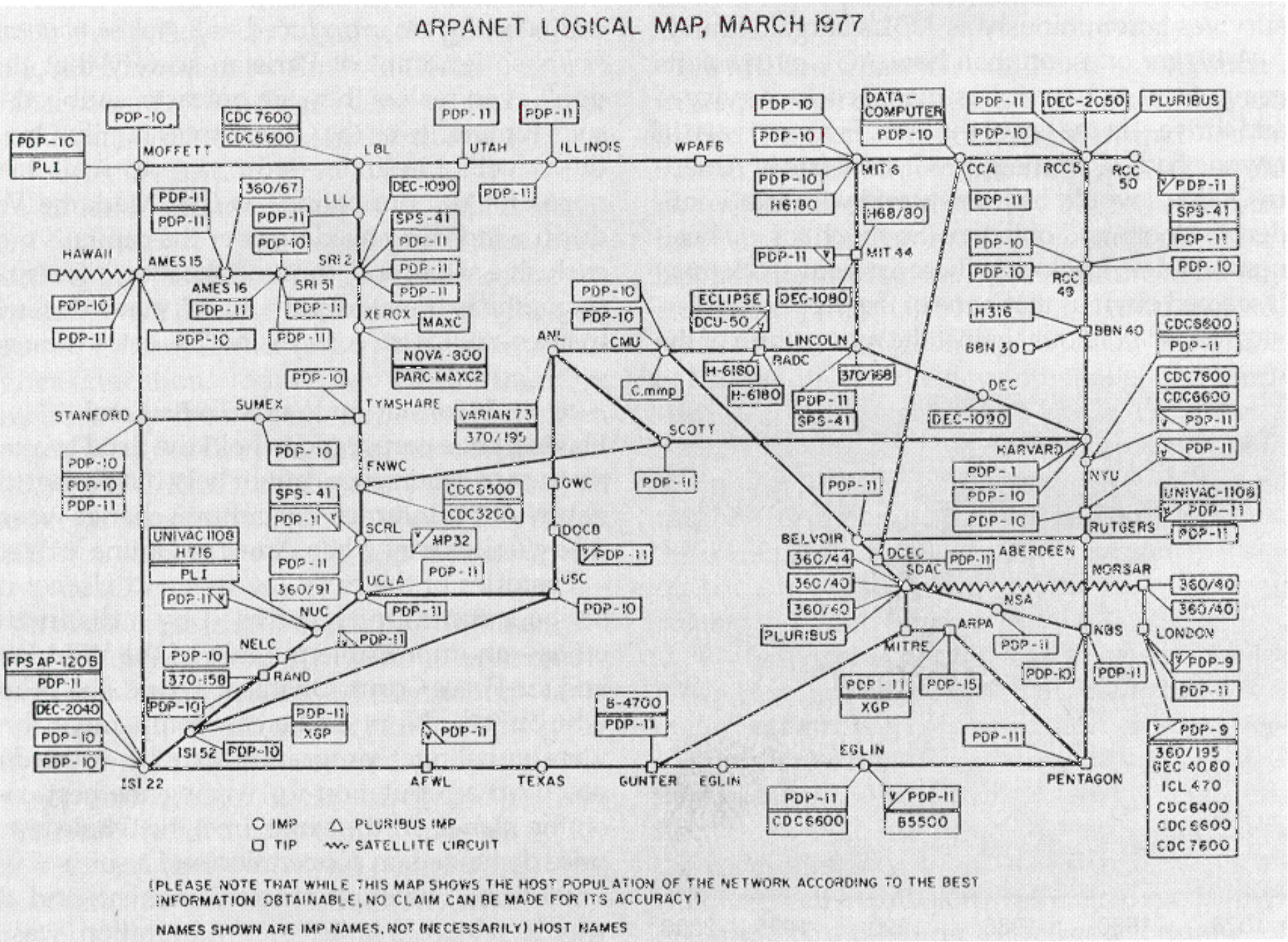
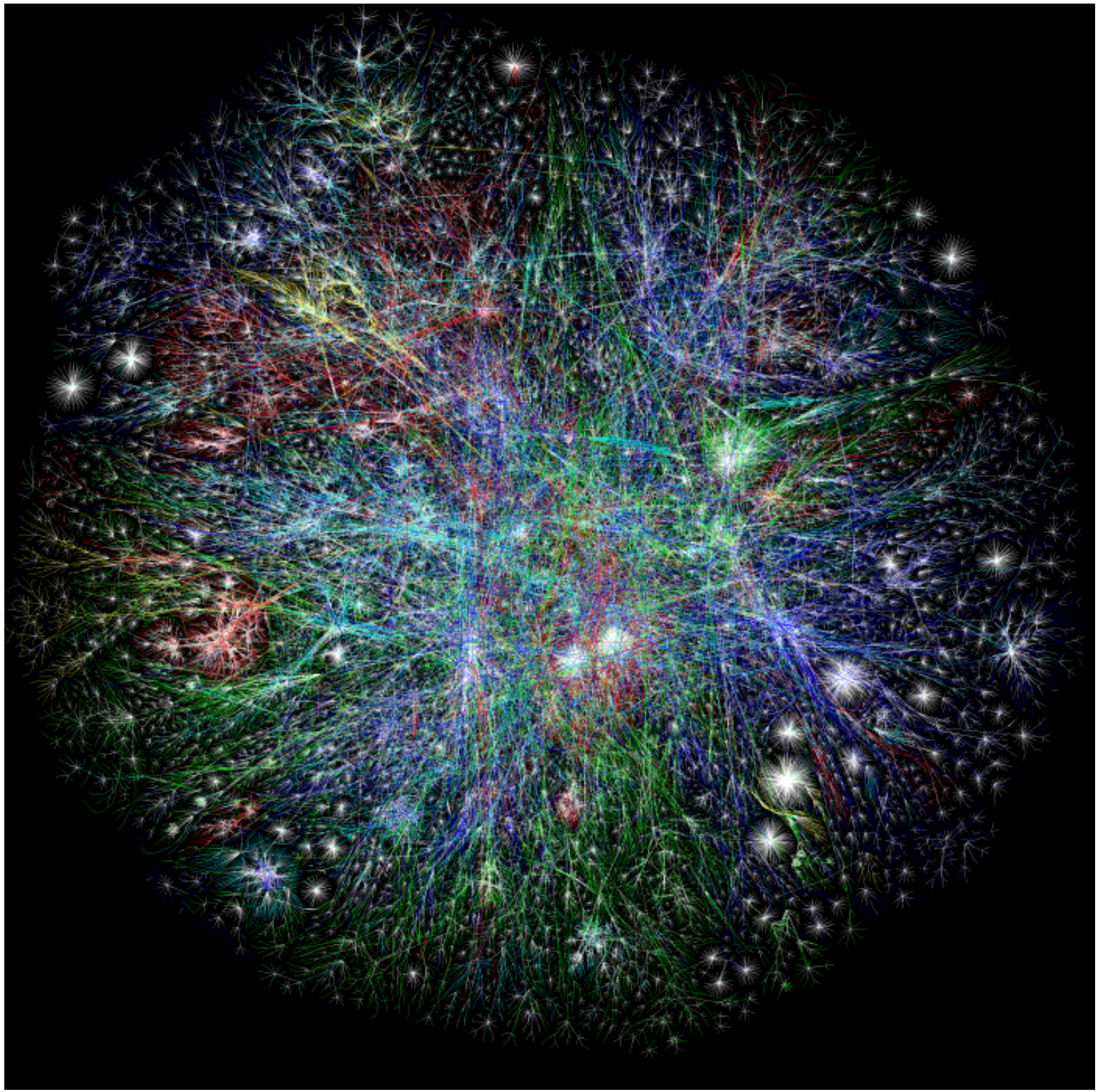
UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

