POLITECNICO
MILANO 1863

Distributed Systems + Middleware Introduction

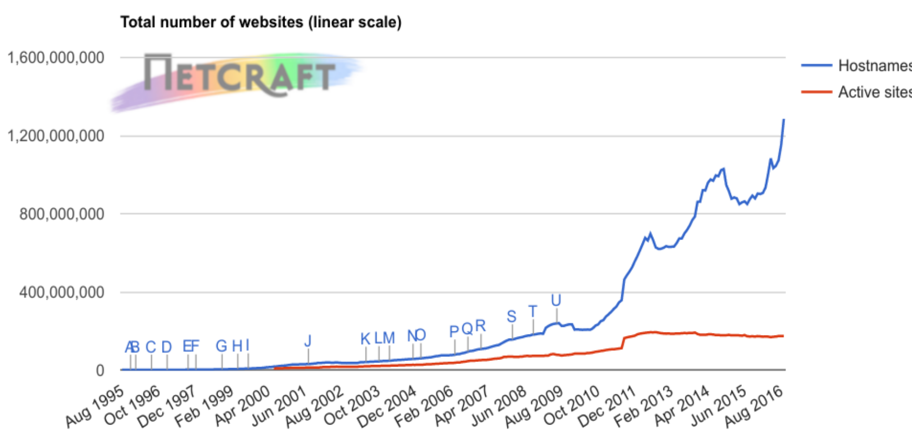
Gianpaolo Cugola

Dipartimento di Elettronica e Informazione
Politecnico di Milano, Italy

cugola@elet.polimi.it
http://home.dei.polimi.it/cugola

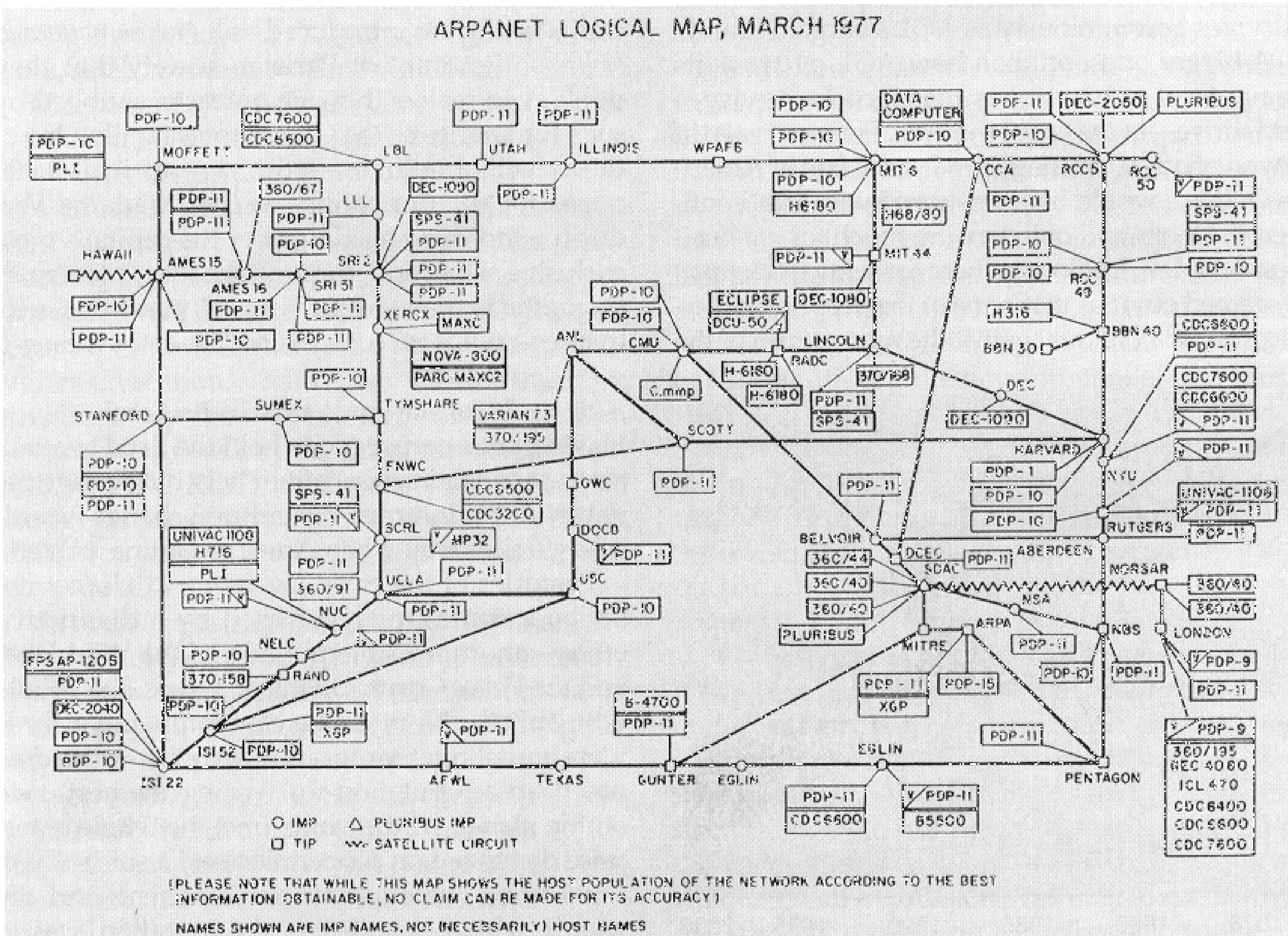
Why?

- Because most applications we use today are "distributed"
  - E-mail is probably the oldest and largest distributed application (billions of users,
    ~2.5M of emails per second)
  - The Web (~1B registered Web server names today)
  - Web applications, like Google maps, Google docs, Microsoft office Web apps (office 365)

