University of Freiburg, Germany
Department of Computer Science

Distributed Systems

Chapter 1 Introduction, Motivation, & Organization

Christian Schindelhauer

27. April 2014

1.1: Organization

Peter Fischer

- Web Science
- peter.fischer@informatik.uni-freiburg.de

Christian Schindelhauer

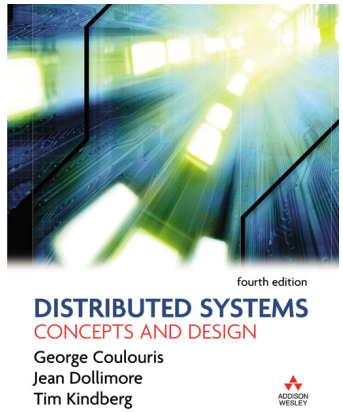
- Computer Networks and Telematics
- schindel@informatik.uni-freiburg.de

Organization

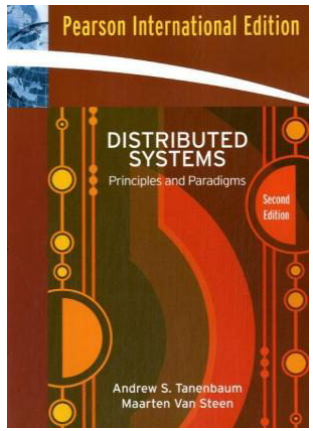
- Web-page
  `http://cone.informatik.uni-freiburg.de/lehre/aktuell/ds-ss14`
  - with slides, exercise, literature
- Forum
  `http://archive.cone.informatik.uni-freiburg.de/forum3/viewforum.php?f=28`
- Lecture (in rotation with exercise)
  - Monday 14-16, room 101-01-009/13 ✓
  - Thursday 14-16, room 101-01-009/13

Literature

- *Distributed Systems: Concepts and Design*. G. Coulouris, J. Dollimore, T. Kindberg. Addison Wesley, fourth edition 2005.



- *Distributed Systems*. A.S. Tanenbaum, M. Van Steen. Pearson Int. Edition, 2007.



- Further literature during the lecture

Lectures & Exercise 1st half (Schindelhauer)

- Mo 28.04.2014 Lecture Introduction, motivation, organization
- Mo 05.05.2014 Lecture System models
- Th 08.05.2014 Exercise
- Mo 12.05.2014 Lecture Time & global states
- Th 15.05.2014 Lecture Lamport clocks & consistent cuts
- Mo 19.05.2013 Lecture Failure models
- Th 22.05.2014 Exercise
- Mo 26.05.2014 Lecture Mutual exclusion, election
- Th 02.06.2014 Lecture Multicast, Consensus
- Mo 05.06.2014 Lecture Paxos
- Mo 16.06.2014 Exercise

Lectures & Exercise 2nd half (Fischer)

- Mo Mo 23.06.2014 Lecture 2nd part: Introduction
- Th 26.06.2014 Lecture Distributed system architectures, Transaction model
- Mo 30.06.2014 Lecture Distributed concurrency control
- Th 03.07.2014 Exercise
- Mo 07.07.2014 Lecture Reliability
- Lecture Replication, Petri Nets [part I] Fischer
- Lecture Petri Nets [part II]
- Th 17.07.2014 Exercise
- Mo 21.07.2014 Lecture Petri Nets [part III]
- Th 24.07.2014 Lecture Petri Nets [part IV]
- Mo 28.07.2014 Lecture MapReduce
- Th 31.07.2014 Exercise

Exercises & Exam

Exercises

- Voluntary exercises
- Every two weeks, two hours

Exam

- Master & bachelor students: oral exam
- Register online (in-time)
- Dates to be announced

