Caltech

CS/IDS 142: Lecture 1.1
Introduction to Distributed Computing

Richard M. Murray

30 September 2019

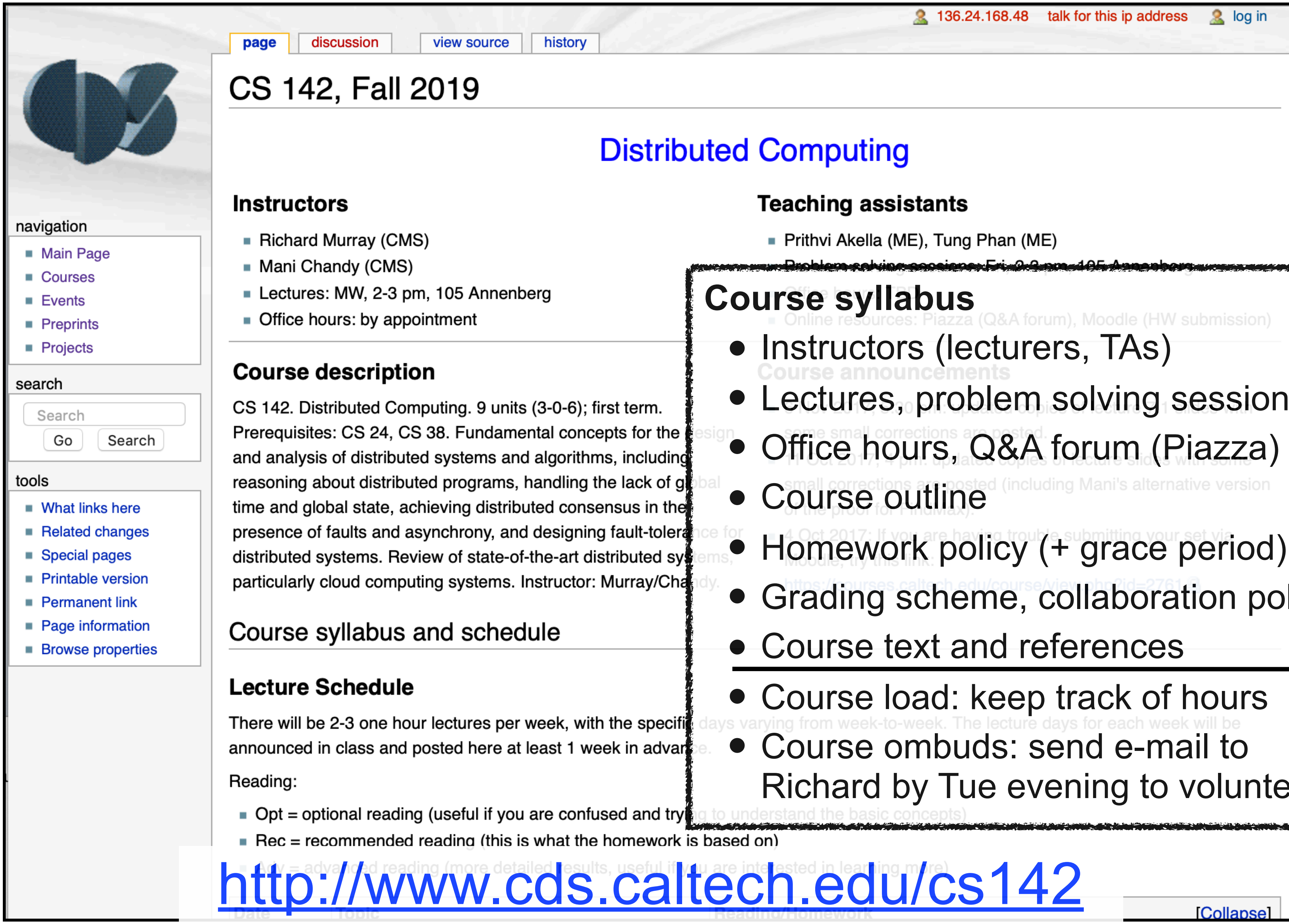
Goals:

- Give an overview of CS/IDS 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



