ICT2107
DISTRIBUTED SYSTEMS PROGRAMMING

Introduction

Dr. Frank Guan
BEng (Hons), PhD

Email: Frank.Guan@SingaporeTech.edu.sg
Tel: +65 6592 1926

Dr. Zhengkui Wang
BS, MS, PhD

Email: Zhengkui.Wang@SingaporeTech.edu.sg
Tel: +65 6592 2077

Major Topics (1st Half)

Introduction to Distributed Systems

Distributed System Architecture and Middleware

Distributed Communications & Synchronization

Consultation upon appointment: Frank.Guan@SingaporeTech.edu.sg

Assessment (1st half)

Item	%
Lab Assignments	5%
Group Project 1	20%
Quiz 1	10%
Sub-total	**35%**
Exam (1st and 2nd Half)	30%

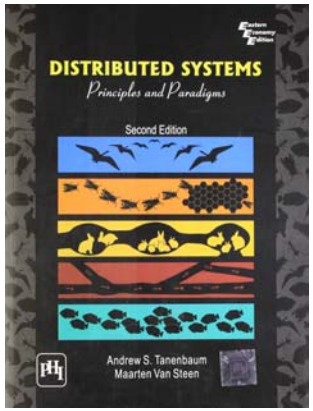
Timeline

Week #	Lecture	Lab
1	Introduction	
2	Architecture	TCP & UDP
3	Communication	RMI
4	Communication & Synchronization	Multicast
5	Synchronization & Review	Lab projects demonstration
6	Quiz 1 and Web Service	Group project presentation

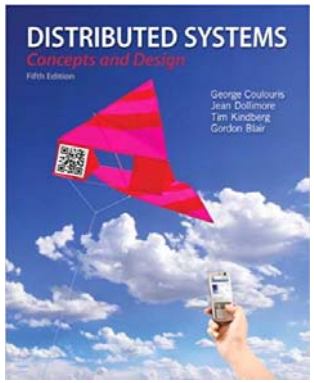
Learning Outcomes

1. Demonstrate understanding of the principles and techniques behind the design and implementation of distributed systems.
2. Explain the basic algorithms and their respective assumptions.
3. Identify software and architecture needed to enable the development of distributed applications.
4. Write programs that can interoperate using well-defined protocols.
5. Debug a code that spans multiple programs running on several machines.
6. Design, implement and debug a real distributed system.

Reference Books



Distributed Systems: **Principles and Paradigms**
Andrew S.Tanenbaum, Maarten Van Steen



Distributed Systems: Concepts and Design
George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair

ICT2107 DISTRIBUTED SYSTEMS PROGRAMMING

LECTURE 01
INTRODUCTION

Examples of DS

What else?