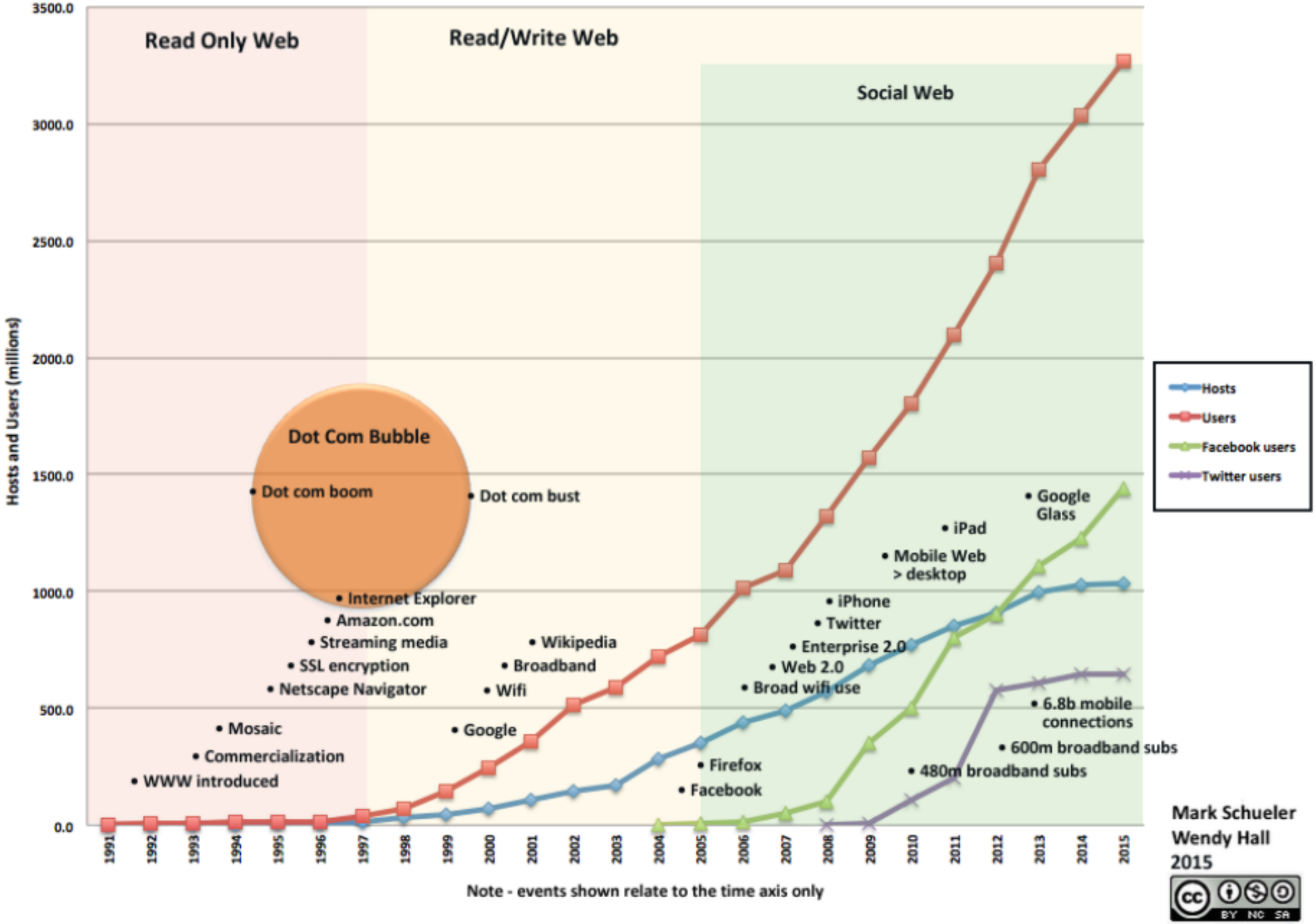
Distributed Systems 1

Gian Pietro Picco
Dipartimento di Ingegneria e Scienza dell'Informazione
University of Trento, Italy
gianpietro.picco@unitn.it
http://disi.unitn.it/~picco


