Goals:
- Give an overview of CS 142: course structure & administration
- Define distributed systems and discuss why they are hard to get right
- Provide some real-world examples (and what can go wrong)

Reading:
- P. Sivilotti, *Introduction to Distributed Algorithms*, Chapters 1 and 2
  - Course notes, available from course home page
  - Chapter 1 should be review; will be covered in lecture on Fri
Course Administration

CS 142 - Distributed Computing

Course syllabus
- Instructors (lecturers, TAs)
- Lectures, problem solving sessions
- Office hours, Q&A forum (Piazza)
- Course outline
- Homework policy (+ grace period)
- Grading scheme, collaboration policy
- Course text and references
- Course load: keep track of hours
- Course ombuds: send e-mail to Richard by Tue evening to volunteer

http://www.cds.caltech.edu/cs142
Distributed Systems

What is a distributed system? Why study them?

- Most of the computing systems that you encounter are distributed: finance, the Internet of Things, social media.
- Concurrent systems deal with systems in which multiple agents operate concurrently.
- Concurrent computing systems can use shared memory or message passing or both. Most of this course deals with message-passing systems.
- Material for concurrent systems could stretch over three terms. We only have one. So we focus on fundamentals.
Example Problems Involving Distributed Computing

**Leader Election**
- Distributed set of processes elect a given process to serve as leader

**Two Phase Commit**
- Collection of processes participate in a database transaction
- Each process has to decide whether transaction should be committed or aborted
- Agreement: no two processes should decide on different values
- Validity: if any process aborts, all must abort
- Weak termination: if there are no failures, all processes eventually decide
- Strong termination: all non-faulty processes eventually decide [requires 3 phase commit]

**Block Chain**
- Open, distributed ledger that records transactions between two parties efficiently and in a verifiable and permanent way
Safety-Critical Systems: Commercial Aircraft

10^9 hours of flight =
~100K years
(30K flights/day in US ⇒
~10^8 flight hrs/year total)

<table>
<thead>
<tr>
<th>Hazard Classification</th>
<th>Development Assurance Level</th>
<th>Maximum Probability per Flight Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catastrophic</td>
<td>A</td>
<td>10^{-9}</td>
</tr>
<tr>
<td>Hazardous</td>
<td>B</td>
<td>10^{-7}</td>
</tr>
<tr>
<td>Major</td>
<td>C</td>
<td>10^{-5}</td>
</tr>
<tr>
<td>Minor</td>
<td>D</td>
<td>--</td>
</tr>
<tr>
<td>No Effect</td>
<td>E</td>
<td>--</td>
</tr>
</tbody>
</table>

DO-178C / ED-12C
Software Considerations in Airborne Systems and Equipment Certification

Latest Revision: 01/05/2012
Prepared by: RTCA SC-205
EUROCAE WG-12

Formal methods supplement
Model-based development supplement
Object-oriented technologies supplement
Official Word from Boeing: ZA002 787 Dreamliner fire and smoke details

By David Parker Brown, on November 10th, 2010 at 3:46 pm

For the last day there are been bits and pieces of information coming from Boeing, inside sources and different media outlets on ZA002’s sudden landing due to reported smoke in the cabin. Boeing has just released an official statement putting some of the rumors to rest and explaining what they know of ZA002’s recent emergency landing in Laredo, TX.

Boeing confirms that ZA002 did lose primary electrical power that was related to an on board electrical fire. Due to the loss, the Ram Air Turbine (RAT), which provides backup power (photo of RAT from ZA003) was deployed and allowed the flight crew to land safely. The pilots had complete control of ZA002 during the entire incident.

After their initial inspection, it appears that a power control panel in the rear of the electronics bay will need to be replaced. They are checking the surrounding areas for any additional damage. At this time, the cause of the fire is still being investigated and might take a few days until we have more answers.

Ram Air Turbine (RAT) deployed and allows safe landing

Loss of primary electrical power => cockpit goes “dark”
Environmental Control System

Aircraft Vehicle Management Systems

How do we design software-controlled systems of systems to insure safe operation across all operating conditions (w/ failures)?
Lost Wingman Protocol Verification

Temporal logic specification

\[ \text{mode} = \text{lost} \implies \text{stable}(d(x_l, x_f) > d_{\text{sep}}) \]

- “Lost mode leads to the distance between the aircraft always being larger than \( d_{\text{sep}} \)”

Protocol specification in CCL

- Use guarded commands to implement finite state automaton
- Allows reasoning about controlled performance using semi-automated theorem proving
- Relies on Lyapunov certificates (invariants, metrics) to provide information about controlled system
CCL Specification for Lost Wingman

CCL-based protocol

- High speed link used to communicate state information between aircraft
- Low speed link used to confirm status
- Update timers based on when we last sent/received data
- Change modes if data is not received within expected period (plus delay)

Program $P_{comm}$

Initial: $T_s = t_0 \land T \in (T_s, T_s + \Delta T]$

Commands:

- $c_{data} \equiv t > T \land data.on :$
  
  $T' \in (T_s + \Delta T, T_s + \Delta T + \tau_d) \land T'_s = T_s + \Delta T$

- $c_{msg,1} \equiv in(1) : msg'_1 = recv(1)$
- $c_{msg,2} \equiv in(2) : msg'_2 = recv(2)$

Program $T_{sm}$

Initial: $m_2 = n$

Commands:

- $c_{lost} \equiv m_2 \in \{n, f\} \land t - T > \Delta T + \tau_d :$
  
  $m'_2 = l_1 \land t'_lost = t$
  
  $\land send(1, "lost")$

- $c_{found} \equiv m_2 \in \{l_1, l_2\} \land t - T < \Delta T + \tau_d :$
  
  $m'_2 = f$

- $c_{lost2} \equiv m_2 = l_1 \land msg_m = "lost" :$
  
  $m'_2 = l_2 \land v'_ref = msg_2.v$
  
  $\land \psi'_{ref} = msg_2.\psi$

...
Flight Test Results (June 2004)

- Event 1: communications lost; T-33 executes tight turn; signals lost comms (slow link)
- Event 2: F-15 confirms communication lost message received
- Event 3: communications restored; T-33 requests rejoin (granted)
- Event 4: rejoin confirmed; return to normal operation
Summary: Introduction to Distributed Computing

Main takeaway points

- Distributed systems (and hence distributed algorithms) are everywhere
- Debugging concurrent systems is much harder than debugging sequential programs
- For safety- (or business-) critical systems, formal proofs of correctness are key

In this class, we will learn to

- Model a distributed algorithm and how it executes
- Write specifications for correctness (safety, liveness)
- Prove that distributed algorithms are correct
Plans for the Week

Monday (25 Sep)
• Introductory lecture, course logistics
• Homework set #1 will be posted to web page tonight (due 4 Oct)

Wednesday (27 Sep)
• Models of execution
• Finite state automata, guarded command programs

Friday (29 Sep)
• Review of predicate calculus ($\wedge \lor \neg \equiv \neq$), quantification ($Qi : r(i) : t(i)$)
• Course ombuds announced

Homework #1 is due 4 Oct (Wed)

Online resources for the course:
• Course web page: http://www.cds.caltech.edu/cs142
• Piazza forum (Q&A): https://piazza.com/caltech/fall2017/cs142/home
• Moodle (for submitting HW): https://courses.caltech.edu → CS 142 (FA 2017)