Distributed Systems

Olivier Gruber

Full-time Professor
Université Joseph Fourier

Senior Researcher (INRIA et IMAG-LSR)

Acknowledgments

- Prof. Sacha Krakowiak
  - Used his lectures as a canvas
- Reference Book

  Distributed Systems
  Principles and Paradigms

  Second Edition

  Andrew Tanenbaum and Maarten Van Steen

- Research Articles
  - Cited on various slides

This Year Outline

- Course Goals
  - Understand architecture and design trade-offs
  - Master core techniques and essential distributed algorithms
  - Discuss existing systems and frameworks
- Today
  - Background on distributed systems

Distributed Systems

- What are they?
  - Collection of cooperative entities
- Humorous Definition from L. Lamport

> ***A distributed system is one that stops you from getting any work done when a machine you've never heard of crashes.***
>
> ***Leslie Lamport***

  - Highlights the cooperative nature of distributed systems
  - States that failures occur and have consequences

Failure Examples

- September 11th, 2001
  - Most businesses in the towers had only regular data backups
  - No disaster recovery from replicated data
- Space Shuttle
  - Four computers, many missions ended with only one left working...
- Ariane 501
  - June 4th 1996, first launch of Ariane 5 fails: Ariane 5 explodes

http://www.cnes.fr/espace_pro/communiques/cp96/rapport_501/rapport_501_2.html

Distributed Systems

- Examples
  - Networked workstations
    - Only if there is software cooperation
    - Either distributing processing or sharing data
  - The World-Wide Web
    - Where world-wide scalability is *the* challenge
    - Client-server or peer-to-peer
  - Cellular wireless networks (telephony)
    - For voice and data, mobile devices
    - Health monitoring of patients at home or travelling

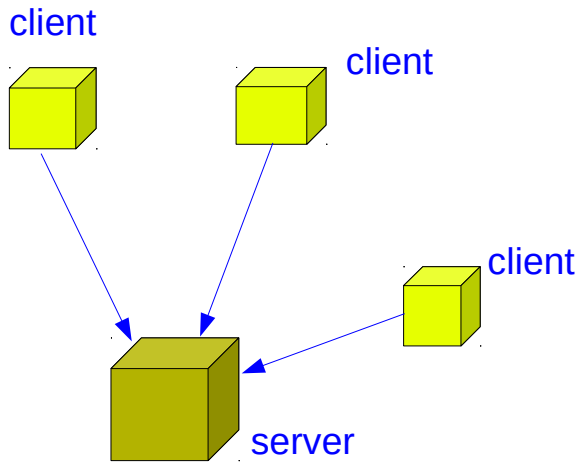
Distributed Systems

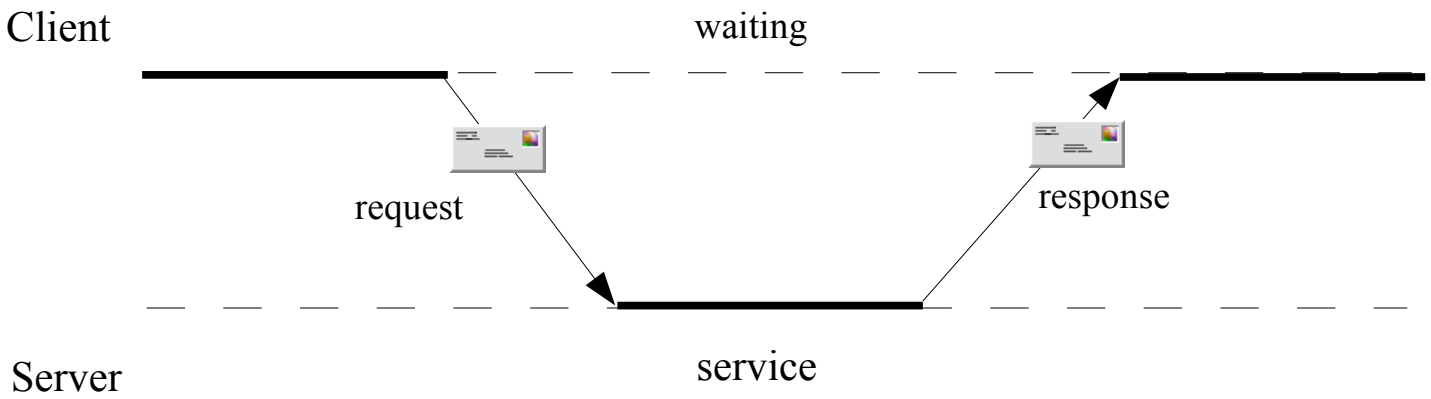
- More Examples
  - Embedded networks
    - In planes or cars
      - BMW Serie 7
      - 4 networks, 70 computers
      - 70% of car failures are computer-related (hardware and software)
    - Sensor networks
  - On-chip networks
    - Distributed systems on chip
    - Soon, more than 64 nodes interconnected on one silicium chip
    - Moving away from consistent shared memory
    - Operating system research goes toward distributed kernels (Barrel-fish)