ARPANET LOGICAL MAP, MARCH 1977
PDP-10
PDP-10
PL1
MOFFETT
CDC 7600
CDC 6600
LBL
PDP-11
UTAH
PDP-11
ILLINOIS
WPAFB
PDP-10
PDP-10
PDP-10
H6180
DATA-COMPUTER
PDP-10
MIT 6
H68/80
PDP-11
MIT 44
PDP-11
DEC-2050
PLURIBUS
PDP-10
PDP-10
CCA
PDP-11
RCC 5
RCC 50
PDP-11
SPS-41
PDP-11
PDP-10
RCC 49
PDP-11
PDP-11
360/67
LLL
PDP-11
PDP-10
DEC-1090
SPS-41
PDP-11
PDP-11
PDP-11
HAWAII
AMES 15
SRI 2
AMES 16
PDP-11
SRI 51
PDP-11
PDP-10
PDP-10
PDP-11
PDP-10
PDP-11
XEROX
MAXC
NOVA-800
PARC MAXC2
PDP-10
PDP-10
ANL
CMU
C.mmp
ECLIPSE
DCU-50
DEC-1080
H-6180
H-6180
LINCOLN
RADC
370/158
PDP-11
SPS-41
H316
BBN 30
BBN 40
CDC6600
PDP-11
CDC7600
CDC6600
PDP-11
PDP-11
DEC
DEC-1090
PDP-10
STANFORD
SUMEX
PDP-10
TYMSHARE
VARIAN 73
370/195
SCOTT
HARVARD
PDP-1
PDP-10
PDP-10
NYU
UNIVAC-1108
PDP-11
RUTGERS
PDP-11
PDP-10
PDP-10
PDP-11
FNWC
SPS-41
PDP-11
CDC6500
CDC3200
GWC
PDP-11
PDP-11
BELVOIR
DCEC
PDP-11
SDAC
ABERDEEN
360/44
360/40
360/60
PLURIBUS
NORSAR
360/40
360/40
NSA
UNIVAC 1108
H716
PLI
PDP-11
360/91
SCRL
MP32
UCLA
PDP-11
NUC
PDP-11
DOCB
PDP-11
USC
PDP-10
PDP-11
PDP-11
ARPA
MITRE
PDP-11
PDP-11
XGP
PDP-15
NDS
PDP-10
PDP-11
LONDON
PDP-9
PDP-11
NELC
PDP-10
370/158
RAND
PDP-11
FPSAP-120B
PDP-11
DEC-2040
PDP-10
PDP-10
PDP-10
ISI 52
PDP-10
PDP-11
XGP
PDP-11
AFWL
B-4700
PDP-11
GUNTER
TEXAS
EGLIN
EGLIN
PDP-11
CDC6600
PDP-11
B5500
PDP-11
PENTAGON
PDP-9
360/195
GEC 4080
ICL 470
CDC6400
CDC6600
CDC7600
ISI 22
IMP
TIP
PLURIBUS IMP
SATELLITE CIRCUIT
(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)
NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES



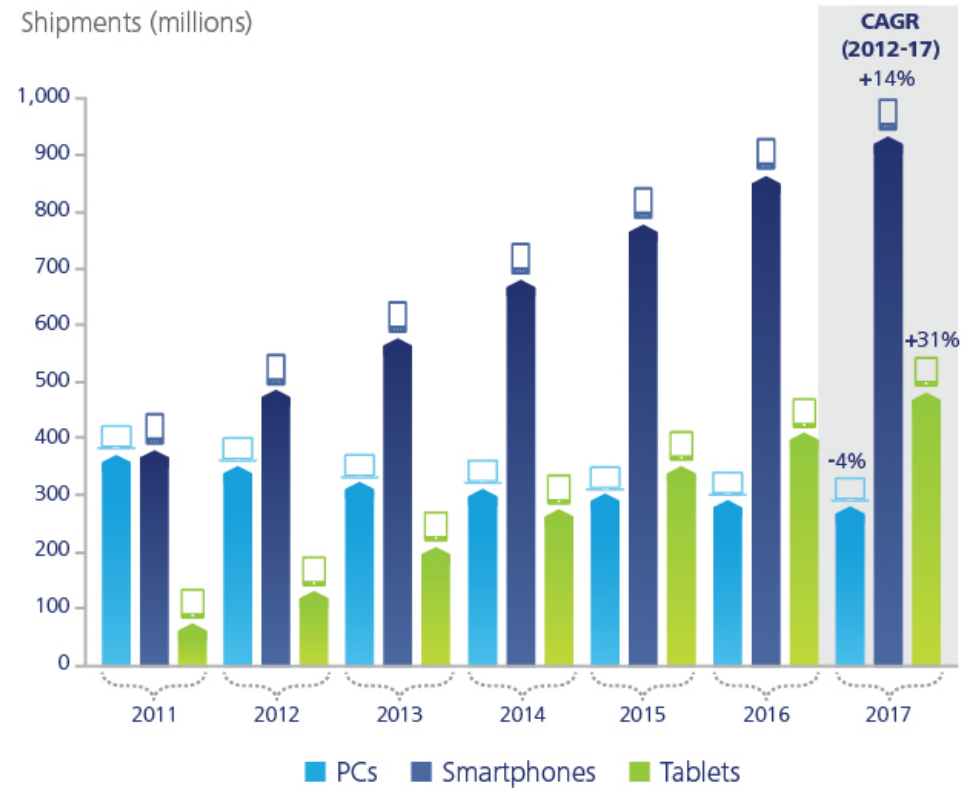
Internet Growth - Usage Phases - Tech Events



Mark Schueler
Wendy Hall
2015

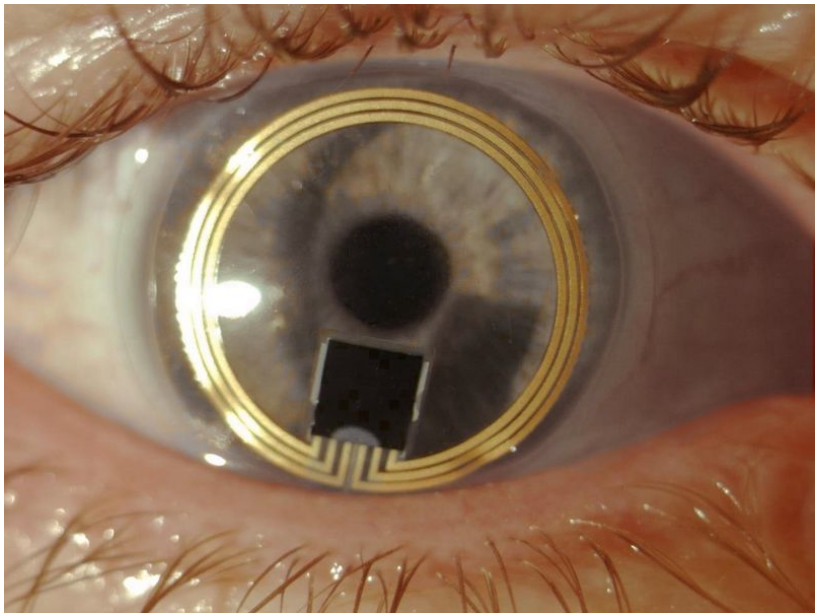
2017 This Is What Happens In An Internet Minute