What we will cover in this lecture…

Enablers of Distributed Systems

Definition of Distributed Systems

Characteristics of Distributed Systems

Goals of Distributed Systems

Types of Distributed Systems

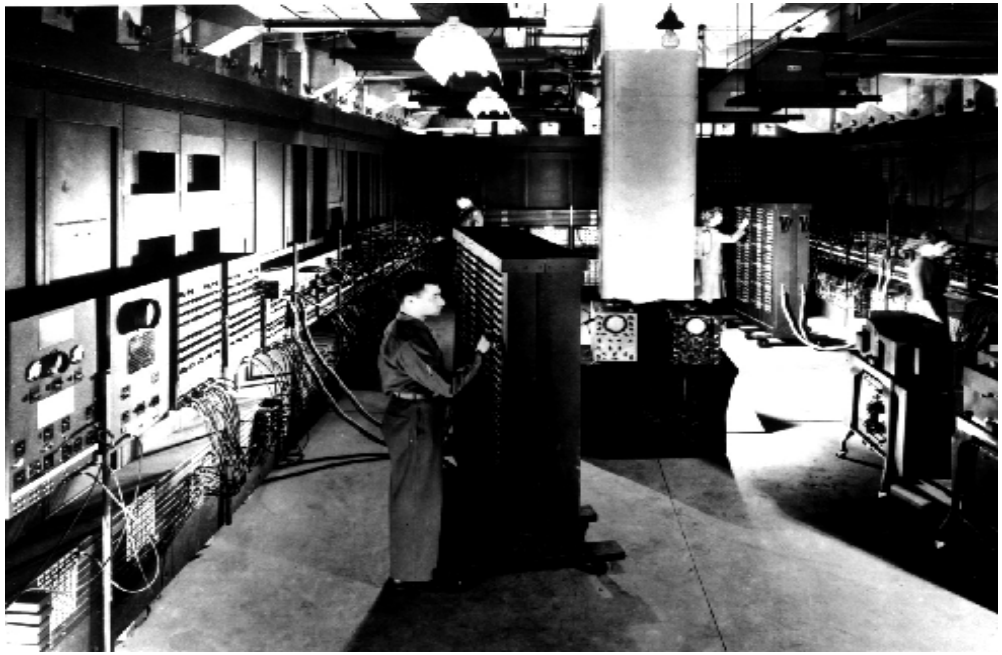
Introduction

ENABLERS

OF DISTRIBUTED SYSTEMS

1st Fully Functional Digital Computer

The ENIAC: Electronic Numerical Integrator And Computer (1943 – 1946)



John Presper Eckert (1919-1995)
and
John Mauchly (1907-1980)

University of Pennsylvania
Moore School of Engineering

50 tons, 18K vacuum tubes, with the computing power of little more than the modern calculator…..

