Modern Distributed Computing

Theory and Applications

Alberto Marchetti-Spaccamela
Ioannis Chatzigiannakis

Sapienza University of Rome

Lecture 1

Goal of Course

- Present some fundamental aspects of distributed computation
- Study characteristic design approaches of distributed systems and networks
  - communication, coordination, fault-tolerance, locality, parallelism, self-organization, symmetry breaking, synchronization, uncertainty
- Examine essential algorithmic techniques and performance limits
- Accompany the theoretical topics with practical aspects
- Engineering algorithms in open-design, resource constrained IoT platform
  - introducing Arduino, codebender.cc

Basic Definitions

- program – code that dictates the behavior of the system.
- process – an instance of the program.
- message – used for inter process communication.
- packet – part of a message, transmitted by the network.
- protocol – a rigorous description of messages, and rules for message exchanges.
- network – infrastructure that connects processing units.
- distributed system – an application that executes a collection of protocols to coordinate the actions of multiple processes on a communication network towards a common goal.

Distributed Systems & Broad definitions

"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable".
– Leslie Lamport, Thu, 28 May 87 12:23:29 PDT

Distributed Systems & Broad definitions

"Distributed systems need radically different software than centralized systems do.". – Andrew S. Tanenbaum

Distributed Systems & Broad definitions

"A distributed system consists of a collection of autonomous computer linked by a computer network and equipped with distributed system software. Distributed system software enables computers to coordinate their activities and to share the resources of the system – hardware, software, and data – " – Coulouris et al., 1994

Distributed Systems & Broad definitions

"A system of multiple autonomous processing elements cooperating in a common purpose or to achieve a common goal". – Burns & Willings

Distributed Systems & Broad definitions

"Most computer software today runs in distributed systems, where the interactive presentation, application business processing, and data resources reside in looselycoupled computing nodes and service tiers connected together by networks." – Buschmann et al., 2007

Computer Networks – Distributed Systems

- Courses related to Computer Networks:
  - Focus on message/packet transmissions.
  - Do not examine how packets are being handled/processed.
- In this block course
  - We assume a mechanism for sending/receiving messages.
  - Focus on the message properties.
  - Design systems that use these messages.

Operating Systems – Distributed Systems

- Courses related to Operating Systems:
  - Resources are reliable.
  - Resources are used without examining failures.
  - Failures are local and straight forward error handlers are used (e.g., in MINIX, device drivers are restarted to recover from failures).
  - A common global clock is used for process synchronization.
- In this block course:
  - Communication over computer network is not always reliable.
  - We may not know when/if a failure has occurred.
  - We do not have access to a common global clock – how are process synchronized?

Parallel Systems – Distributed Systems

- In courses related to Parallel Systems & Concurrency:
  - Multiple processors are installed in the same processing unit.
  - Communication between processors is very fast & efficient.
  - A common global clock is used for process synchronization.
  - Processing units are high quality – they rarely fail.
- In this block course:
  - Communication over computer network is not always fast & efficient.
  - We do not have access to a common global clock – how are process synchronized?
  - Processing units do not always achieve high reliability.

Framework

- Our goal is to design systems for accomplishing specific identified purpose (e.g., leader election, mutual exclusion etc.)
  - We model the operating conditions of the system.
  - We design a distributed algorithm (protocol).
- We examine the behavior of the system (e.g., correctness, performance, resilience to failures)
  - In the presence of failures.
  - Under predetermined mode of execution (e.g., synchronous execution).
- We always assume that the processing units cooperate to achieve the common goal.
  - Any other case is considered to be unnatural – we consider it as an erroneous case.

A more realistic framework

- What if the processes do not adhere to the common goal?
  - e.g., a process does not wish to be elected a the leader, or a process does not wish to wait for her turn in order to use the critical resource
- When we wish to consider/study the behavior of "rational" players, such cases are very common during the execution of the system.
  - How do we model them? How do we study them?
- Algorithmic Game Theory
  - Based on the assumption that players are rational and may have egoistic goals.
  - The 2008 and 2006 Nobel prizes for Economics where awarded in the field of Game Theory.

Design for Correctness

- Current Software & Network infrastructures already pervade our everyday life.
- We rely deeply on software-based infrastructure and when it fails to function, there can be serious side effects.
- We become aware of our dependence only when the infrastructure is down.
- Future systems will be constantly present monitoring all aspects of our life.
- Clearly, society expect future systems to be dependable, robust and resilient to sudden environmental changes.
- It is important to understand the impact of our systems when we design & implement them.