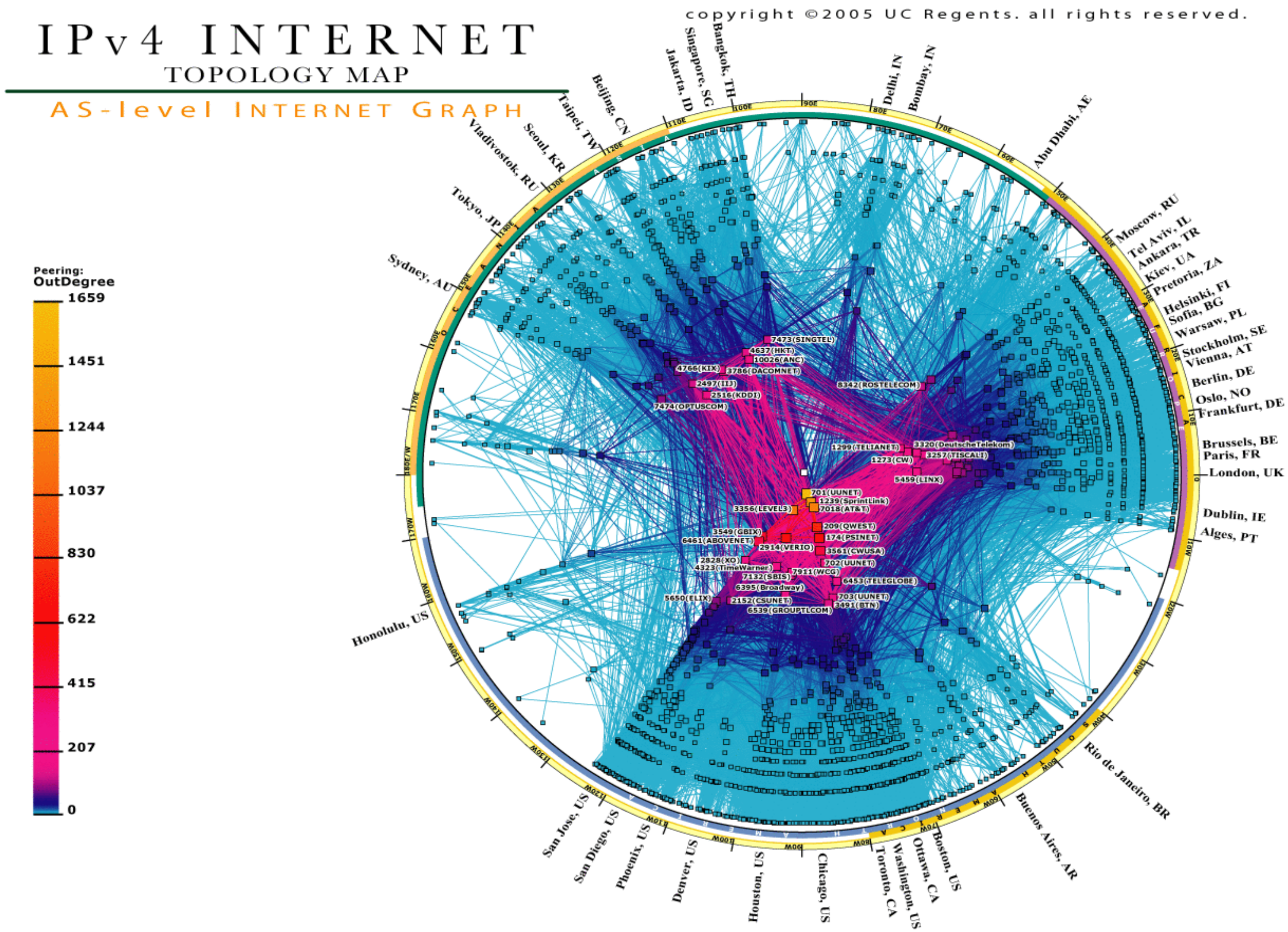


- Google itself is a distributed system…

Distributed systems: Yesterday



Distributed systems: Today

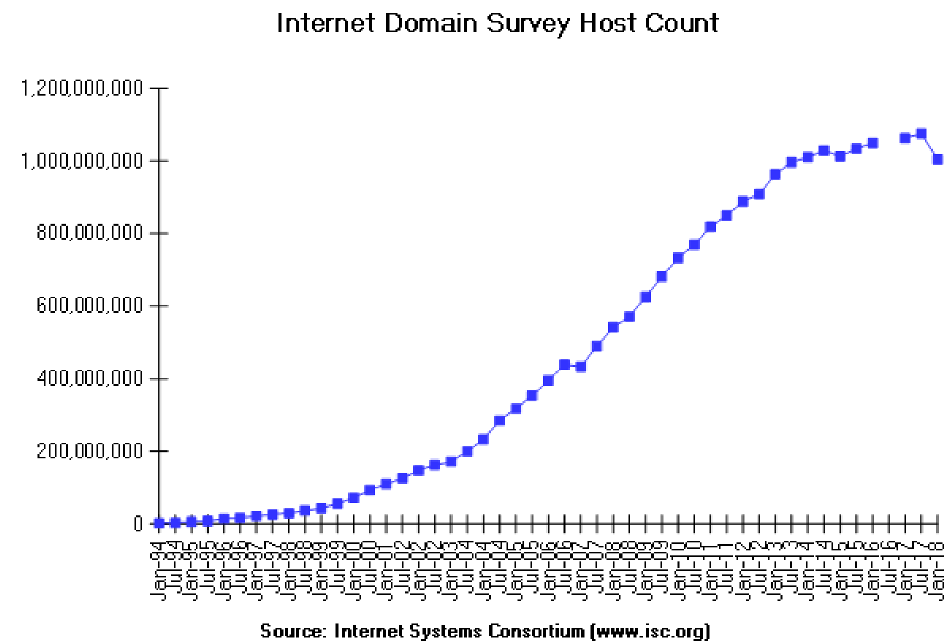


Internet growth: Hosts

(advertised in dns)

Date	Hosts
Dec 1979	188
July 1989	130,000
July 1999	56,218,000
July 2001	125,888,197
July 2003	171,638,297
July 2005	353,284,187
July 2007	489,774,269
July 2009	681,064,561
July 2011	849,869,781
July 2013	996,230,757
July 2015	1,033,836,245
July 2017	1,074,971,748
July 2018	1,015,787,389

Source: Internet Systems Consortium



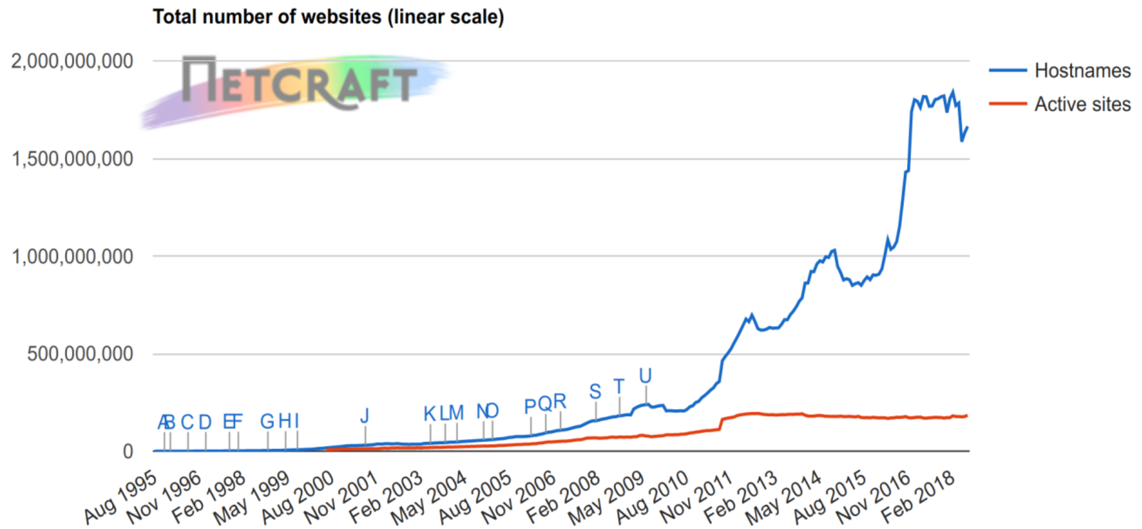
A survey of IPv4 addresses *that have a domain name attached to them*

We have not finished researching the 5% drop from July 2017 to July 2018 but we tentatively believe that it is the result of some ISPs moving their customers to IPv6.

