GC course: 72020 (Distributed Operating Systems), Spring 2020

1. Course motivation

Distributed computing and networking can be viewed both as an art and science pervading through our lives in many ways. It forms the foundational pillar of various technology applications we have seen (and continue to see): such as the Internet, Emails, online transactions, Social networks, data centers & clouds, and multi-player gaming. The present course (82100) covers the core operating systems topics to be understood when building a distributed system of networked computers to support such applications.

Consider, for instance, a seemingly simple online purchase triggers a workflow process that gets executed at multiple computers to process information: such as product description, store inventory, credit card charging, scheduling shipment, and the like. The purchase involves a coordinated execution of these processes by message-passing and communications (without relying on a shared memory between them).
The underlying support mechanisms for these processes can be viewed as provided by a distributed operating system that glues the various computers together through software abstractions and primitives realized on top of network hardware. The key attributes of such a distributed system as expected by applications are:

(i) Fault-tolerance & reliability (e.g., completion of purchase transaction
    despite computer failures);

(ii) Good quality-of-service (QoS) - say, low latency of a purchase transaction;

(iii) High performance - say, a high transactional throughput for lower costs.

The aim of GC course 82100 is to expose students to the computer

science issues
in the design of correctly functioning networked systems while meeting the
attributes (i)-(iii). The technical challenge is to meet this goal despite
the absence of a physically shared memory between various sub-systems.

2. Course Synopsis

We shall begin with two fundamental concepts: Lamport's notion of logical time and
global snapshots of network state. We shall examine different communication models
(synchronous, partially-synchronous and asynchronous), and how they impact the design
of distributed algorithms. We shall cover primitives for broadcast and gossip based
communications between a distributed system of nodes, and provide case studies of
how they are employed in cloud servers and data centers today. The course will also
cover replication methods to achieve fault-tolerance and assured QoS of systems,
in the presence of malicious/benign errors randomly occurring at sub-system levels
(e.g., majority voting among multiple components to make correct decisions in the
presence of liars). Students will also be assigned a programming project that involves
developing & testing a distributed algorithm on a network test-bed (under artificially
injected failures).

3. Prior knowledge desirable

System programming on UNIX-like systems, JAVA and C-like languages, inter-process
communications in operating systems, usage of networked systems.

4. Course Topics to be covered (total of 40 lecture hours: each hour is 50-mt session)

4.1 Introductory portion (2 hours)

Brief coverage of the basic computer network and distributed system concepts and terminologies

4.2 Distributed algorithm structure and design (10 hours)

Failure models: crash failures, omission failures, byzantine failures;

Programming level abstractions: fail-stop behaviors, failure detectors;

Communication models: synchronous, partially synchronous, and asynchronous message-passing;

Causality tracking: logical clocks, vector clocks;

Distributed global snapshots: Chandy-Lamportís algorithm; system monitoring; global predicates.

4.3 Distributed programming primitives (15 hours)

Service-level specification & verification of networked systems;

Axioms of distributed consensus and agreement;

Distributed coordination: consistency, dead-reckoning (e.g., multi-player games, shared white-board, multi-robot teams);

Broadcast primitives: atomic, causal, and ordered multicasts.

4.4 Distributed system control techniques (8 hours)

Replication and fault-tolerance; majority voting, primary-backup methods;

Rollback & replay algorithms and correctness conditions;

4.5 Case studies of distributed systems (5 hours)

DOD's Resilient Clouds (BBN-Raytheon), IBM Websphere, Cloud/network auditing tools,
HPOpenView based system management tools.

5. Grading

Programming project 35%, Midterm exam 25%, Final exam 40% (exams are take-home).

6. Reference materials

Distributed Systems: Sape Mullender, Addison-Wesley (ACM Press);

Distributed Operating Systems: M. Singhal, and N. Shivratri, McGraw-Hill Publ.

Conference proceedings on Network Management Systems, Real-time Systems,
Cloud Services and Computing, Monitoring and Debugging of Distributed Real-time Systems ó IEEE-CS and ACM publications.

Distributed Networks journals: IEEE Transactions on Network Services and management,
IEEE/ACM Transactions on Networking, Springer journals on Clouds and Network Services

7. Instructor profile:

Kaliappa Ravindran is a Professor of Computer Science at the City College of CUNY
and a doctoral faculty member at the CUNY Graduate Center. His research areas
are in distributed computing over clouds, service-level network management,
fault-tolerance & replication algorithms, autonomic networked systems,
software-defined networks, and cyber-physical software systems. He has
published over 150 papers under the broad umbrella of distributed computing
and networking, keeping abreast with the technological and scientific advancements

in the field. His research has been supported by many federal government agencies
and industries (such as Air Force, Navy, NSF, CISCO, and Genetral Motors).