Recent Game Changers



Source: Gartner (2013)

facebook®



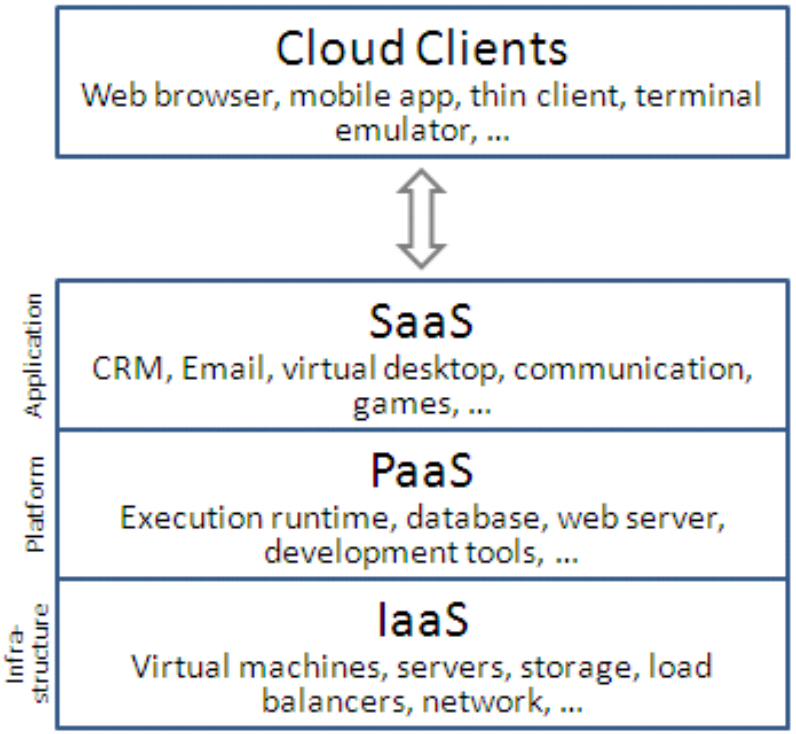
Some of Today's Trends
Data Centers

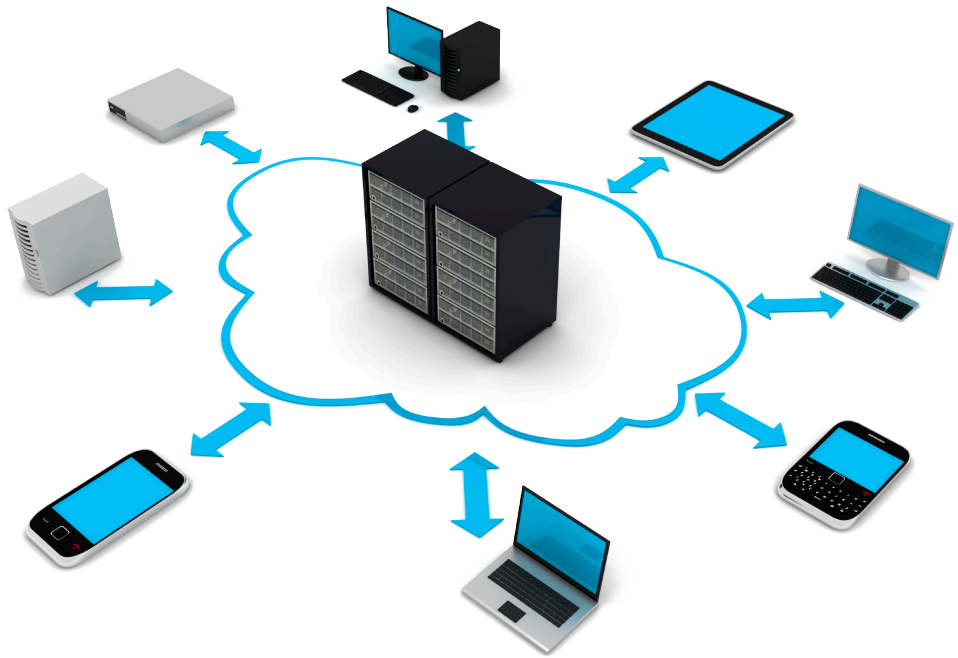
- Servers are more and more powerful (and cheap)
- They can be "aggregated" to yield high-performance, reliable, scalable computing
  - at a fraction of the cost of a mainframe
  - fundamental role of software and algorithms!
- Data centers are typically composed of thousands machines
  - hosting several virtual machines
  - organized in racks along with high-performance disks
  - connected by high-speed networks
  - with dedicated network stacks