Internet growth: Web servers

Date	Hosts	Web servers	%
July 1993	1,776,000	130	0.008
July 1995	6,642,000	23,500	0.4
July 1997	19,540,000	1,203,096	6
July 1999	56,218,000	6,598,697	12
July 2001	125,888,197	31,299,592	25
July 2003	212,570,000	42,298,371	20
July 2005	353,284,187	67,571,581	19
July 2007	489,774,269	125,626,329	39
July 2009	681,064,561	239,611,111	35
July 2011	849,869,781	357,292,065	42
July 2013	996,230,757	698,823,509	70
July 2015	1,033,836,245	849,602,745	82
July 2017	1,074,971,748	1,767,964,429	164
July 2018	1,015,787,389	1,663,673,364	163

Source: Netcraft web server survey



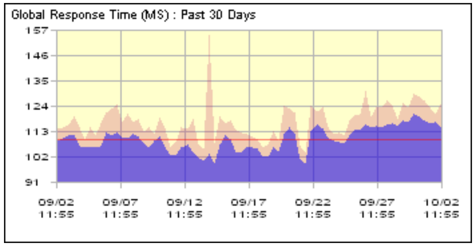
~200M of those sites are active

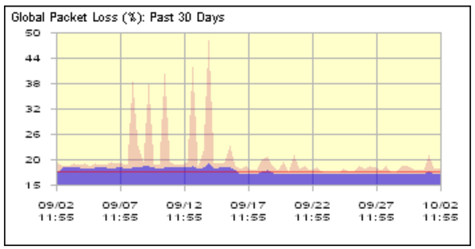
Current estimates: 7B mobile phones, 2.5B smartphones

They generate half of the Internet traffic

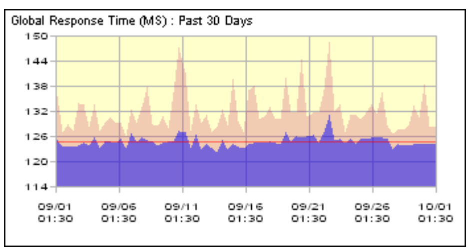
Internet performance

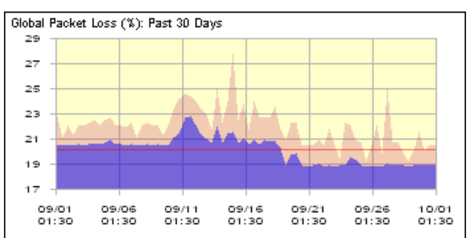
2011



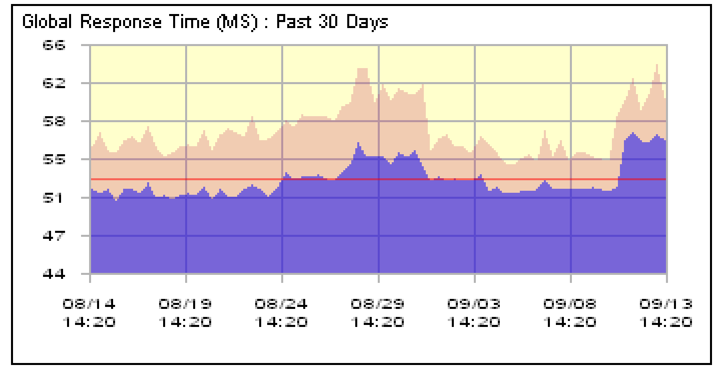


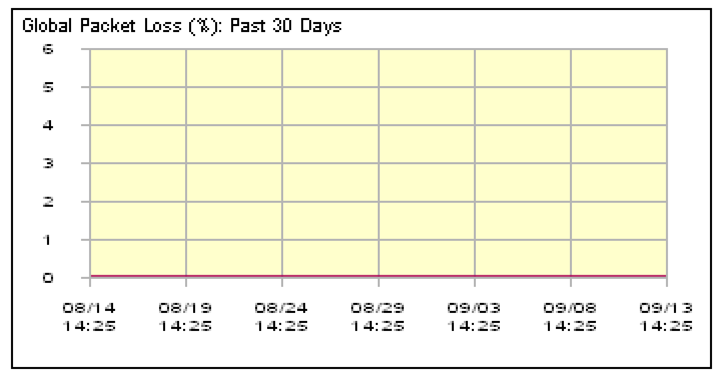
2012





2018





Legenda

red=max

purple=avg

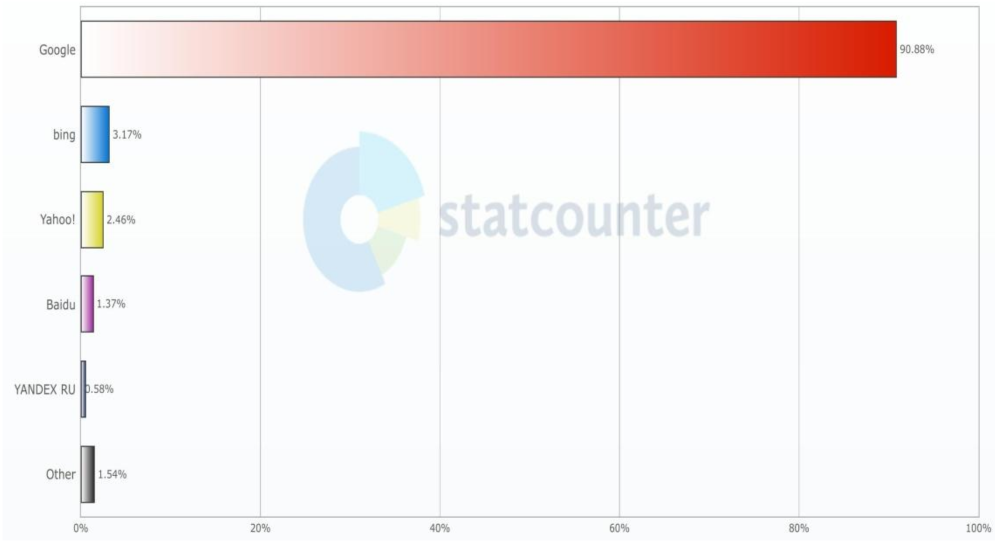
Source: Internet Traffic Report

Top sites and traffic

Site	% of users visiting 1/10/2012
google.com	45.82
facebook.com	43.00
youtube.com	30.35
yahoo.com	20.50
wikipedia.org	12.01
baidu.com	11.74
qq.com	9.30
twitter.com	8.39

