change. *Mobile ad-hoc networks*
- Latency is zero and Bandwidth is infinite. *Scalability limits*
- Transport/transmission cost is zero. *It is relative to use*
- There is one administrator. *May traverse several zones*

Design Issues

Heterogeneity

Scalability



Reliability

Security

Transparency

Performance

Mobile Computing Challenges

- Mobile computing introducing new challenges
- Mobility and dynamic environments

  - Communications
    - Cost of Disconnections
    - Network protocols and technologies (old and new)

  - Mobile devices
    - Resource constraints (i.e. battery) Energy management