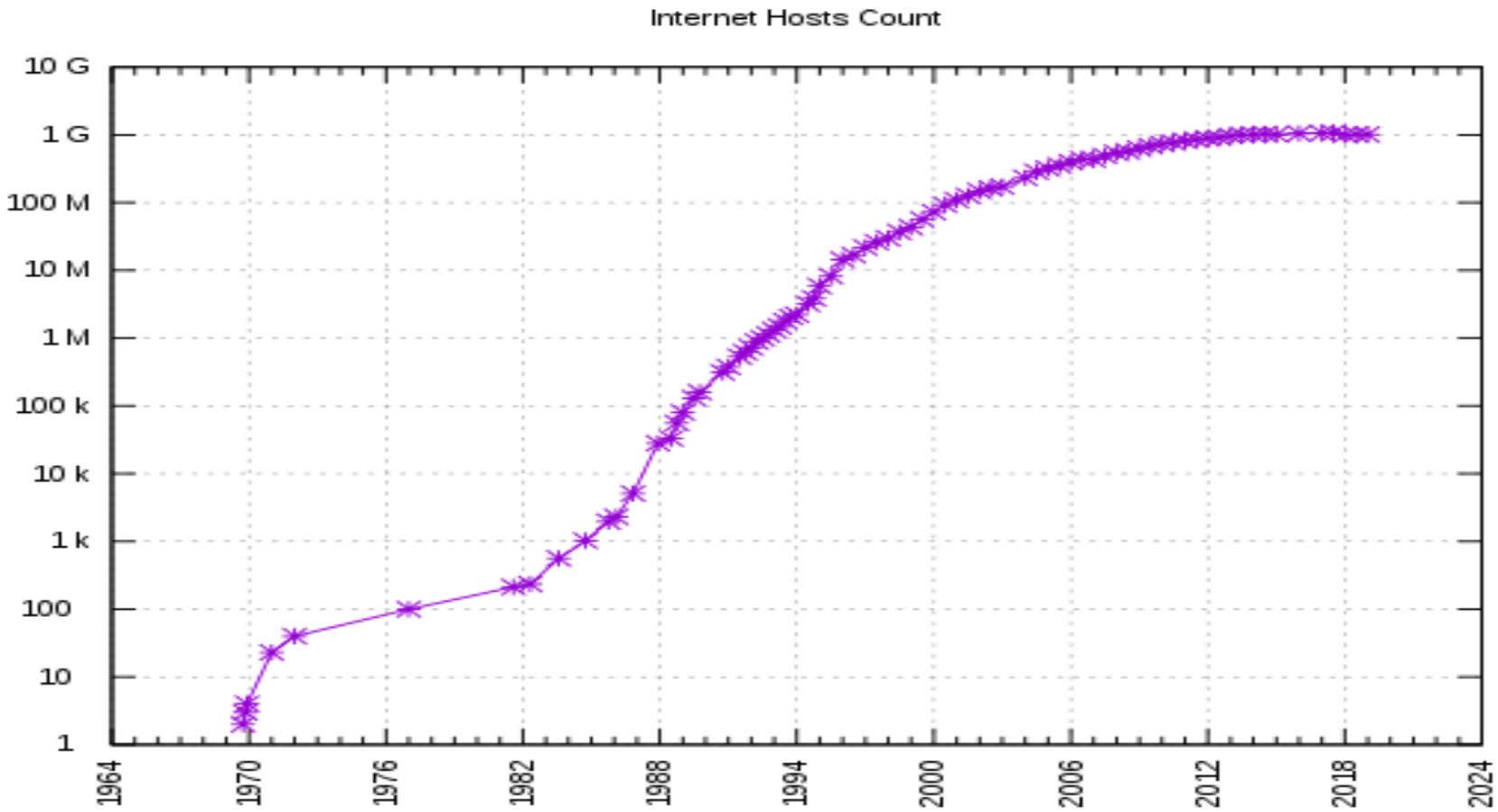
High Cost of Earlier Computing Devices

Typical 1968 prices—Excluding maintenance & support!

Model	Description	Purchase Price	Installation Fee
3011-95	1108 CPU	$566,460	$2,200
7005-72	131 K word Core Memory	$823,500	$2,250
5009-00	FASTRAND^{tm} Controller	$41,680	$600
6010-00	FASTRAND II Storage Unit	$134,400	$1,080
5012-00	FH-432/FH-1782 Drum Controller	$67,360	$600
6016-00	FH-432 Drum (capacity 262,144 words)	$34,640	$480
6015-00	FH-1782 Drum (capacity 2,097,152 words)	$95,680	$540
4009-99	Console (TTY-35)	$29,365	$200