Dependence to Infrastructure

- A common theme we hear in many conversations is concern for the fragility or brittleness of our networked-and software-driven world.
- Most of us do not lie awake worried that the power will go out.
- When the power does go out, we suddenly become aware of the finiteness of battery power or the huge role that electricity plays in our daily lives.
- Mobile phones went out during Hurricane Sandy because the cell towers and base stations ran out of power either because of battery failure or because the back-up generators could not be supplied with fuel or could not run because they were underwater.

Robustness and Resilience

"I therefore propose a development project to make our system more robust." – Leslie Lamport, Thu, 28 May 87 12:23:29 PDT

What's a Robot ?

"I believe it would be a contribution to our society to encourage deeper thinking about what we in the computing world produce, the tools we use to produce them, the resilience and reliability that these products exhibit and the risks that they may introduce."
– Vinton G. Cerf, ACM President, Jan 2013

What's a Robot ?

"For decades now, Peter Neumann has labored in this space, documenting and researching the nature of risk and how it manifests in the software world. We would all do well to emulate his lead and to think whether it is possible that the three or four laws of robotics might motivate our own aspirations as creators in the endless universe of software and communications."
– Vinton G. Cerf, ACM President, Jan 2013

The Three Laws

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

The Three Laws

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

Later on the zeroth law was added to precede the others:

0. A robot may not harm humanity, or, by inaction, allow humanity to come to harm.

The Three Laws

Gaia may not harm life or, through inaction, allow life to come to harm.

Safety, Validation & Robustness

Edmund M. Clarke and E. Allen Emerson, Joseph Sifakis, laureates of the 2007 Turing Award, for their roles in developing model checking into a highly effective verification technology, widely adopted in the hardware and software industries.

Barbara Liskov, laureate of the 2008 Turing Award, for her contributions to practical and theoretical foundations of programming language and system design, especially related to data abstraction, fault tolerance, and distributed computing.

Hierarchical Analysis of Systems Performance

A fundamental method for studying the performance of a system is the top-down approach

- Initially we abstract all technical details and study the system at high level (i.e., bird's eye view)
- Then, we look into specific modes of operations and investigate the most important parameters that affect performance.
- Step - by step, we introduce additional levels – until we end up to our final system, operating in the actual conditions

This approach leads to good results for organizing and analyzing a broad range of systems

Contemporary Systems

Hierarchical, centralized, top-down approaches have allowed us to design very good contemporary systems

- e.g., database management systems, mobile telephony networks

However our always-connected world is becoming more complex

- We should not ignore the fact that many contemporary systems have a totally different structure.
- e.g., the stability and effectiveness of contemporary politico-economic models relies on decentralized, distributed mechanisms that are independent and self-regulated
- The Internet is another example of a similar approach, at a techno-social level.
- How to efficiently organized extremely huge collections of unstructured or structured data ?

Limitations of Top-down approach

Studying a distributed system from a theoretical perspective contributes to a basic level of understanding its behavior and rigorously defining its performance bounds.

However, it entails certain pitfalls:

- Abstracting certain technical details may lead to totally unrealistic / non-implementable solutions.
- Measuring complexity does not take into account the so-called "hidden" constants.
- A "poor" complexity solution may be very efficient in almost all practical case.
- It is very hard (if not impossible) to analyze the performance of a system using theoretical tools.

Performance evaluation by Experimentation

A different approach is the implementation of the system and its evaluation using practical means:

- The implementation may use an experimental framework – e.g., simulator, testbed facilities, . . .
- The performance study is done using well-defined evaluation scenaria
- Measure performance of the "actual" performance.
- Immediate validation of the applicability of a solution in existing technologies.
- Results can be deployed to devices in real-world deployments.

Necessity for Experimental Driven Research

- Identified at European level and started to attract attention in the latest very-competitive FP7 funding framework.
- The Future Internet Research and Experimentation Initiative (FIRE) is addressing this need.
- A multidisciplinary research environment for investigating and experimentally validating highly innovative and revolutionary ideas for new networking and service paradigms.
- In the period 2008-2012 a total of 115 million Euro have been invested in the context of FIRE.
- Research and Experimentation will also be a visible component of Horizon 2020, with an overall objective to build a world-class, versatile, integrated and sustainable experimental infrastructure.