Source: Alexa survey

Search engine market share worldwide



Google answers around 5 B searches per day

60k per second on average

http://www.internetlivestats.com/google-search-statistics

Google – An Intranet :-)

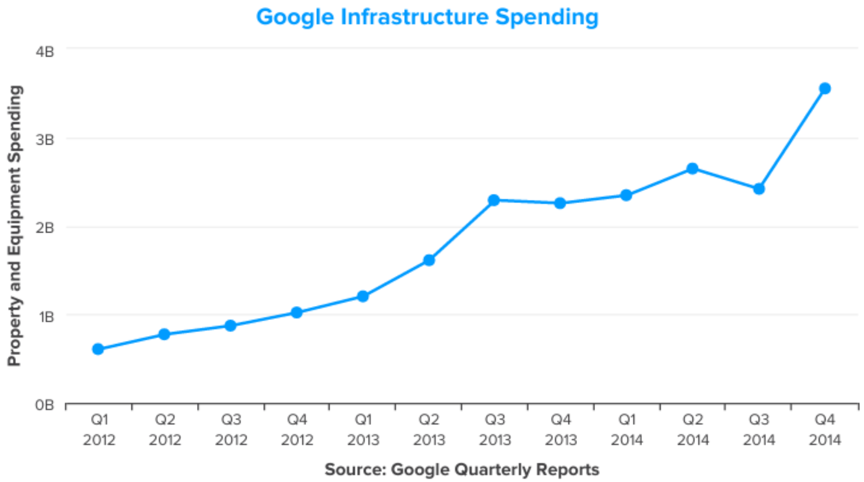
- "Google is technologically a large supercomputer. It's a distributed supercomputer among many data centers doing all sorts of interesting things over fiber optic network that eventually are services available to end-users." - Eric Schmidt, Google CEO, 2007
- "Google runs on hundreds of thousands of servers—by one estimate, in excess of 450,000—racked up in thousands of clusters in dozens of data centers around the world. It recently opened a new center in Atlanta, and is currently building two football-field-sized centers in The Dalles, Ore." – 2006
  - An estimate today says Google runs more than 2,000,000 servers in 36 data centers around the globe

Google (continued)

- "Our view is it's better to have twice as much hardware that's not as reliable than half as much that's more reliable...You have to provide reliability on a software level. If you're running 10,000 machines, something is going to die every day." Jeff Dean, Google fellow, 2008
  - The web site monitoring service Pingdom tracked Google's worldwide network of search sites for a one-year period ending in October 2007, and found that all 32 of Google's worldwide search portals maintained uptime of at least "four nines" – 99.99 percent. The best performer was Google Brazil (google.com.br), with 3 min. of downtime
- "A typical search will require actions from between 700 to 1,000 machines today." - Marissa Mayer, vice president of Google's search products and user experience, 2008
  - Google processes over 40,000 search queries every second on average, which translates to over 3.5 billion searches per day (internetlivestats)
- "Our current generation — Jupiter fabrics — can deliver more than 1 Petabit/sec of total bisection bandwidth. To put this in perspective, such capacity would be enough for 100,000 servers to exchange information at 10Gb/s each, enough to read the entire scanned contents of the Library of Congress in less than 1/10th of a second." Amin Vahdat, Google Fellow and Technical Lead for Networking, 2015

Google (continued)

- In the Dalles, Oregon, Google's site includes three 68,680 square foot data center buildings, a 20,000 square foot administration building, a 16,000 square foot "transient employee dormitory" and an 18,000 square foot facility for cooling towers.
- Google spent more than $3.5 billion on "real estate purchases, production equipment, and data center construction" during the fourth quarter of 2014 and nearly $11 billion for the year.

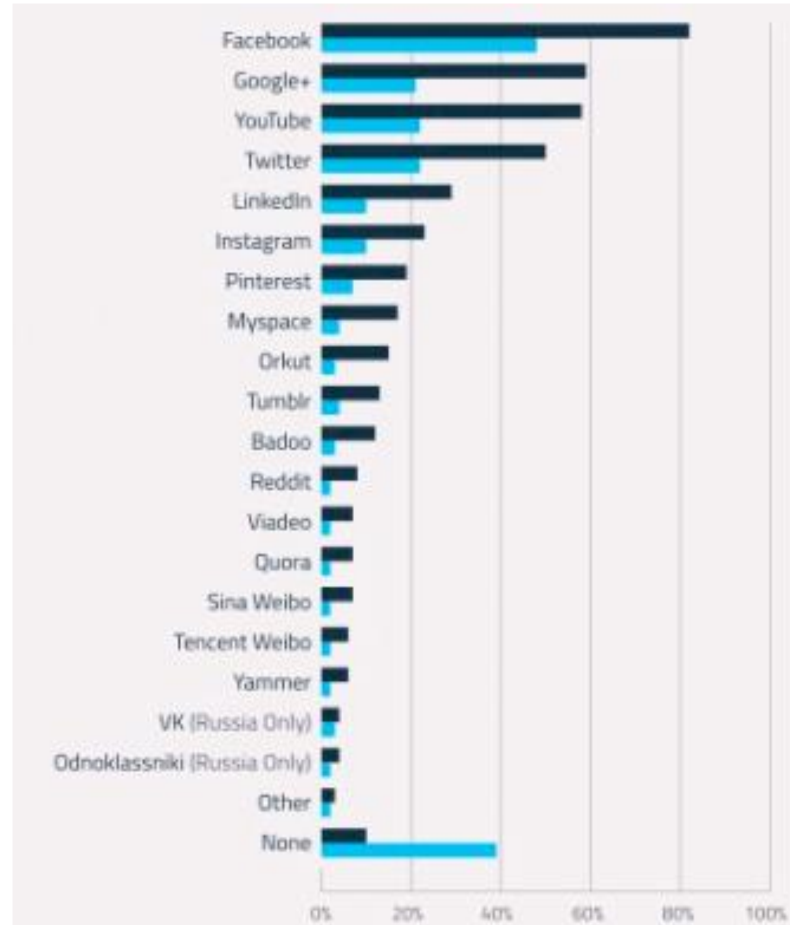


Google (continued)

- Google's criteria when selecting locations for data centers
  - Large volumes of cheap electricity.
  - Green energy. Focuses on renewable power sources.
  - Proximity to rivers and lakes. They use a large amount of water for cooling purposes.
  - Large areas of land. Allows for more privacy and security.
  - The distance to other Google data centers (for fast connections between data centers).
  - Tax incentives.