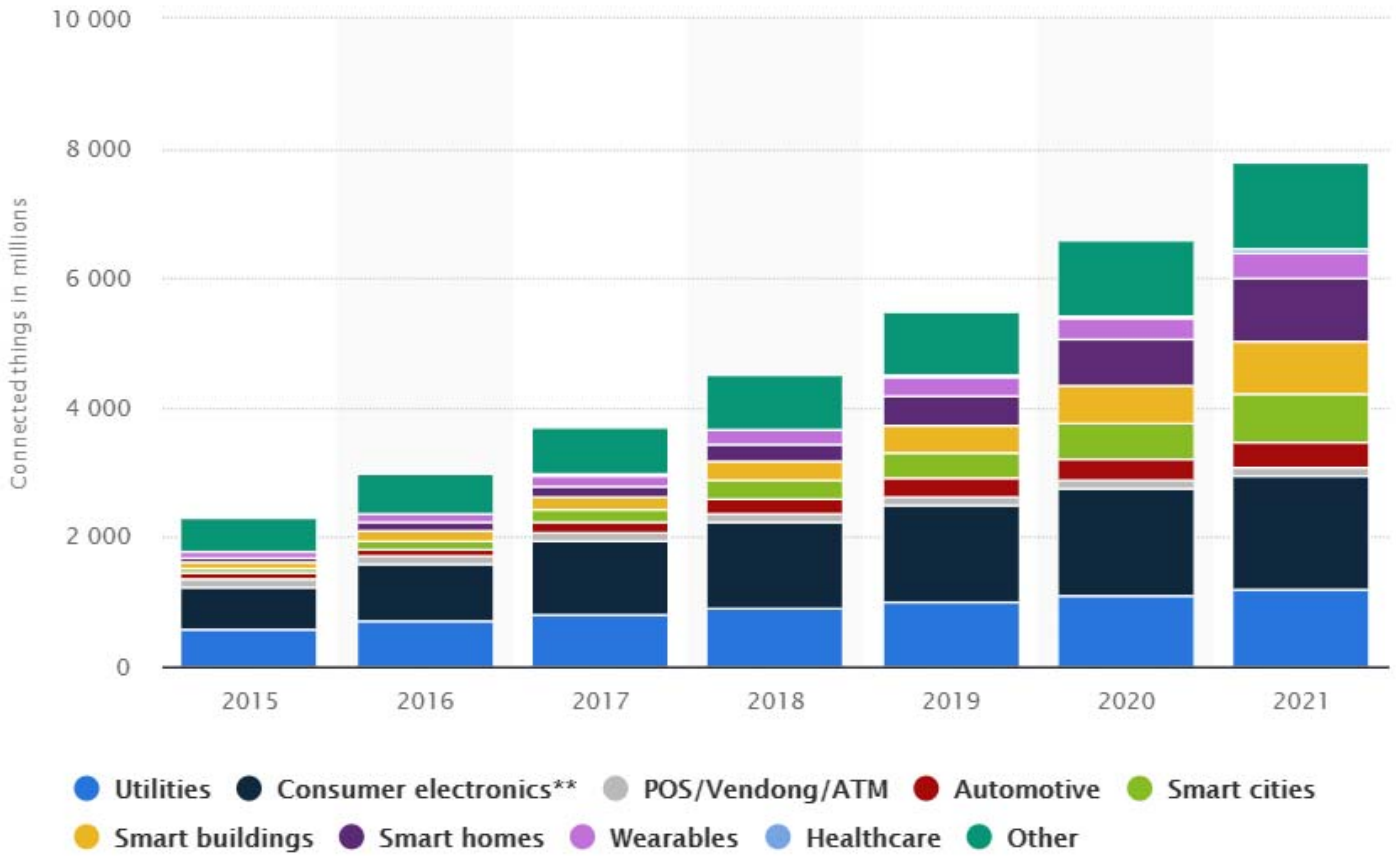
In total: US$1.8 Million in 1968

Growth of Internet



https://en.wikipedia.org/wiki/History_of_the_Internet

Connected Devices in the World



https://mytechdecisions.com/it-infrastructure/internet-things-will-affect-smbs

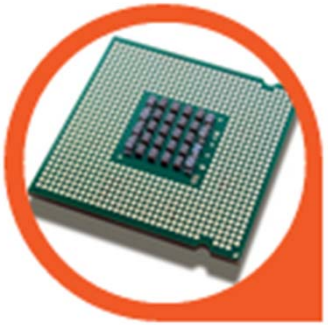
ENABLERS OF DISTRIBUTED SYSTEMS

Computers were large and expensive

1945

mid 1980s

Two advances in technologies begin to change the situation



#1 - Development of microprocessor
- Reasonable cost
- Reasonable level of performance

As the consequence
- There are many computers
- There is an increasing needs for connecting them together

#2 - Development of network technologies such as
- Local Area Network (LAN)
- Wide Area Network (WAN)

Allowing us to easily
- Setup and
- Connect many computers together

Introduction

DEFINITIONS

OF DISTRIBUTED SYSTEMS

DEFINITION OF DISTRIBUTED SYSTEMS

"A distributed system consists of a collection of autonomous computers, connected through a network and distribution middleware, which enable the computers to coordinate their activities and to share the resources of the system so that users perceive the system as single, integrated computing facility"

Prof Wolfgang Emmerich
University College London
1997

CONNECTED AUTONOMOUS COMPONENTS

"**Autonomous**" means individual component may be able to operate independently

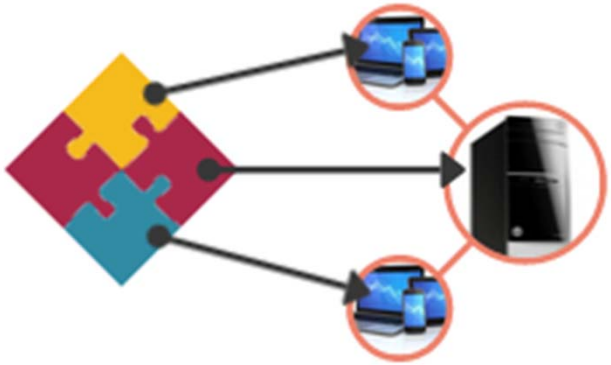
In DS, components are connected via a network using
- Network technologies
- Distribution middleware

Network technologies
- Hardware: hub, switches, cables, etc.,
- Software: Network OS, Protocols, Algorithms, etc.,

Distribution middleware
- E.g., Web sever, application server, messaging, etc.
- Based on XML, SOAP, Web services, etc.,

WHY CONNECTED?

Autonomous components are connected in order to share resources and coordinate actions



Coordinate actions means sharing the processing load of a task over the network

- Task is divided into smaller parts
- Each is processed in a component

Resources may be in form of hardware or software resources

Examples
Hardware resources
Disk, Printers, etc.