- Message-Oriented Paradigm
  - A message is a byte stream of known length
  - Asymmetrical relationship
    - Sender (client): build and send a message
    - Receiver (server): wait and receive a message
  - Relies on
    - Naming scheme: names the destination of messages
    - Routing scheme: routes messages to their destination

Distributed Programming

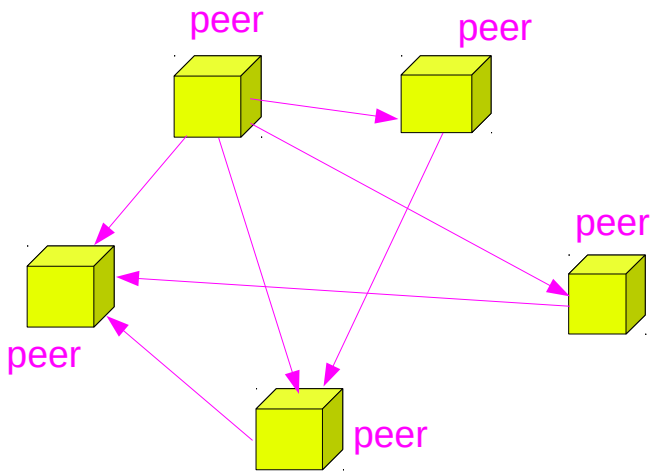
- Client-Server Basics
  - A server provides a centralized decision point
  - But also a single point of failure
  - When the server fails, nothing works