Google facts

- The search engine (source: comscore qsearch)
  - Amount of monthly Google searches in August 2015 (from the U.S. Only): 11.3 billions
- YouTube (source: youtube statistics)
  - Over 6 billion hours of video are watched each month on YouTube —that's almost an hour for every person on Earth
  - > 100 hours of video are uploaded to YouTube… every minute
  - According to Nielsen, YouTube reaches more US adults ages 18-34 than any cable network



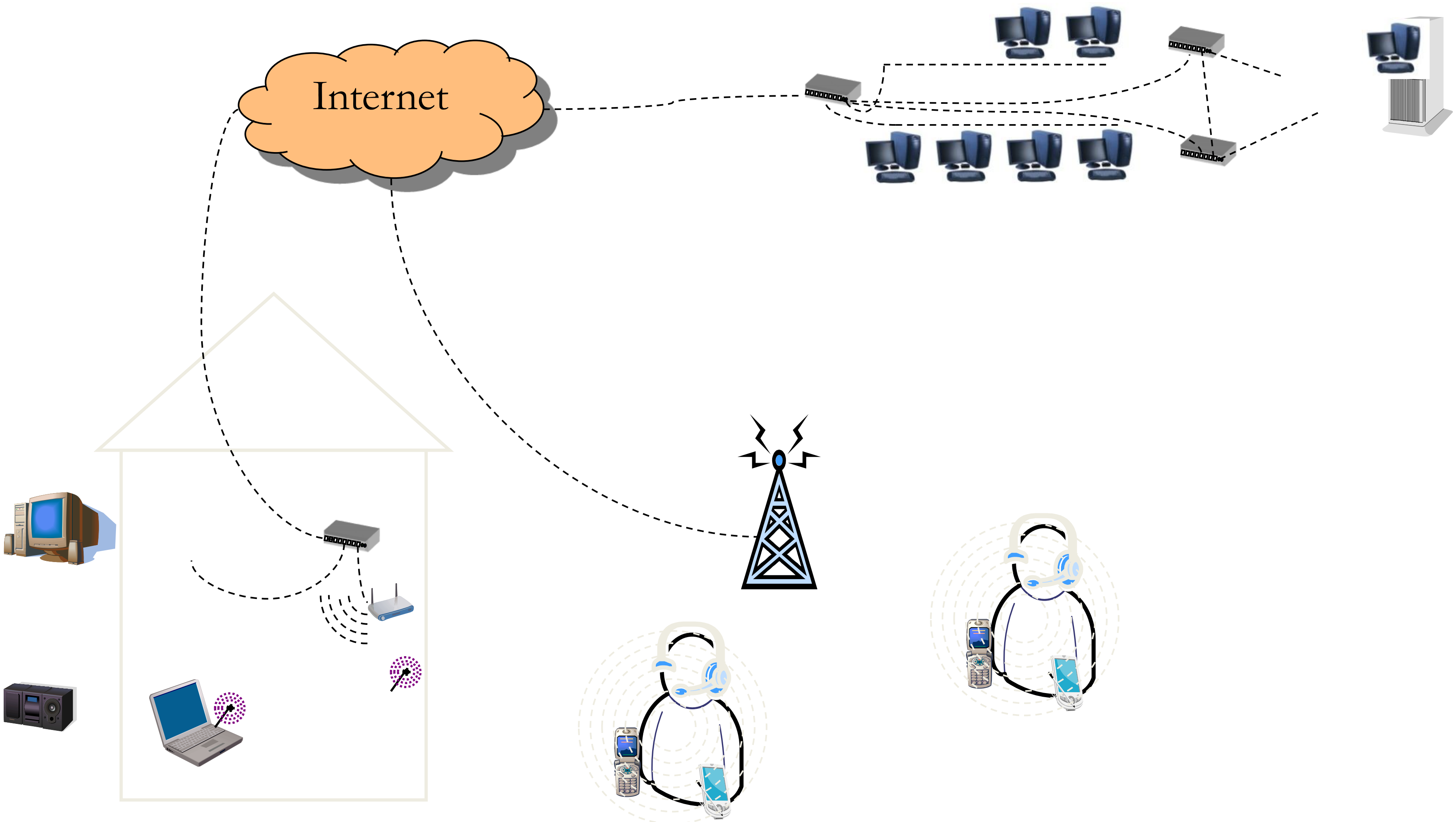
Account ownership vs. Active usage

Source: GlobalWebIndex Social Q1 2014

However…

- Internet and the Web are distributed systems, but not all distributed systems are on (or related to) the Internet and the Web
  - Intranets (banking and corporate systems, scientific computing, server farms, grids, clouds,...)
  - Home networks (home entertainment, multimedia sharing, UPnP)
  - PANs (cellular phones, PDAs, wireless headsets, GPS radio receivers, healthcare monitoring devices)
  - Wireless Sensor Networks
  - …
- Mobility further complicates matters