Examples
Software resources
Database, file, etc.

USERS PERCEIVE DS AS A SINGLE SYSTEM

DS should be designed so that users can perceive it as a single system

This will reduce the complexity of the system

- Users don’t have to care about distributed aspects of the system
- Users can use the system as if it was a local system

This aspect depends heavily on the **Transparency** of the design of DS

Introduction

IMPORTANT CHARACTERISTICS OF DISTRIBUTED SYSTEMS

Important Characteristics

- Heterogeneity
- Transparency
- Scalability
- Partial Failure

HETEROGENEITY

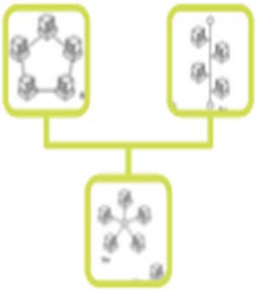
Components of DS may be different from one another

Different hardware

e.g., Computers, Mobile Devices, etc.

Different network technologies

e.g., LAN, WAN, Wired, Wireless, etc.



Different Operating Systems

e.g., Mac OS X, Windows OS, Linux OS, etc.

TRANSPARENCY

Differences among various components, the ways they communicate, their internal organization and processing are hidden from users.

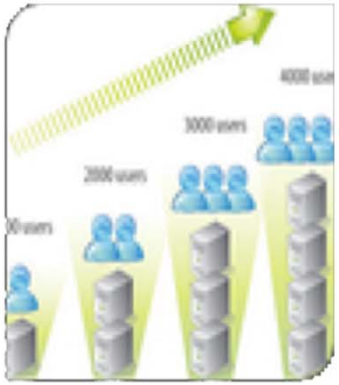
Users and applications can interact with distributed system in a consistent and uniform way regardless of where and when the interaction takes place.

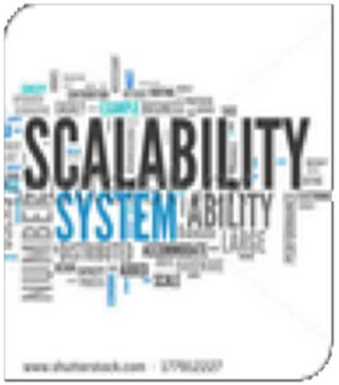
SCALABILITY

Distributed systems should be relatively easy to expand or scale.

Scalability is the ability of the system to

- Add/handle more resources: disks, data, etc
- Serve more users





PARTIAL FAILURE

DS must be able to tolerate partial failure of system

Example

A Website is running on two servers using distributed technologies, if one failed

- Another one still keeps the website working
- Probably with lower performance level

https://www.google.com/about/datacenters/inside/locations/index.html

10 Minutes Break

Introduction

DESIGN GOALS

OF DISTRIBUTED SYSTEMS

MAIN DESIGN GOALS

There are 4 main goals for DS Designs

- **Accessibility**: Making resources easily accessible
- **Transparency**: Hiding the fact that resources are distributed
- **Openness**: Can integrate different hardware/software platforms from different administrative domains
- **Scalability**: Can add more hardware/software and serve more users

GOAL 1: MAKING RESOURCES ACCESSIBLE

Resources might be in form of

- Hardware (printers, computers, storage, etc.)
- Software (data, files, web pages, etc.)

Connecting users and resources also makes it **easier to collaborate task and exchange information**

This sharing of resources also increase the **demand for security**

GOAL 2: TRANSPARENCY

This means to hide the fact that its processes and resources are physically distributed over the network

These are main types of transparency

- Access Transparency
- Location Transparency
- Migration Transparency
- Relocation Transparency
- Replication Transparency
- Concurrency Transparency
- Failure Transparency
- Implementation Transparency

ACCESS TRANSPARENCY

To hide differences in data representation and how a resource is accessed

We could access and retrieve data in the same way regardless of whether the system is distributed or not

Examples

- Different OS will have different ways of representing and manipulating a file
- However, these should be hidden
- So that, users can manipulate and work with the files in the same way

LOCATION TRANSPARENCY

- To hide where a resource is physically located
- Users don’t have to remember physical locations of resources
- Users could just use logical names to access resources
- Example of a logical name is a URL