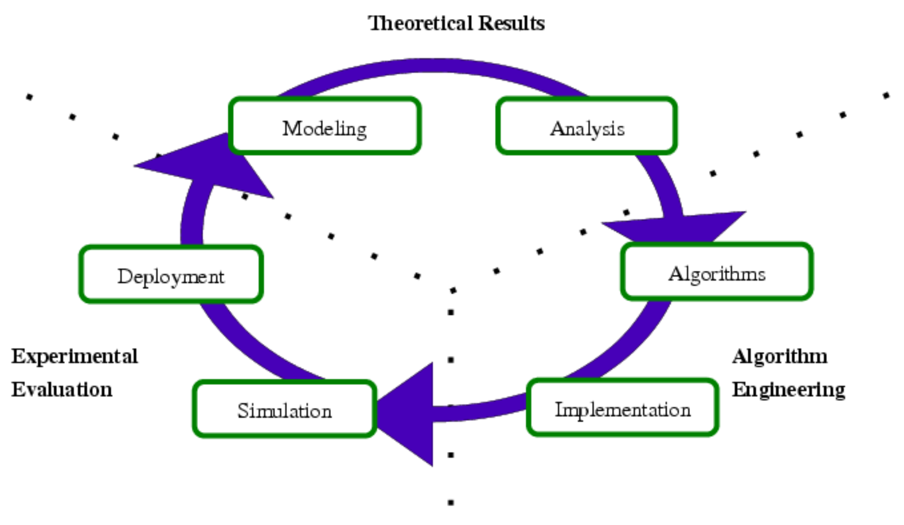
Dual Approach

Each approach has certain benefits and handicaps:

- A theoretical approach allows to develop solutions that are correct by proof, efficient . . . may not be applicable (or very hard) in current technologies.
- A practical approach immediately deals with all technological issues and provides effective solutions . . . may not result in innovative solutions that are efficient in large scale systems.

We need to be both efficient and effective.

Theoretical – Practical Approach Cycle



Necessity of Dual Approach

- Surprisingly, the need for association between theory and practice has been identified long before the computer science era.
- Philosophers of the antiquity, already, state that the notion of effectiveness requires two components: design an efficient prototype, an *ideal version* that is used to plan the goal; then, *apply* this plan in practice.
- According to Plato, $\nu o \eta \sigma \iota \varsigma$ (cognition) "captures optimal" plans, and $\theta \epsilon \lambda \eta \sigma \iota \varsigma$ (goodwill) is required to apply the ideal plans in reality.
- This is defined by Aristotle as $\phi \rho o \nu \eta \sigma \iota \nu$, the process of associating the ideal with its application, thus reducing the gap between these two approaches.

Foundation and Earth

"How do you decide what is injurious, or not injurious, to humanity as a whole?"

"Precisely. In theory, the Zeroth Law was the answer to our problems. In practice, we could never decide. A human being is a concrete object. Injury to a person can be estimated and judged. Humanity is an abstraction."

Part 1: Static Synchronous Networks

1. Synchronous Message-passing Model, Definitions
2. Anonymous Leader Election, Impossibility Results
3. Symmetry Breaking Algorithms, Randomization
4. Leader Election Algorithms
5. Broadcast, Convergecast
6. Lower Bounds

Part 2: Failures

1. Link failures, Node failures, Impossibility Results
2. Agreement, Commit
3. Byzantine Failures
4. Failures in Asynchronous Systems
5. Failure Detectors

Part 3: Static Asynchronous Networks

1. I/O Automata Model, Definitions, Lower bounds
2. Distributed Data Structures
3. Time, Clocks and Ordering of Events
4. Synchronizers
5. Global Predicates, Termination Detection

Part 4: Stabilization

1. Definitions
2. Mutual Exclusion
3. Breadth First Search
4. Power Supply Technique

Part 5: Population Protocols

1. Population Protocol Model, Rendezvous, Fairness
2. Passively Mobile Machines
3. Mediated Population Protocols
4. Self Awareness

Practical Aspects

1. Arduino Platform and simple electronic boards
2. Cloud based Development Environment: codebender.cc
3. Algorithms for Ring networks and General Networks
4. Asynchrony & Synchronization
5. Failures & Consensus