Related Lectures

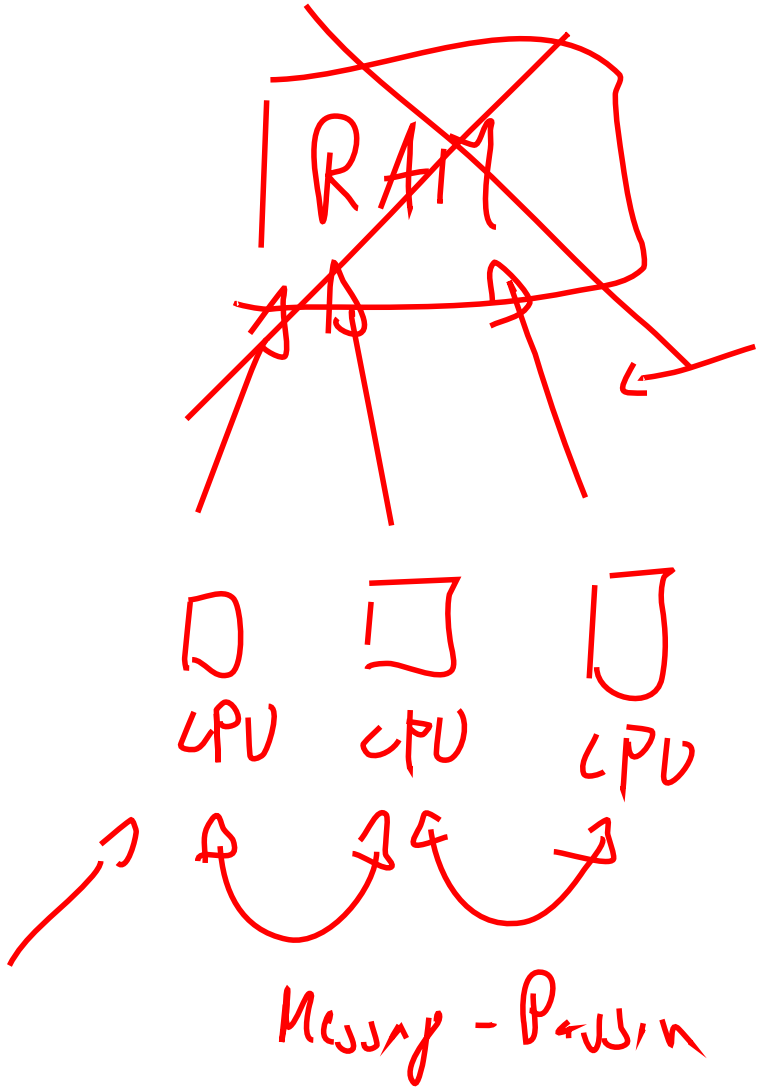
- This semester
  - Computer Networks / Rechnernetze I (Systeme II — Schindelhauer)
- Required knowledge:
  - Operating Systems/Betriebssysteme (Systeme I — Scholl)
- Continuing
  - Data Bases and Information Systems/Datenbanken und Informationssysteme (Lausen)
  - Distributed Algorithms (Kuhn)

1.2: Motivation

Distributed Systems are everywhere!!

- The Internet
- WWW
- Local Area Networks
- Multi-core processors
- Smart phones
- Massive Multiplayer Games
- Peer-to-Peer Networks
- Data centers
- ...

Cloud



Special Problems

Distributed Systems have special problems:

- How to organize a distributed system
- There is no global time
- Agreement with lazy, faulty and malicious partners
- Coordination of heterogeneous partners

1.3: Introduction

Definition: Distributed System (DS)

In a distributed system hardware or software components located at networked computers communicate and coordinate their actions **only by passing messages.**