MIGRATION TRANSPARENCY

To hide the fact that a resource may move to another location

Examples

- A resource relocation
- The users could still use the same name to access to the resource
- The system will know how to map this name to the new physical location accordingly
- This could be done by a standard naming directory (e.g., changing a record in a DNS)

REPLICATION TRANSPARENCY

Hide the fact that a resource is replicated

This is important since services can be replicated

- To increase availability by having many copies or
- To increase performance by placing a copy close to the place where it is accessed

E.g., Google has its servers placed in many countries

CONCURRENCY TRANSPARENCY

Hide the fact that a resource may be shared by several competitive users

Example

- Several users can access the same table in a shared DB
- Several users can access the same webpage at the same time
- The users don't have to know about these

Any issues related to concurrency (e.g., consistency) needed to be handled by the DS itself

FAILURE TRANSPARENCY

To hide the failure and recovery of a resource

DS failure is normally partial because it has multiple points of processing

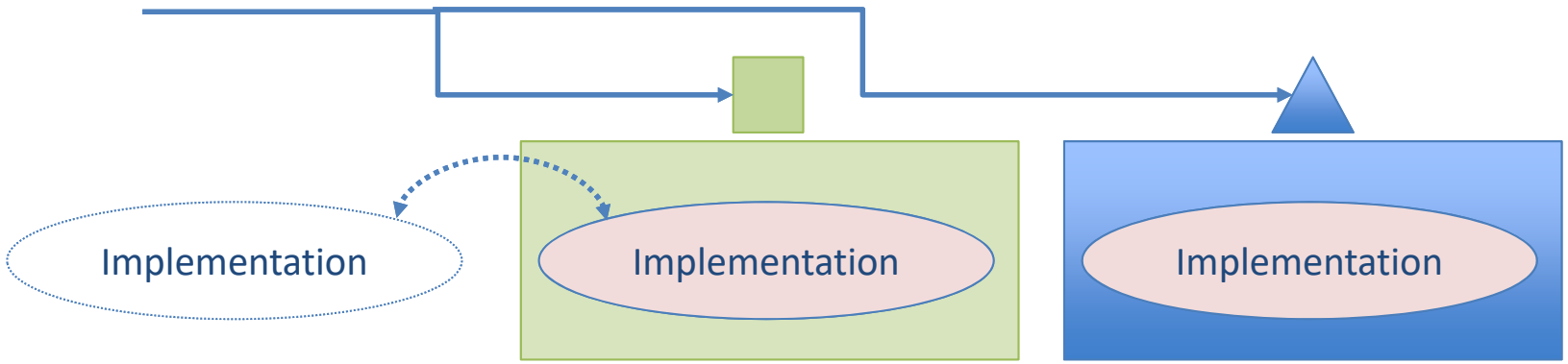
If DS is well designed e.g., with replication and recovery, the partial failure can be hidden

IMPLEMENTATION TRANSPARENCY

Its actual implementation is hidden from other components or users

Services of a component are exposed via the form of Interfaces

The server can replace/change the implementation without affecting other users or components



GOAL 3: OPENNESS

One of the requirements of the DS is to connect different components and to server different types of users (Heterogeneity)

In order for these different types of components and users to understand and work together, DS should be open

To achieve openness, DS should:

- Use standard protocols
- Expose components' functionalities via Interfaces
- Easily be configured in case changes needed

PROTOCOLS

In DS, components distributed over the network communicate via sending/receiving messages

In order for components to understand one another, the messages must follow a set of rules (called protocol) about

- Format of the messages
- Sequence (order) of the messages

Example, HTTP is used to request/reply web pages, SMTP and POP3 are used to send/receive emails, etc.

INTERFACES

Services of DS are usually described in form of Interface Description Language (IDL)

IDL nearly always captures only the syntax of the service

- Names of the functions, input parameters and return values
- Possibly exceptions can be raised

EASY CONFIGURATION

Open DS should be easy to configure out of different components

It should be easy to add new components or replace existing ones without affecting those components stay in place

This means open DS should also be **extensible**.