A glimpse of the UNITN data center in Povo2

Some of Today's Trends
Cloud Computing

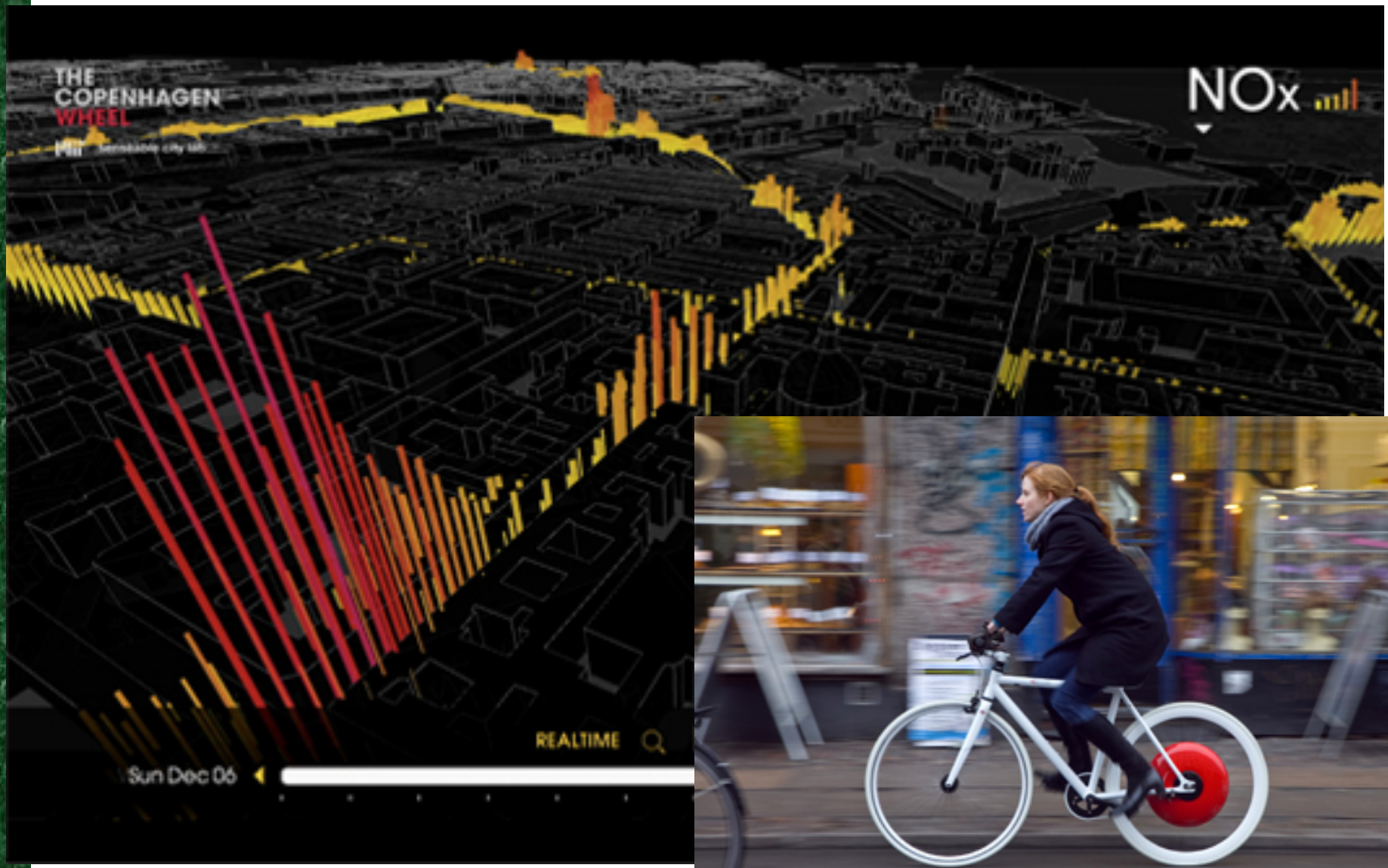
- ***Cloud computing*** pushes the concept further, providing (via data centers) users with generic services, at various layers (utility computing)
  - maintenance and operation costs are sustained by the cloud provider instead of the end user





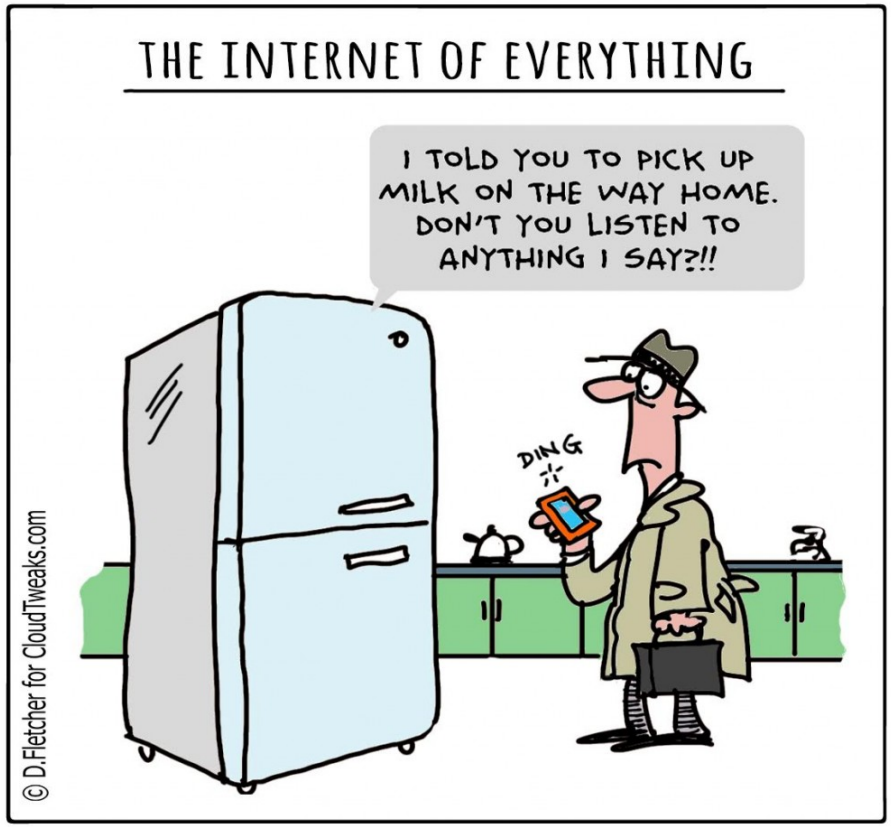
Some of Today's Trends:
Wearable Computing, Robotics