136.24.168.48 talk for this ip address log in

page | discussion | view source | history

navigation
- Main Page
- Courses
- Events
- Preprints
- Projects

search

Go | Search

tools
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link
- Page information
- Browse properties

CS 142, Fall 2019

Distributed Computing

Instructors
- Richard Murray (CMS)
- Mani Chandy (CMS)
- Lectures: MW, 2-3 pm, 105 Annenberg
- Office hours: by appointment

Teaching assistants
- Prithvi Akella (ME), Tung Phan (ME)

Course description

CS 142. Distributed Computing. 9 units (3-0-6); first term. Prerequisites: CS 24, CS 38. Fundamental concepts for the design and analysis of distributed systems and algorithms, including reasoning about distributed programs, handling the lack of global time and global state, achieving distributed consensus in the presence of faults and asynchrony, and designing fault-tolerance for distributed systems. Review of state-of-the-art distributed systems, particularly cloud computing systems. Instructor: Murray/Chandy.

Course syllabus and schedule

Lecture Schedule

There will be 2-3 one hour lectures per week, with the specific days varying from week-to-week. The lecture days for each week will be announced in class and posted here at least 1 week in advance.

Reading:
- Opt = optional reading (useful if you are confused and trying to understand the basic concepts)
- Rec = recommended reading (this is what the homework is based on)

[Collapse]

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

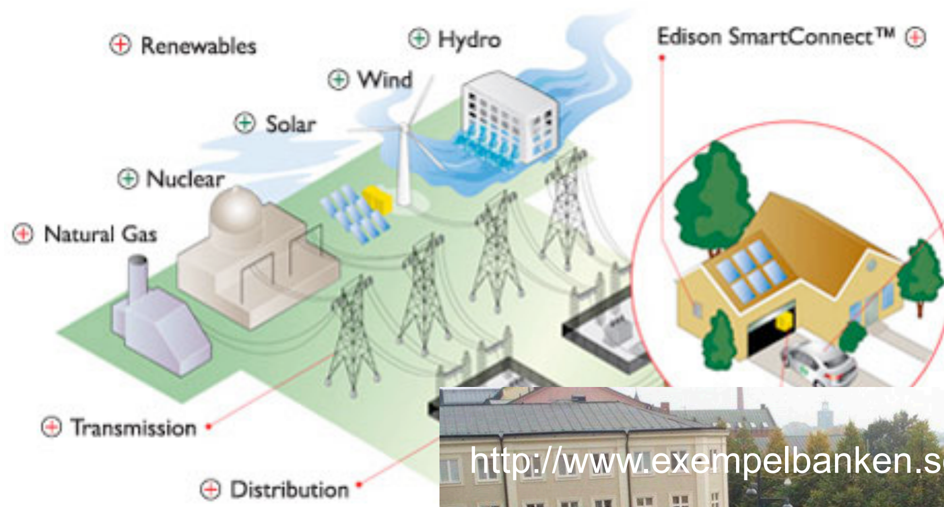
- Course load: keep track of hours
- Course ombuds: send e-mail to Richard by Tue evening to volunteer

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

Distributed Systems

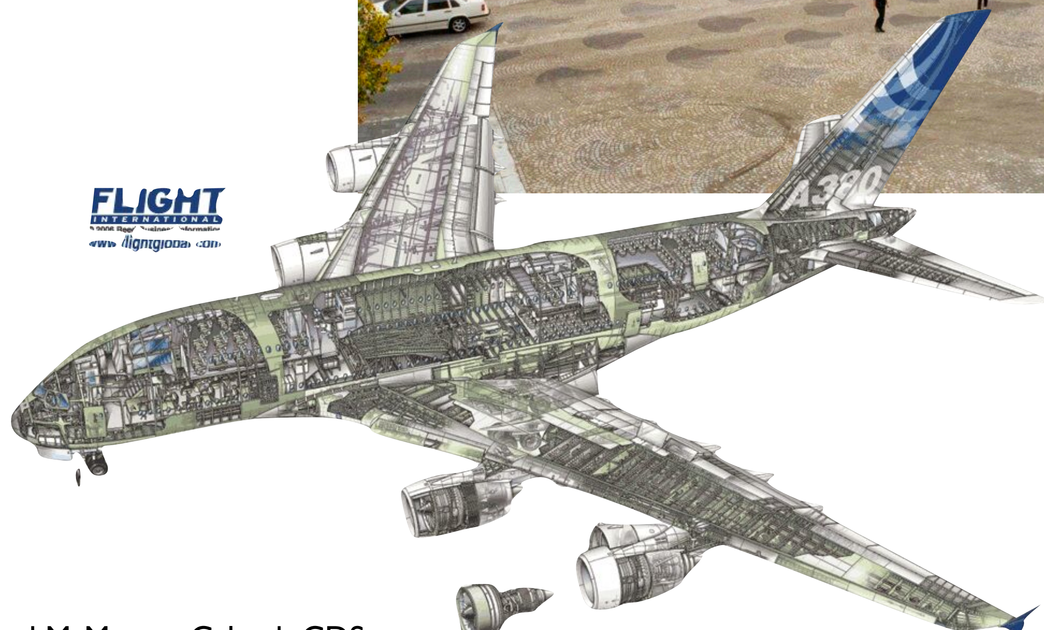
What is a distributed system? Why study them?