GOAL 4: SCALABILITY

Scalability is measured in three dimensions, its size, geography, and administration

- Scalable with respect to size means we can add more users and resources to the system
- Scalable with respect to geography means users and resources may lie far apart
- Scalable with respect to administration means it can still be easy to manage even if it spans many different administrative organizations

SCALABILITY AND SCALABILITY TECHNIQUES

When more users and resources are added, we often confronted with the limitations of centralized services, data and algorithms

So we need to decentralize the services, data, and algorithms using scaling techniques

Scaling techniques:

- Hiding communication latencies
- Distribution
- Replication
- Caching

HIDING COMMUNICATION LATENCIES

Try to avoid waiting for responses to remote service requests as much as possible

For instance, after making request, immediately do other useful work while waiting for a reply (asynchronous communication)

DISTRIBUTION

Distribution involves taking a component

- Split it into smaller parts
- Spread these parts across the network

Example

- The web is physically distributed across number of servers
- Each handling a number of web documents

REPLICATION

Replication not only increases availability, but also helps to balance load between components leading to better performance

Example, having a copy nearby can hide much of the communication latency mentioned above

CACHING

Caching is a special form of replication

Caching is a decision made by the client of a resource and not by the owner of a resource

Caching happens on demand whereas replication is often planned in advance

Introduction

TYPES

OF DISTRIBUTED SYSTEMS

Types of Distributed Systems

There are three main types of DSs:

- Distributed Computing Systems
- Distributed Information Systems
- Distributed Embedded Systems (Pervasive)

TYPE 1: DISTRIBUTED COMPUTING SYSTEMS

Used for high-performance computing tasks

There are two subgroups: Cluster computing and Grid computing

Cluster Computing (homogeneity):

- Constructed by collection of similar hardware
- Connected by high speed LAN
- Each node normally runs the same OS

Grid Computing (heterogeneity):

- Constructed as a federation of computer systems
- Each may fall under different administrative domain
- May have different hardware, software
- May be deployed on different network technology

TYPE 2: DISTRIBUTED INFORMATION SYSTEMS

Information systems or applications operating over the network

Applications are separated into independent components

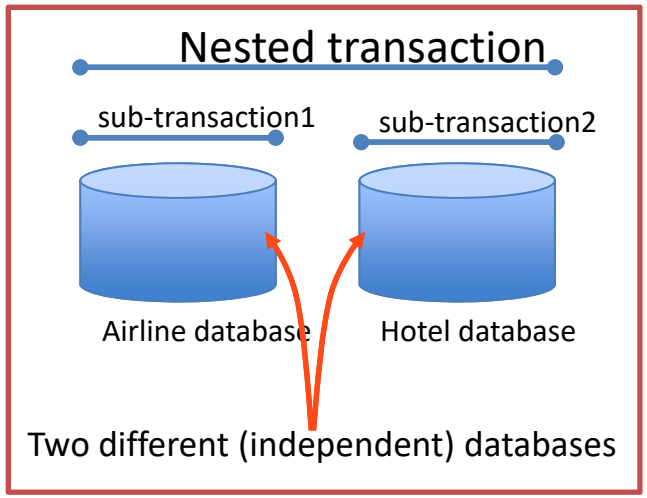
- Notably distinguish database from processing and presentation

We will focus on two main forms of this type

- Transaction Processing Systems
- Enterprise Application Integration

TRANSACTION PROCESSING SYSTEMS

Transaction is a logical unit of work to be performed with the system



In DIS, transaction may be constructed from number of sub-transactions

- Each can be executed in parallel on different machines
- Each can also execute one or more sub-transactions

ENTERPRISE APPLICATION INTEGRATION

Application components should be able to communicate directly with each other

- Not just by means of request/reply behavior that was supported by Transaction Processing Systems

The need for inter-application communication led to different communication models

- Help existing applications to directly exchange information

Several types of communication middleware exist

- RPC (Remote Procedure Call)
- RMI (Remote Method Invocation)
- MOM (Message Oriented Middleware e.g., Web services)