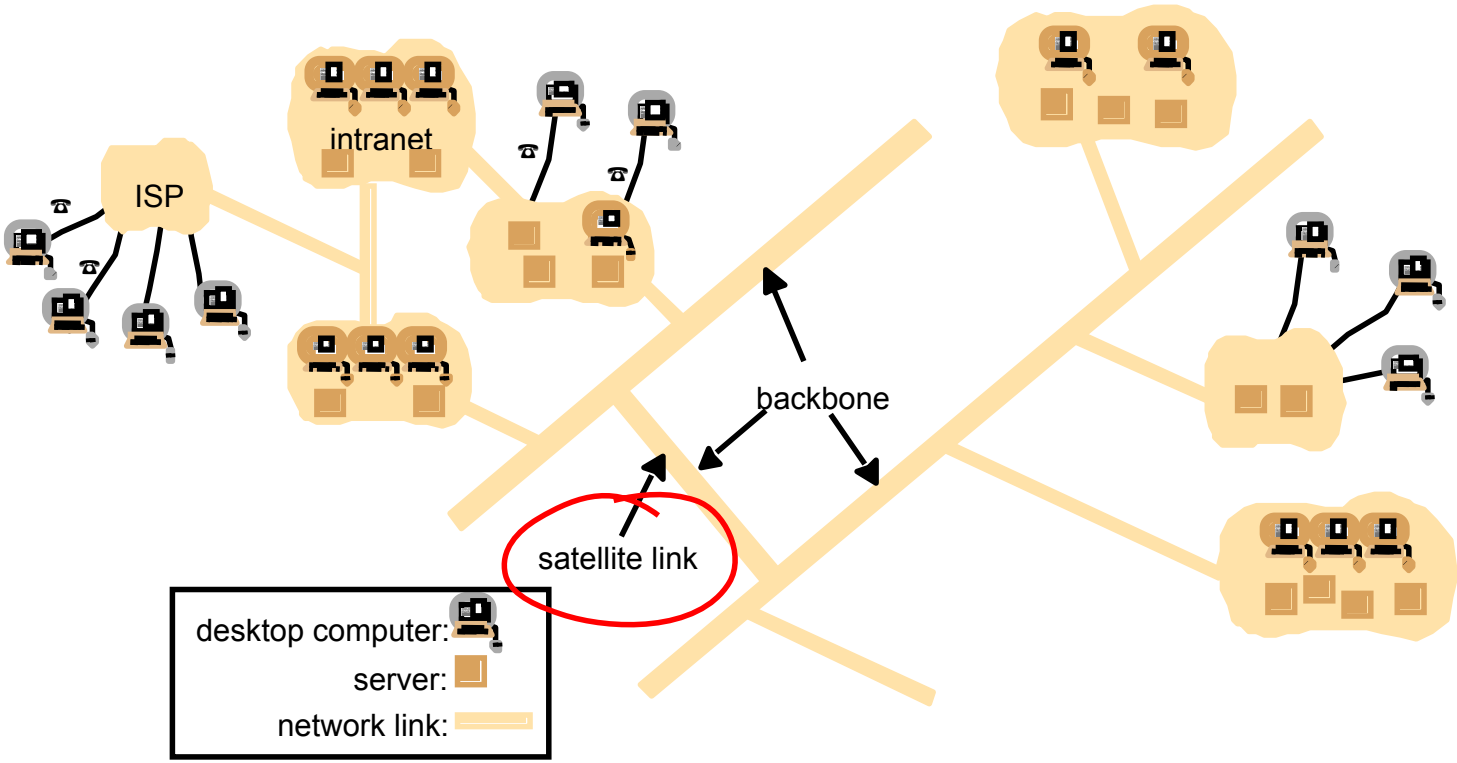
Consequences

- Concurrency
- No global clock
- Independent failures

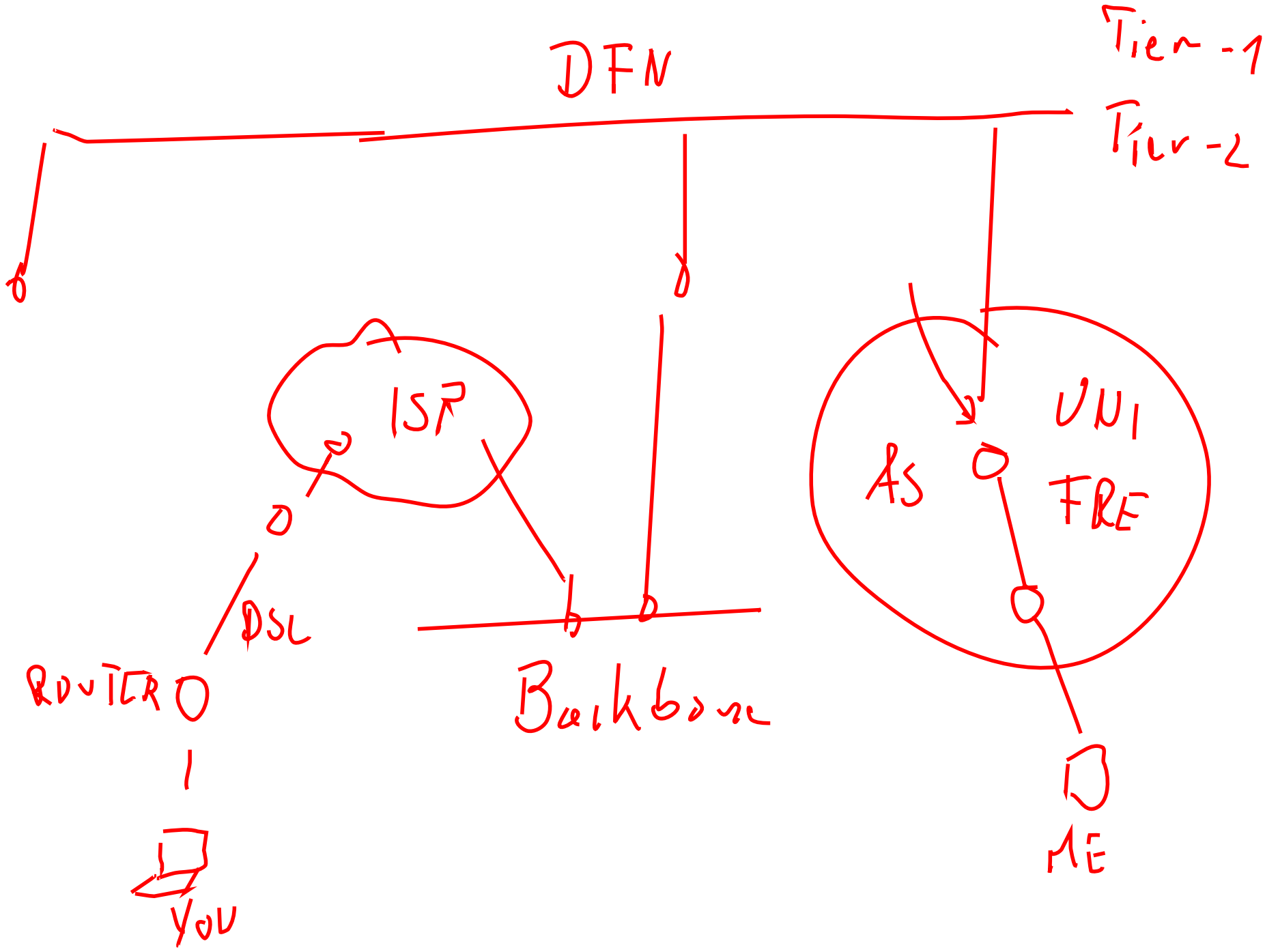
Examples of DS

- The Internet
- Intranets
- Mobile and ubiquitous computing

The Internet

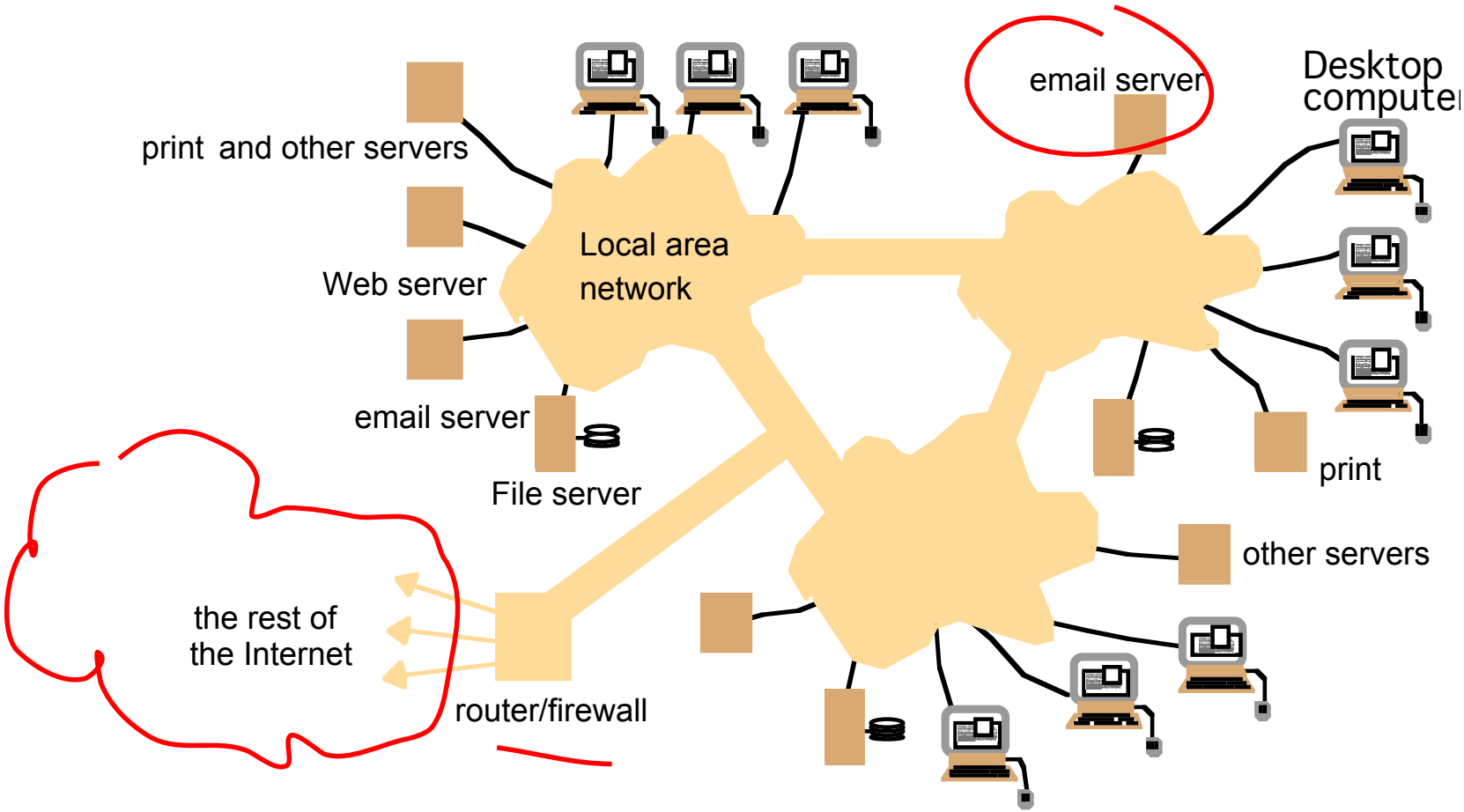


from *Distributed Systems – Concepts and Design*, Coulouris, Dollimore, Kindberg


DFN
Tier-1
Tier-2
ISP
DSL
ROUTER
You
Backbone
UNI
AS
FRE
ME

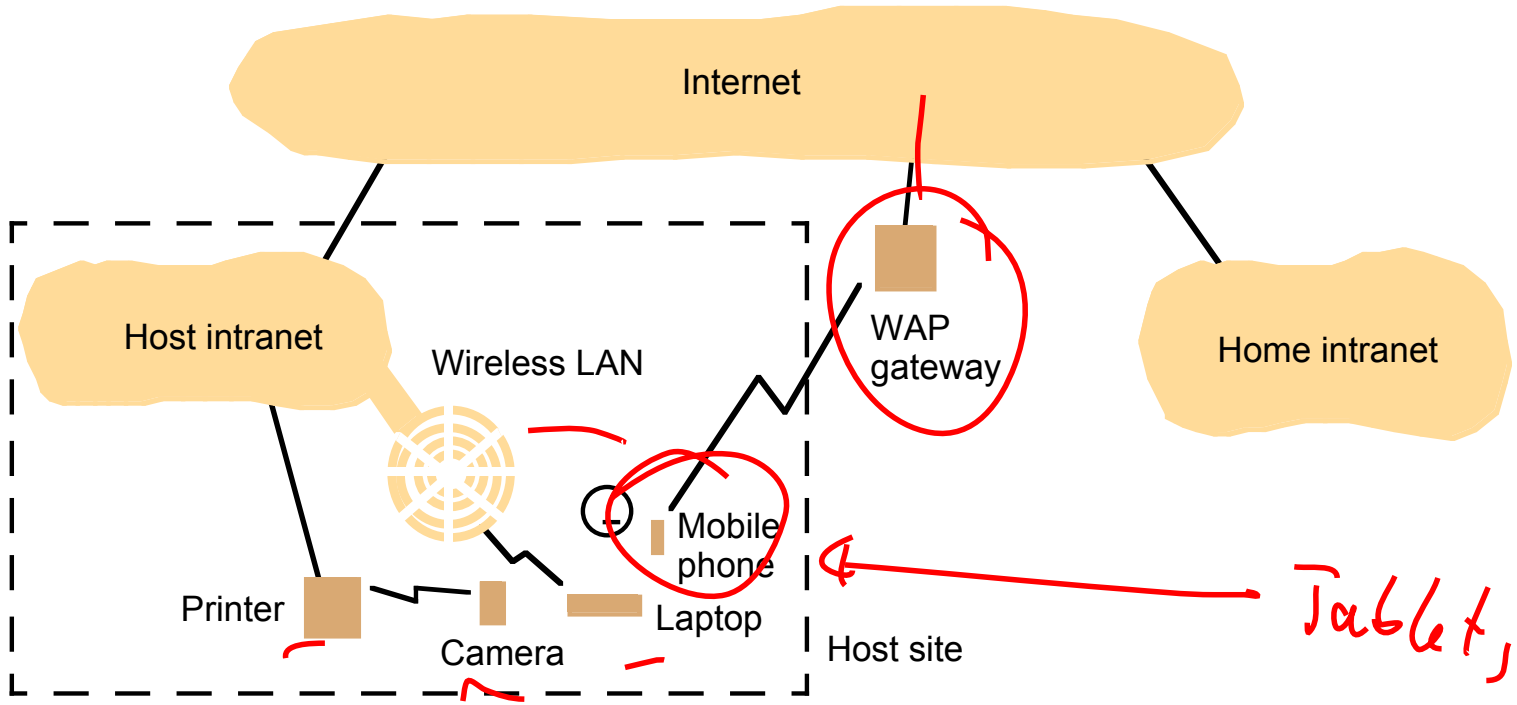
A Typical Intranet

AS Autonomous System



from *Distributed Systems – Concepts and Design*, Coulouris, Dollimore, Kindberg

Portable and Handheld Devices in a Distributed System



from *Distributed Systems – Concepts and Design*, Coulouris, Dollimore, Kindberg

Challenges of DS: Heterogeneity

- networks
- computer hardware
- operating systems
- programming languages
- implementations

Definition: Middleware