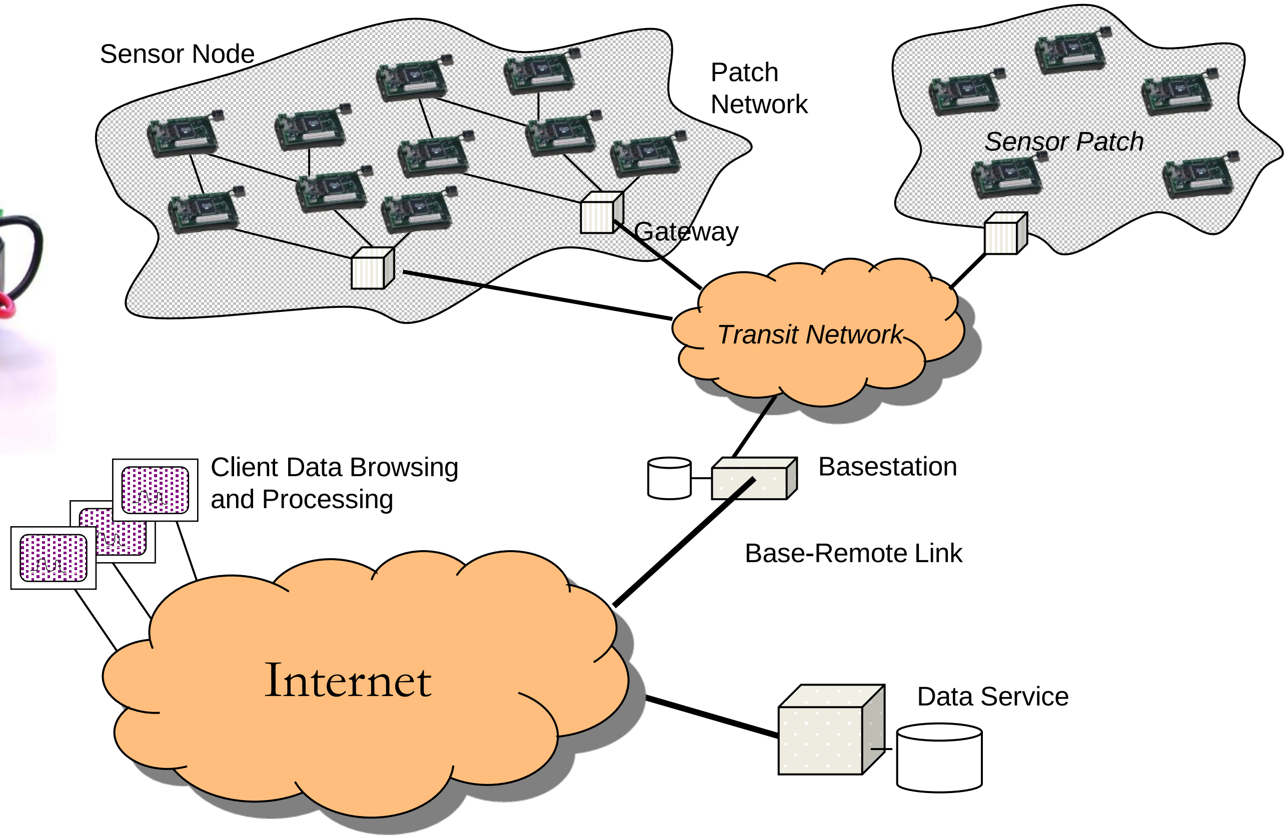
Home and personal networks



Pervasiveness: An Internet of Things

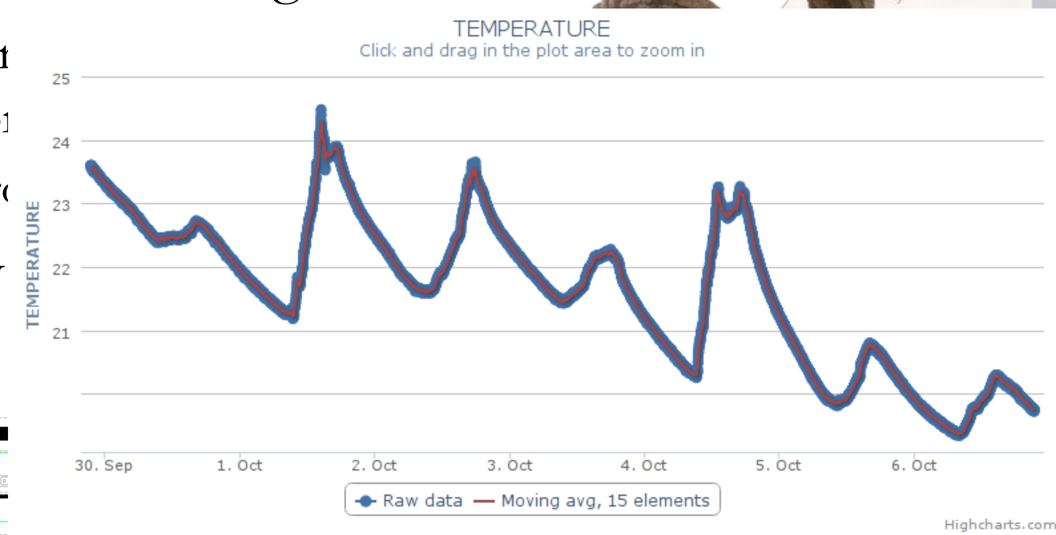
- "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it... only when things disappear in this way are we freed to use them without thinking and so to focus beyond them on new goals." – Mark Weiser, 1991
- "... by embedding short-range mobile transceivers into a wide array of gadgets and everyday items..., a new dimension has been added to the world of information and communication technologies: from anytime, any place connectivity for anyone, we will now have connectivity for anything." – The Internet of Things, ITU Internet Reports 2005.
- "In the future, everything of value will be on the network in one form of another," - John Fowler, Software CTO of Sun Microsystems.
- "When we talk about an Internet of things, it's not just putting RFID tags on some dumb thing so we smart people know where that dumb thing is. It's about embedding intelligence so things become smarter and do more than they were proposed to do." - Nicholas Negroponte, 2005