TYPE 3: DISTRIBUTED PERVASIVE SYSTEMS

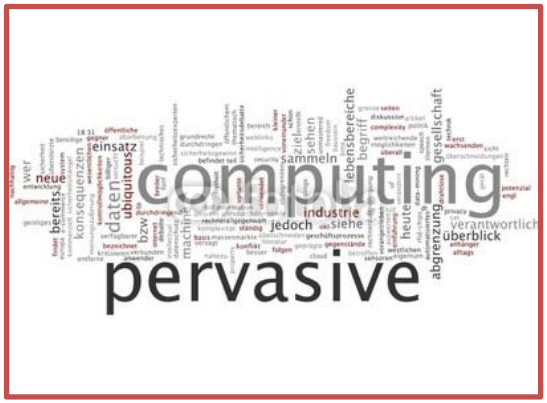
New class of DS in which instability is the default behavior

Components are often small, battery-powered, mobile, and having wireless connection

- Not all these characteristics apply to all devices

Some Concrete types of Distributed Pervasive Systems

- Home Systems
- Electronic Health Care Systems
- Sensor Networks



HOME SYSTEMS

Systems built around home networks

Generally consists of

- One or more personal computers
- Integrate with typical consumer electronics (TVs, audio video equipment, kitchen appliances, surveillance cameras, controllers for lightning, etc.)

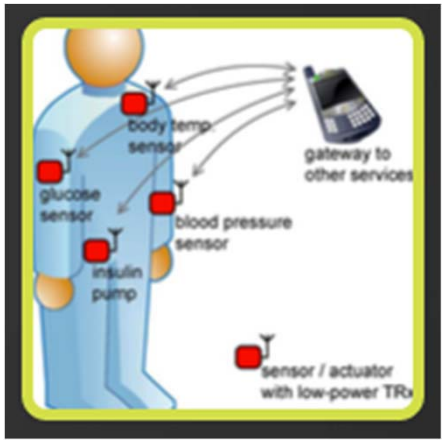
All hooked up into a single DS

ELECTRONIC HEALTH CARE SYSTEMS

Devices are used to monitor well-being of individuals and automatically contact physicians when needed

Often equipped with various sensors organized in a body-area network (BAN)

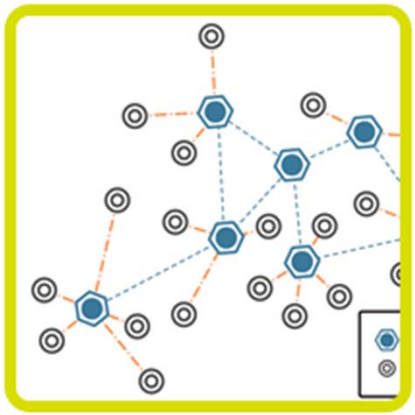
- BAN is a network surrounded the person
- The network should operate even while the person is moving



SENSOR NETWORKS

Sensor network

- Consists of tens to hundreds of nodes
- Uses wireless communication
- Nodes are often battery powered



Sensor networks can be considered as distributed databases

- Networks are deployed for measurement & surveillance applications

Lecture Recap

Enablers of Distributed Systems

Definition of Distributed Systems

Characteristics of Distributed Systems

Goals of Distributed Systems

Types of Distributed Systems

Main Characteristics of DS

- Heterogeneity – How align is a DS in supporting heterogeneity?
- Transparency – How well a DS hide the details of internal organization, communication and processing from users?
- Scalability – How well a DS can expand or scale up?
- Partial Failure – How well a DS reacts to failure?

4 Design Goals of DS

- **Accessibility**: Making resources easily accessible
- **Transparency**: Hiding the fact that resources are distributed
- **Openness**: Can integrate different hardware/software platforms from different administrative domains
- **Scalability**: Can add more hardware/software and serve more users

3 Types of Distributed Systems

- Distributed Computing Systems:
  - High performance computing tasks such as Cluster, Grid and Cloud.
- Distributed Information Systems:
  - Business information systems.
- Distributed Embedded Systems (Pervasive):
  - Small devices and components communicating over wireless, e.g. sensor networks, home automation, etc.

About Lab

- Lab will start from Week 2 and no lab in Week 1.

- Preparations for the lab in Week 2:
  - Bring a laptop
  - Download Java SDK and install on your laptop
  - Download and install the Eclipse IDE on your laptop (https://www.eclipse.org/downloads/packages/)
  - Learn how to program Java with Eclipse IDE
  - Study reference materials:
  - LMS->ICT2107->Labs->Week 2->"ICT2107 - Lab1 - Reference_UDP&TCP Illustration.pptx"