is a software layer that provides a programming abstraction wich masks the heterogeneity of the underlying networks, hardware, operation systems and programming languages.

Definition: Mobile Code

refers to code that can be sent from one computer to another and run at the destination.

Challenges of DS: Openness

By definition: the key interfaces of **open systems** are published

- Open distributed systems provide uniform communication mechanism
- Open DS publish interfaces for access to shared resources
- Open DS can be constructed from heterogeneous hardware and software
- Open DS must be carefully tested and verified.

Challenges of DS: Security

Components

- confidentiality
- integrity
- availability

Typical cases

- A doctor requesting access to hospital data.
- Electronic commerce and banking

Unsolved security challenges

- Denial of service attacks
- Security of mobile code

Challenges of DS: Scalability

A system is described as scalable

if it remains effective when there is a significant increase in the number of resources and the number of users.

- Controlling the cost of physical resources
- Controlling the performance loss
- Preventing software resources running out
- Avoiding performance bottlenecks

Challenges of DS: Failure Handling

Failures in a DS are partial. Some components fail, while other continue to function.

- Detecting failures
- Marking failures
- Tolerating failures
- Recovery from failures
- Redundancy

Challenges of DS: Concurrency

- Services and applications provide resources that can be shared
- Resources can be accessed at the same time
- A shared resource in a DS must ensure correct operation in a concurrent environmnent
- Operation must be synchronized such that the data of a shared object remains consistent

Challenges of DS: Transparancy

1. Access transparency
   enable local and remote resource to be accessed with identical operations
2. Location transparency
   access without knowledge of their physical location
3. Concurrency transparency
   concurrently operate several processes using shared resources
4. Replication transparency
   enables multiple instances of recources to be used to increase reliability and performance without the users knowing
5. Failure transparency
   concealment of faults, allowing users to complete their tasks despite failures
6. Mobility transparency
   allows the movement of resources and clients without affecting the operation
7. Performance transparency
   allows the system to be reconfigured to improve the performance as loads vary
8. Scaling transparency
   allows the system and application to expand in scale without a change to the system

RAID

End of Section 1