Some of Today Trends:
Mobile Sensing + Mining Mobile Big Data



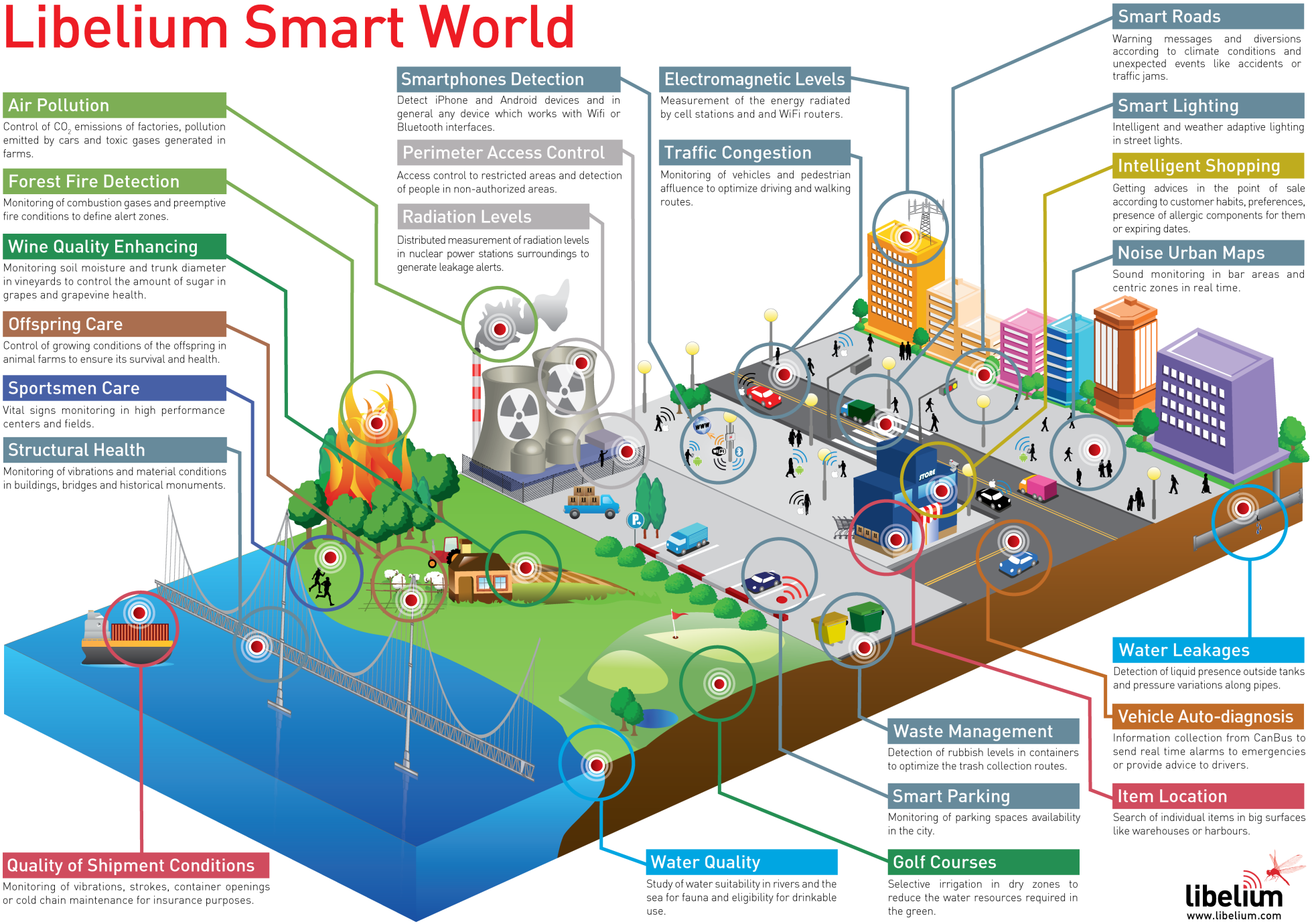
Some of Today Trends:
The Internet of Things

- In 2010, the Ericsson CEO publicly stated that, by 2020, there would 50 billion connected devices*



** in 2015, they revised the estimate to 28B by 2021*

Libelium Smart World



Air Pollution
Control of CO_2 emissions of factories, pollution emitted by cars and toxic gases generated in farms.

Forest Fire Detection
Monitoring of combustion gases and preemptive fire conditions to define alert zones.

Wine Quality Enhancing
Monitoring soil moisture and trunk diameter in vineyards to control the amount of sugar in grapes and grapevine health.

Offspring Care
Control of growing conditions of the offspring in animal farms to ensure its survival and health.

Sportsmen Care
Vital signs monitoring in high performance centers and fields.

Structural Health
Monitoring of vibrations and material conditions in buildings, bridges and historical monuments.

Quality of Shipment Conditions
Monitoring of vibrations, strokes, container openings or cold chain maintenance for insurance purposes.

Smartphones Detection
Detect iPhone and Android devices and in general any device which works with Wifi or Bluetooth interfaces.

Perimeter Access Control
Access control to restricted areas and detection of people in non-authorized areas.

Radiation Levels
Distributed measurement of radiation levels in nuclear power stations surroundings to generate leakage alerts.

Electromagnetic Levels
Measurement of the energy radiated by cell stations and and WiFi routers.

Traffic Congestion
Monitoring of vehicles and pedestrian affluence to optimize driving and walking routes.

Water Quality
Study of water suitability in rivers and the sea for fauna and eligibility for drinkable use.

Golf Courses
Selective irrigation in dry zones to reduce the water resources required in the green.

Waste Management
Detection of rubbish levels in containers to optimize the trash collection routes.

Smart Parking
Monitoring of parking spaces availability in the city.

Smart Roads
Warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.

Smart Lighting
Intelligent and weather adaptive lighting in street lights.

Intelligent Shopping
Getting advices in the point of sale according to customer habits, preferences, presence of allergic components for them or expiring dates.

Noise Urban Maps
Sound monitoring in bar areas and centric zones in real time.

Water Leakages
Detection of liquid presence outside tanks and pressure variations along pipes.

Vehicle Auto-diagnosis
Information collection from CanBus to send real time alarms to emergencies or provide advice to drivers.

Item Location
Search of individual items in big surfaces like warehouses or harbours.

libelium
www.libelium.com

What Is a Distributed System?

"pragmatic"

A collection of independent, autonomous hosts connected through a communication network. Hosts communicate via message passing to achieve some form of cooperation.

"A collection of independent computers that appears to its users as a single coherent system" (A.S. Tanenbaum, M. van Steen)

"optimistic"

"by negation"

A parallel system in which there are no:

- *shared, global clock*
- *shared memory*
- *accurate failure detection*

"One in which I cannot get any work done because some machine I have never heard of has crashed" (L. Lamport)

"pessimistic"

Why Distributed Systems?