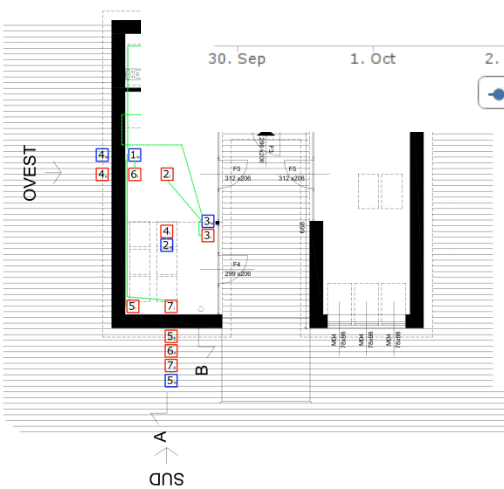
Wireless sensor networks



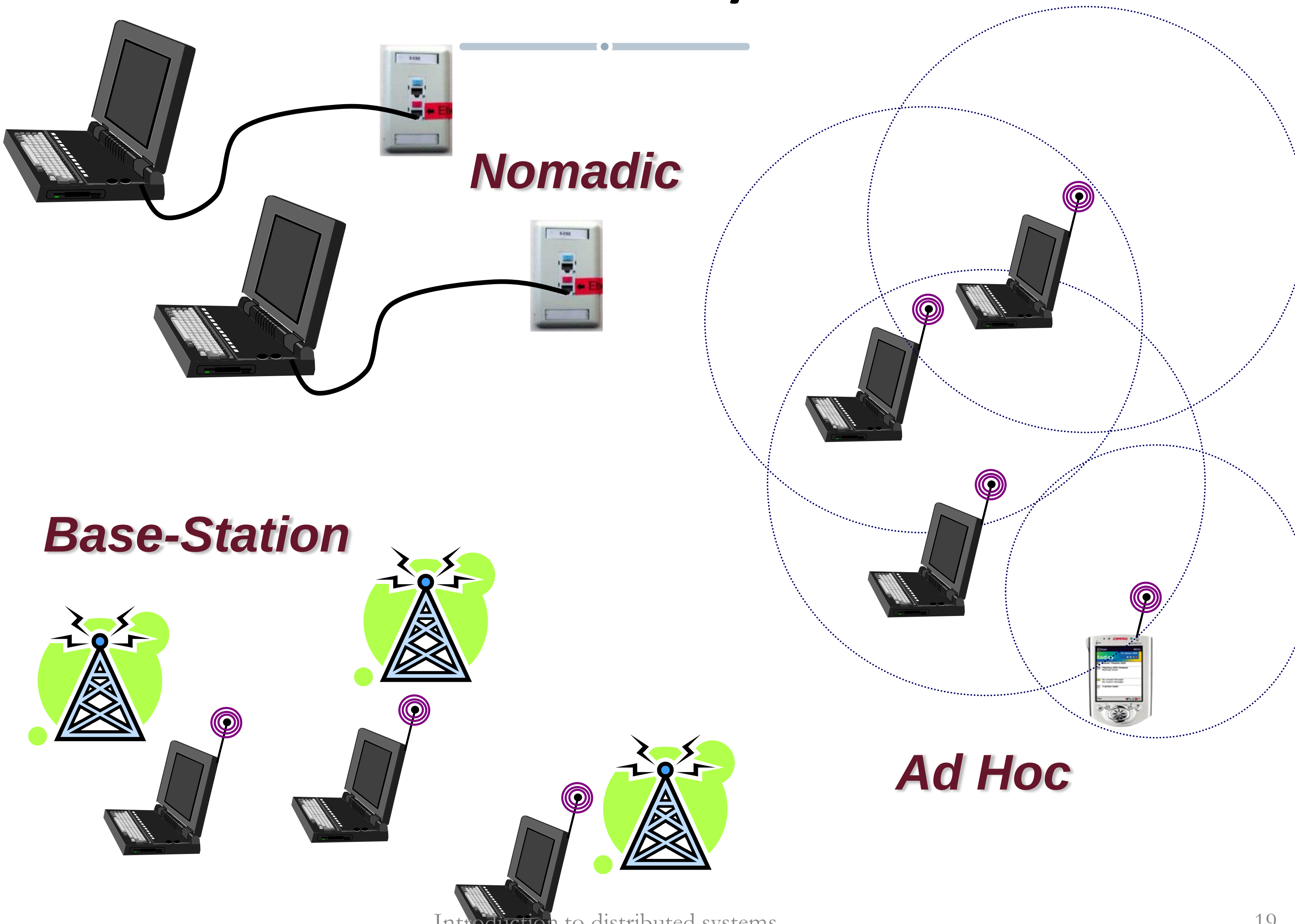
Example application: VeluxLab

- Measuring the behavior of a modern building
  - High pr
    monito
    - Err
- Humidity





Mobility



Why to distribute?

- Economics
  - Originally mainframes consolidated processing
  - Price/performance ratio in favor of distribution
- Incremental growth and load balancing
  - Easier to evolve the system and use its resources uniformly
- Inherent distribution
  - e.g., banks, reservation services, distributed games, CSCW, mobile apps
- Reliability
  - Failure does not bring down the whole system (important for mission critical applications)
- However, distributed systems are considerably more difficult to get right!!!

Ok, but what's special? Why studying something new?

- Distributed systems are more complex than standard (centralized) ones
- Building them is harder
- Building them correct is much harder

The Eight Fallacies of Distributed Computing

Peter Deutsch

Essentially everyone, when they first build a distributed application, makes the following eight assumptions. All prove to be false in the long run and all cause *big* trouble and *painful* learning experiences.

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

Course outline

Distributed Systems

- Concurrent programming in Java (check what you know and what you do not)
- Modeling distributed systems
- Communication models
- Naming
- Synchronization
- Consistency and Replication
- Fault tolerance
- Security
- Peer-to-peer computing
- Simulating distributed systems with OmNet++

Middleware…

- Concurrent programming in Java
- Programming object-oriented systems: RMI
- Programming Service Oriented Systems: REST
- Programming high performance computing systems: OpenMP & MPI
- Actor programming: Akka
- Data streaming: Kafka
- Middleware for WSNs: TinyOS

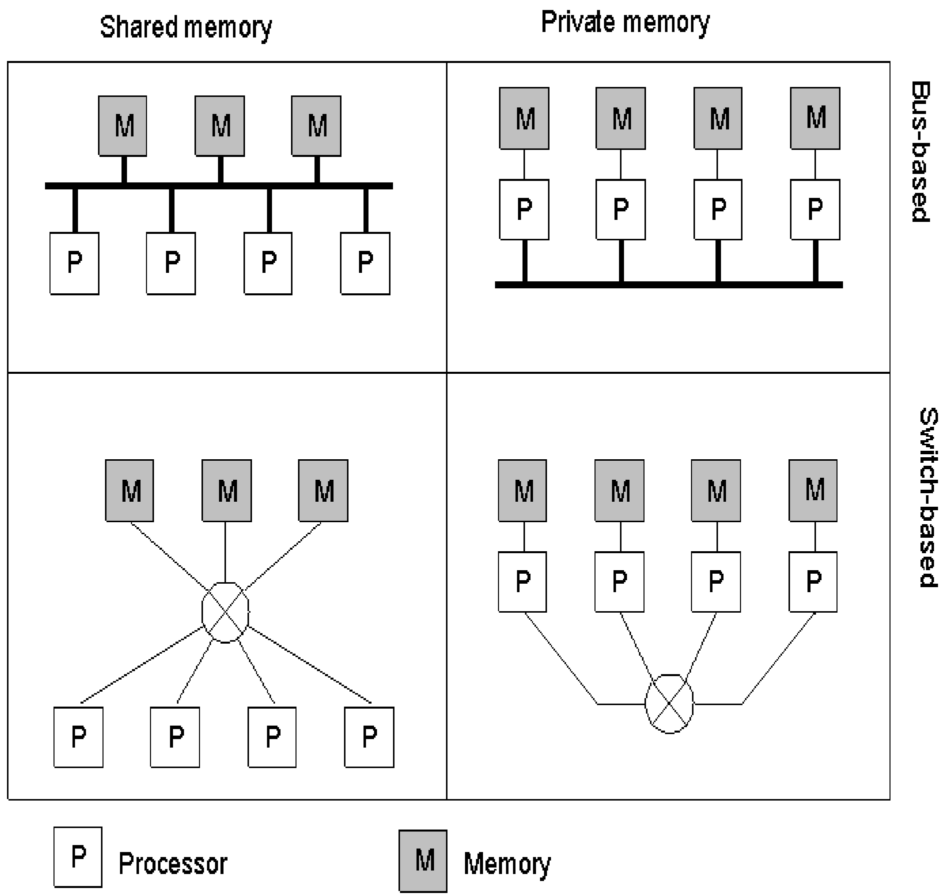
What is a distributed system?