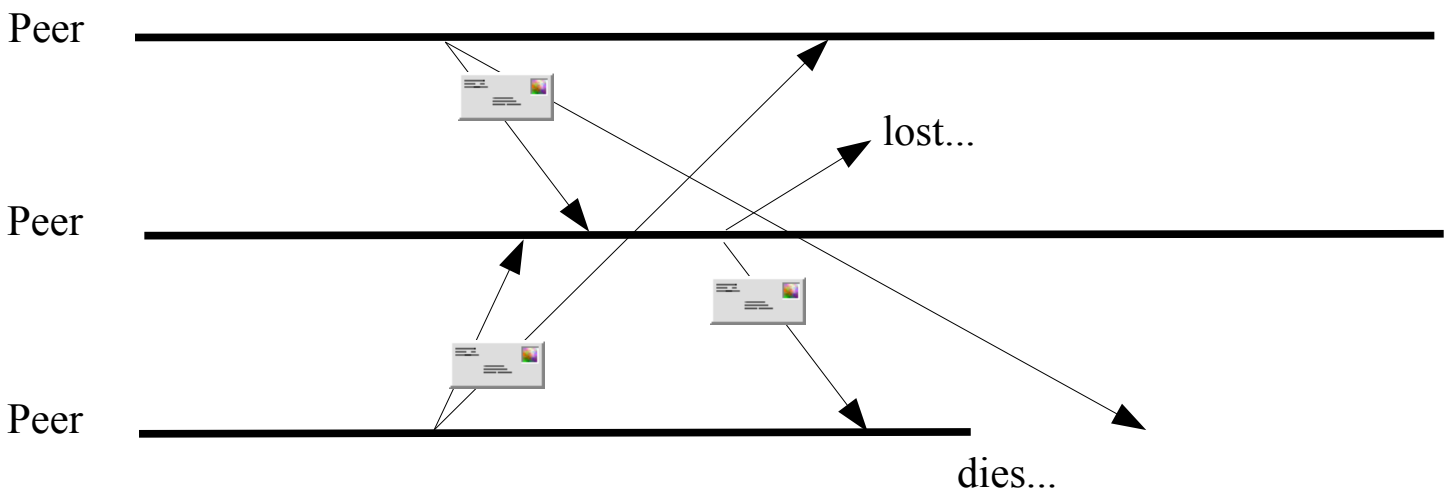




Distributed Programming

- Peer-to-Peer Architecture
  - No more clients or servers, only identical peers
    - No central decision point
    - No single point of failure
  - No global knowledge or synchronization
    - Each peer is a Finite State Machine





Distributed Programming

- Humans or machines?
  - It does not matter, we are talking about cooperative entities
  - Let's discuss distributed systems through human cooperation
- Aliens invaded the Earth
  - One human per cell
    - No one can leave their cell
    - There are no ways to communicate
    - You have a tablet, to be ordered by Aliens
  - After a while, you found a loophole
    - You hacked your tablet to connect to maintenance robots that are passing by
    - You discover that maintenance robots know about prisoner cells
    - Your discover that maintenance robots know how to deliver a message to a cell

Distributed Programming

- How do you organize the community?
  - How do you spread the knowledge?
  - How do you build a naming service?
- If someone does not respond, what does it mean?
- How can you have conversation?
  - Between two people first, then more than two
- How do you tell what time is it?
  - You have a watch, but you know it drifts and it has been a while...
- How do you protect the naming system?
  - Aliens wipe clean tablets at random

Distributed Programming

- How do you elect a watchman?
  - Someone that monitors the health of a group of persons
  - How do watchmen watch other persons?
  - Who watches the watchman?
- How do you reach a consensus?
  - Example: who will try to escape first...

Discussing some Challenges

- Scalability
  - Scale in number of nodes or users
    - From a few nodes to thousands of nodes...
    - The Web... millions of nodes... such as Gnutella with 50 million peers
  - Scale geographically
    - Physical network capabilities are a concern
      - *The speed of light can't be changed...*
    - Limited bandwith and high latency
      - Latency is more of a problem than bandwidth for distributed systems
    - Unreliable:
      - The larger the system, the more probable are faults
      - Loss of messages, partitioning, failed nodes...

Discussing some Challenges

- Failures
  - Failures occur and must be handled
    - No longer possible to ignore faults (the simple life of crash-restart processes)
    - Distributed systems are about cooperation and must survive partial failures
  - Fault-tolerance is expensive
    - More complex algorithms, more messages exchanged, use of stable storage, etc.
  - What failure model?
    - The simpler world of fail-stop entities
      - Detect, repair, reinsert...
    - The real world of byzantine failures
      - No all entities are well-behaved (compromised systems for e.g.)
      - Bugs often introduce byzantine failures

Conclusion

- Course Content
  - Fundamentals on both client-server and peer-to-peer architectures
- Course Goals
  - Prepare you to use and design distributed systems
  - Why? Because they are the very fabric of our computer systems
- Course Philosophy
  - Learn individual techniques/algorithms
  - Apply them to solve concrete problems in practical sessions
  - Master them by combining and evolving them

Conclusion

- Advices
  - This course is for all students
    - Network knowledgeable: Do not assume you already know, you will be proved wrong
    - Not network knowledgeable: Do not assume that this lecture is not for you
  - There is a progression in given lectures
    - Get lost in the beginning and it will be hard on you.
  - Learn as you go, benefit fully from practical sessions
    - The lectures will not be repeated in practical sessions
    - Practical sessions are representative of the final exam
    - The exam will be about evolving and combining core techniques
  - Attend or not attend...
    - Stay in bed at your own risk, slides are not self-contained
      - Take notes, you will need them when you prepare for the exam
    - Do not expect to master this course in two days the week before the exam
      - Especially that software projects are demanding this final year
      - They finish right when you prepare for your exams