- Distribution as a ***requirement***
  - Applications that require sharing of resources or dissemination of information among geographically distant entities are “natural” distributed systems
  - Examples: banks with several branches, distributed file systems, pervasive systems, …
- Distribution as a ***design/engineering choice***
  - To satisfy non-functional requirements like fault-tolerance, performance vs. cost ratio, quality of service, …
  - Examples: replicated servers

Some Defining Features

- ***Concurrency***
  - In centralized systems, concurrency is a design choice
  - In distributed systems, it is a fact of life: computers are always (co)-operating at the same time
- ***Absence of a global clock***
  - In centralized systems, the computer's physical clock can be used for the purpose of synchronization
  - In distributed systems, clocks are many and not necessarily synchronized
- ***Absence of a shared memory***
  - In centralized systems, process may coordinate by using a memory area (e.g., a variable) accessible to all of them
  - In distributed systems, the processes reside on different machines with distinct memory address spaces; (asynchronous) message passing is the only means of coordination
- ***Independent (and partial) failures***
  - Centralized systems typically fail completely
  - Distributed systems fail only partially, often due to communication; recovery is complicated by the fact that the application state is itself distributed

Delays in Distributed Systems

Operation	Delay
CPU calculation	0.000001 ms
Parallel computing (MPI) latency	0.001-0.01 ms
Overhead of method call	0.001-0.01 ms
Wake up thread or process	1 ms
Internet delays	10-1000 ms

- This is one of the reasons why the absence of shared memory introduces a great deal of complexity in distributed systems...

More on (Distributed) Failures

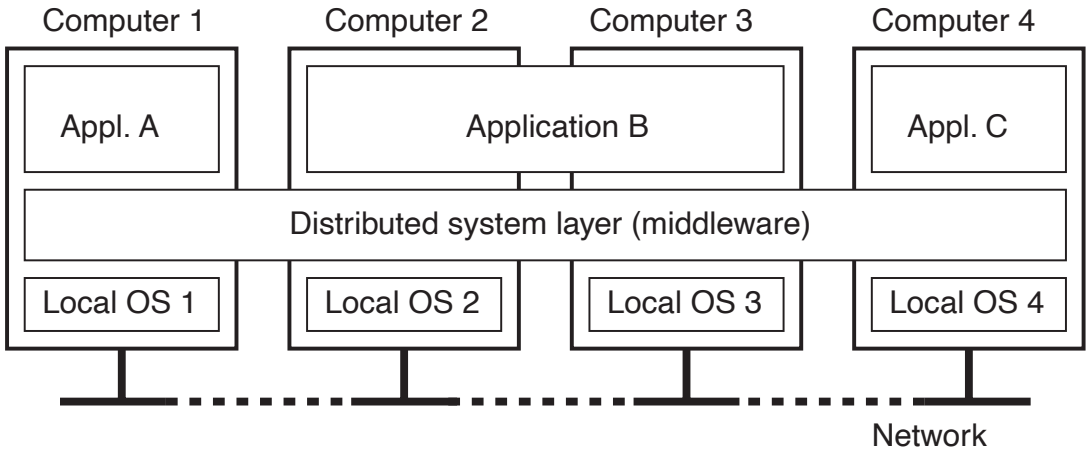
- Failures are a ***problem*** for distributed systems
  - Dependencies among hosts amplify the effect of failures
  - Distributed systems are more complex to design
  - Example: n dependent hosts, each with probability of failure p ➔ availability is $(1-p)^n$
- Distributed systems can be a ***solution*** to mitigate failures
  - Providing replicated services with multiple independent processes enables fault tolerance
  - Distributed systems can be made very reliable
  - Example: n **in**dependent hosts, each with probability of failure p ➔ availability is $1-p^n$

Parallel vs. Distributed

- ***Parallel*** systems are composed of multiple processors that:
  - ***share memory***
    - used by processors to coordinate their activities
  - ***share time***
    - processors access a common clock
  - ***share fate***
    - no independent failure
- Massively parallel systems are typically realized via custom hardware architectures
  - Multi-core introduce some amount of parallelism
- The different characteristics of parallel and distributed systems induce different design goals
  - Parallel systems exploit determinism to achieve efficiency
  - Distributed systems are inherently more “loosely coupled” and address the uncertainty this induces

Challenge: Heterogeneity

- Parallel systems are homogeneous; distributed systems are not, due to differences in
  - network
  - computing hardware
  - operating system
  - programming language
  - multiple implementations (versions)



Middleware is a software layer that abstracts from the differences above, providing a uniform computational model

Goal: Openness

- An open distributed system offers services according to standard rules that describe the syntax and semantics of those services
- These access rules are formalized in (standardized and/or public) protocols or interfaces
- Openness fosters
  - interoperability
    - think about TCP/IP and what it did for the Internet
  - portability
  - extensibility
- Examples: CORBA, JavaEE, Web services, …

Challenge & Goal: Security

- Many of the advantages of distributed systems backfire when security is at stake
  - e.g., open standards make it easy for (malicious) nodes to join the distributed system
- Key security aspects:
  - ***confidentiality***: information should be disclosed only to authorized parties
  - ***integrity***: information should be altered only by authorized parties; unauthorized alterations should be detectable and recoverable
  - ***availability***: services should be provided in spite of malicious behavior
- We will not discuss further these aspects; see other courses in the curriculum

Challenge: Failure Handling

- Dealing with failures is key in distributed systems
- Several techniques exist, with different goals:
  - failure detection (e.g., message checksums)
  - failure masking (e.g., email retransmissions)
  - failure tolerance (e.g., replicated servers)
  - failure recovery (e.g., log files)
- Their are not always applicable (e.g., masking)
- Real systems use a combination of the above, depending on the application requirements and fault model