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

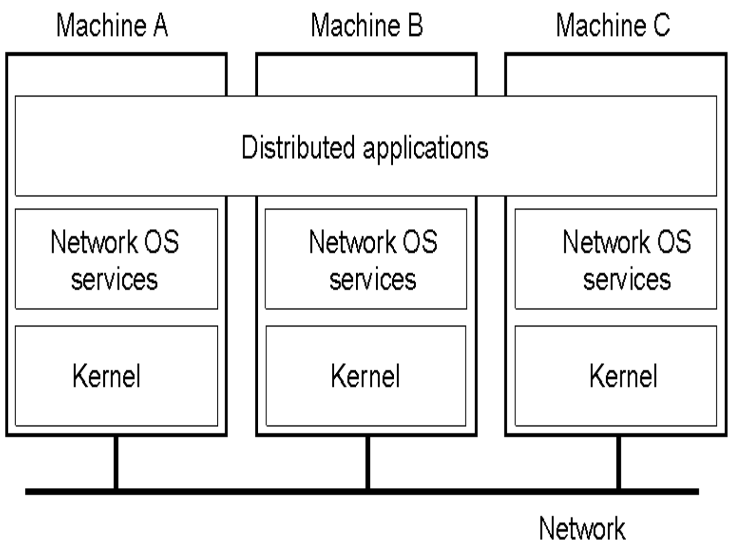
"One in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages"
-- G. Coulouris, J. Dollimore, T. Kindberg

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

Parallel computing vs. distributed computing



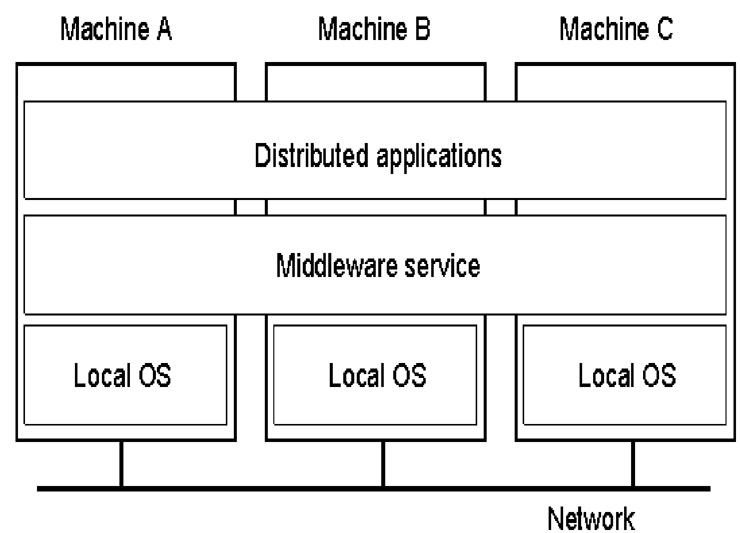
Where middleware fits



Network/OS based distributed system

- Middleware provides horizontal services to help building distributed applications
- It also masks platform differences

Middleware based distributed system



Some defining features

- ***Concurrency***
  In centralized systems, concurrency is a design choice. In distributed systems, it is a fact of life to be dealt with: 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 synched
- ***Independent (and partial) failures***
  Centralized systems typically fail completely. Distributed systems fail only partially, and often due to communication. Each component can fail independently leaving the others still running. Hard/impossible to detect failures. Moreover, recovery is made more complicated by the fact that the application state is itself distributed

Some challenges

- ***Heterogeneity***
  Of hosts, platforms, networks, protocols, languages, …
- ***Openness***
  A distributed system should foster interoperability through standard access rules. Protocols and interfaces play a key role
- ***Security***
  The ease of attaching a node to the system can be exploited maliciously
- ***Scalability***
  Use decentralized algorithms to allow the system to grow with the lowest possible impact on performance
- ***Failure handling***
  Hosts can fail, links are unreliable, the two are usually undistinguishable, global state complicates matters
  Detecting, masking, tolerating, recovery from failures
- ***Concurrency***
  Synchronization without a physical clock and without shared memory
- ***Transparency***
  Hide the most to simplify the life of programmers/users

Heterogeneity

- Networks
  - Even on the Internet, where TCP/IP seems to hide most of the networking differences, the differences at the OSI layers 1 and 2 must be carefully considered in building distributed applications
    - E.g., A wireless ad-hoc scenario is totally different from a wired, fixed scenario
  - Things become more complex when the differences span to higher OSI layers
- Computer hardware
  - Data representation vary with the hw platform and the same holds for several other aspects (e.g., performance)
- Operating systems
  - The API provided to access the basic networking protocols (including TCP/IP) may vary. Other aspects change also with the platform (e.g., the semantics of thread interleaving in Java)
- Programming languages
  - Differen languages represent data structures differently. These differences must be addressed if programs written in different languages must communicate

Openness

- Determines whether a system can be extended and re-implemented in various ways
- To be achieved requires
  - Publishing all the key aspects of the system
    - Protocols
    - Interfaces to services
    - ...
  - Adopting standards as much as possible
  - Take design decisions that could ease interoperability and portability
    - Always choosing the simplest solution
- The Internet case: RFCs and an open standardization body (the IETF)

Security

- Security for the information resources made available and maintained in distributed systems has three components:
  - Confidentiality
    - Protection against disclosure to unauthorized individuals
  - Integrity
    - Protection against alteration or corruption
  - Availability
    - Protection against interfearence with the means to access the resources (DOS attacks)
- Encription is a powerfull mechanisms but several issues still to be solved
  - DOS attacks
  - Security in mobile code applications
  - Secure content-based routing
  - ...

Scalability

- Ability to increase the size of the system in terms of users/resources, geographic span, administrative span
  - Challenge is to reconcile scale with performance
- Typical points of centralization:
  - Services
    - E.g., a single server for all users
  - Data
    - E.g., a single on-line telephone book
  - Algorithms
    - E.g., doing routing based on complete information
- Decentralized algorithms:
  - No machine has complete information about the system state
  - Machines make decisions based only on local information
  - Failure of one machine does not ruin the algorithm
  - There is no implicit assumption that a global clock exists
- Several techniques available
  - Asynchronous communication, caching and replication, epidemic dissemination, hierarchical structuring, …

Failure handling

- Distributed systems fails only partially... This is BAD
- Issues
  - Detecting failures
    - Some failures are easy to detect (e.g., corrupted messages through checksums), while other are impossible (e.g., crash of a remote server)
    - The challenge is to manage the rpesence of failures that cannot be detected for sure but may be suspected
  - Masking failures
    - Some failures that have been detected can be easily masked (e.g., messages can be retransmitted when they fail to arrive)
  - Tolerating failures
    - Components may be designed to tolerate failures of other components, degrading the services they offer, if necessary
  - Recovering from failures
    - By periodically saving the state of a component on a permanent storage it is possible to recover from failures of that component
  - Redundancy
    - It is a common technique to mask/tolerate failures (e.g., using replicated servers or replicated routes)
    - Keeping replicas in a distributed setting is not easy

Concurrency

- Concurrency in distributed systems is a matter of fact
- Access to shared resources (information or services) must be carefully synchronized
- OS and language run-time systems provide several mechanisms to cope with this issue

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
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource may be shared by several competitive users
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk

- 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