Goal: Scalability

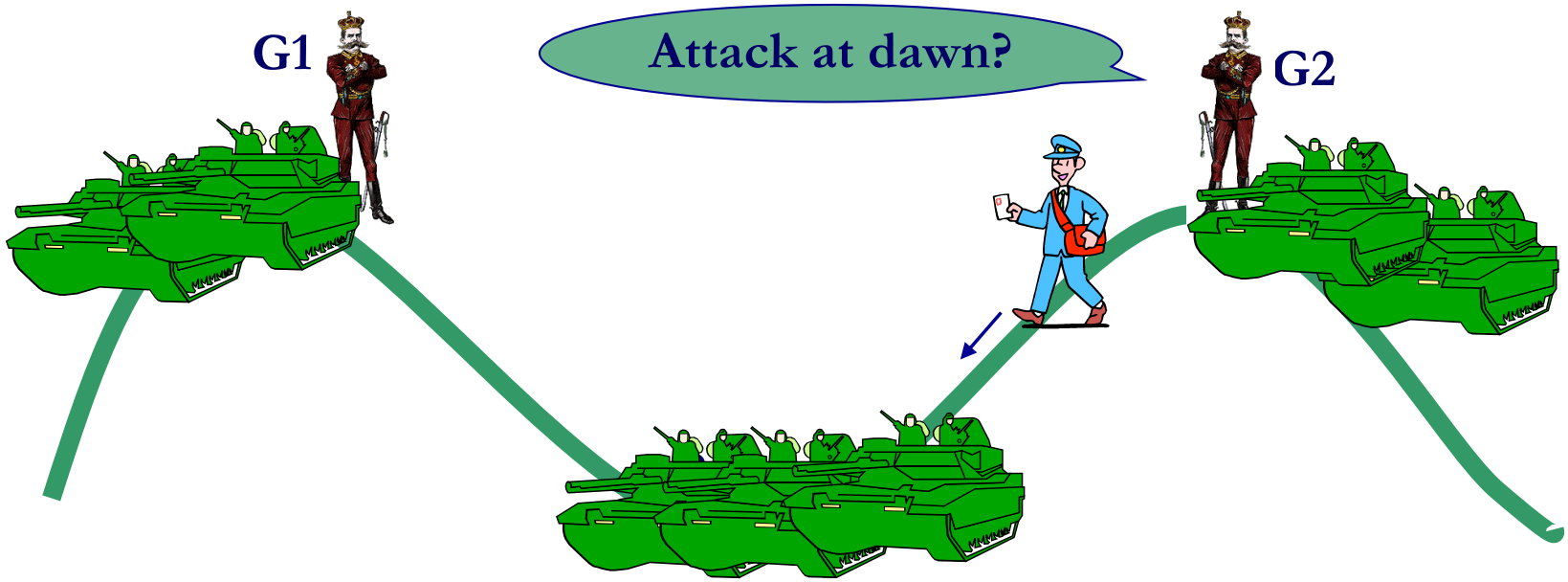
- It is the ability to increase the scale of the system in terms of:
  - size, i.e., number of users, hosts, or resources
  - geographic span
  - administrative span
- Tradeoff between scale and performance
- Centralized solutions are easier to design and reason about, but may yield to
  - a performance bottleneck…
  - … and a single point of failure
  - centralization of services, data, algorithms
- Decentralized solutions are desirable
  - e.g., using asynchronous communications, hierarchical structuring, caching and replication, …
- In practice, systems often use a mix
  - true for e.g., several services in the Google back-end

Goal: Transparency

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Replication	Hide that a resource may exist in multiple copies
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

- Some things cannot be made transparent
  - Timezones, communication delays
- Hiding too much may have a strong, negative performance impact
  - E.g., accessing repeatedly a remote object without knowing

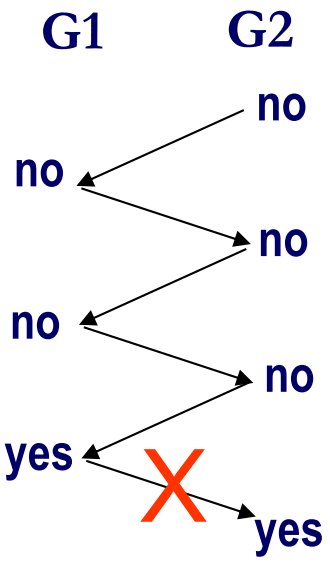
A Seemingly Easy Problem …



- Generals decide separately whether or not to attack
  - If ***both*** attack: success
  - If only one attacks: failure
- Messengers are employed for coordinating the decision
  - But they can be lost/captured/delayed
- ***Question:*** How do the generals ensure that they take the same decision? What algorithm do they use?

(Theoretical) Impossibility of Distributed Consensus

- It is impossible to design an algorithm to guarantee success or no attack if the messengers can be lost:
  - **Impossibility of distributed consensus**
  - Formally demonstrated (Fischer, Lynch, Patterson, 1985)
- (Informal) proof:
  - Reaching an agreement on one of two possible decisions requires the successful arrival of at least one message
  - Consider scenario A in which the **fewest** delivered messages that will result in agreement to attack are delivered
  - Let scenario B be the same as A except that the last message delivered in A is lost in B, and any other messages that might be sent later are also lost
  - Suppose this last message is from General 1 to General 2
  - General 1 sees the same messages in both scenarios, so he definitely attacks
  - However, the **minimality** assumption of A implies that General 2 cannot also decide to attack in scenario B, so he must make a different decision
  - Hence the problem is unsolvable



What does it mean in practice?

Distributed Consensus in Practice

- Does it really matter in real life? ***Yes!!!***
  - E.g., when you withdraw 100 EUR (at the ATM), your balance (at the bank) should be decreased by 100 EUR
    - This is the "commit" of a transaction in a distributed database, over a communication network
  - Agree on values of replicated, distributed sensors
  - Agree on whether a system component is faulty
- Then we cannot design distributed systems? ***No!!!***
- In practice, real systems either
  - Change the assumptions
    - E.g., make links reliable (enough)
  - Reduce the guarantees:
    - E.g., only probabilistic instead of deterministic

> ***In theory, theory and practice are the same. In practice, they are not.***
>
> *-- Yogi Berra*

The Course

- Goals:
  - lectures:
    - discuss fundamental concepts in distributed systems
  - labs:
    - understand some of these concepts hands-on …
    - … while experiencing with software technology for developing distributed systems
- Macro-topics:
  - synchronization
  - replication and consistency
  - fault tolerance
  - middleware