- 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.



Southern California Edison (sce.com)

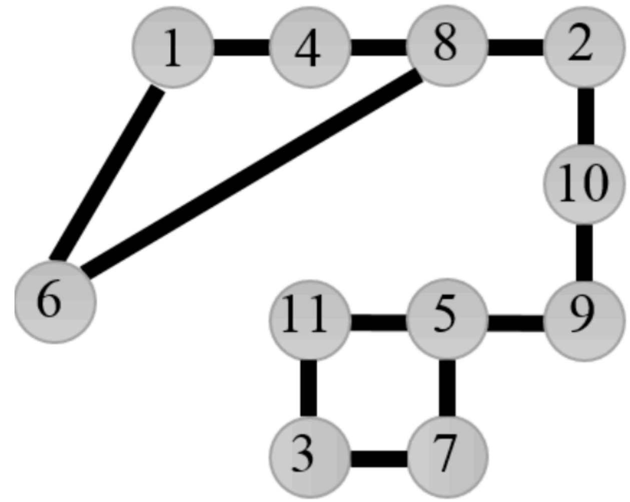
http://www.exempelbanken.se/examples/347



Example Problems Involving Distributed Computing

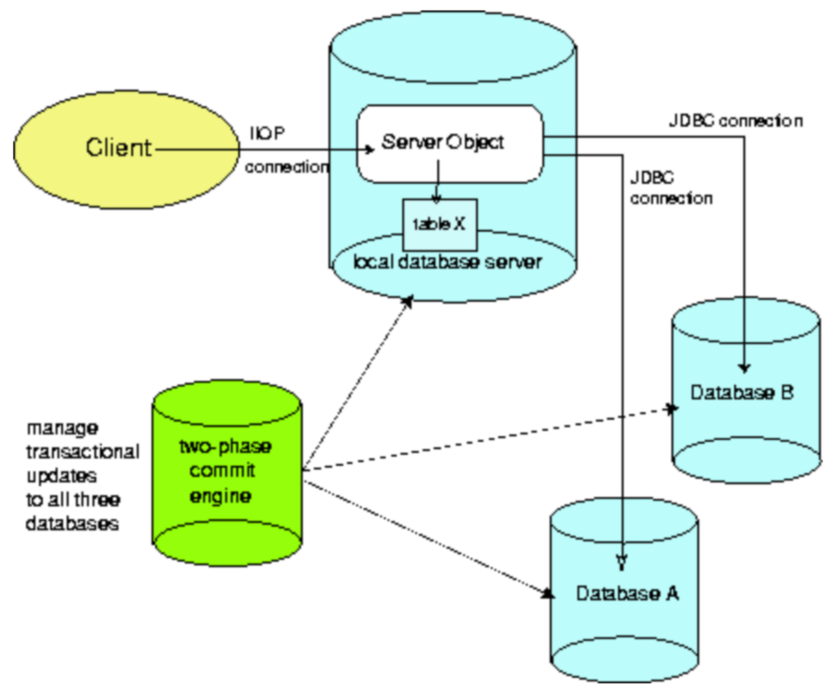
Leader Election

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



Two/Three Phase Commit

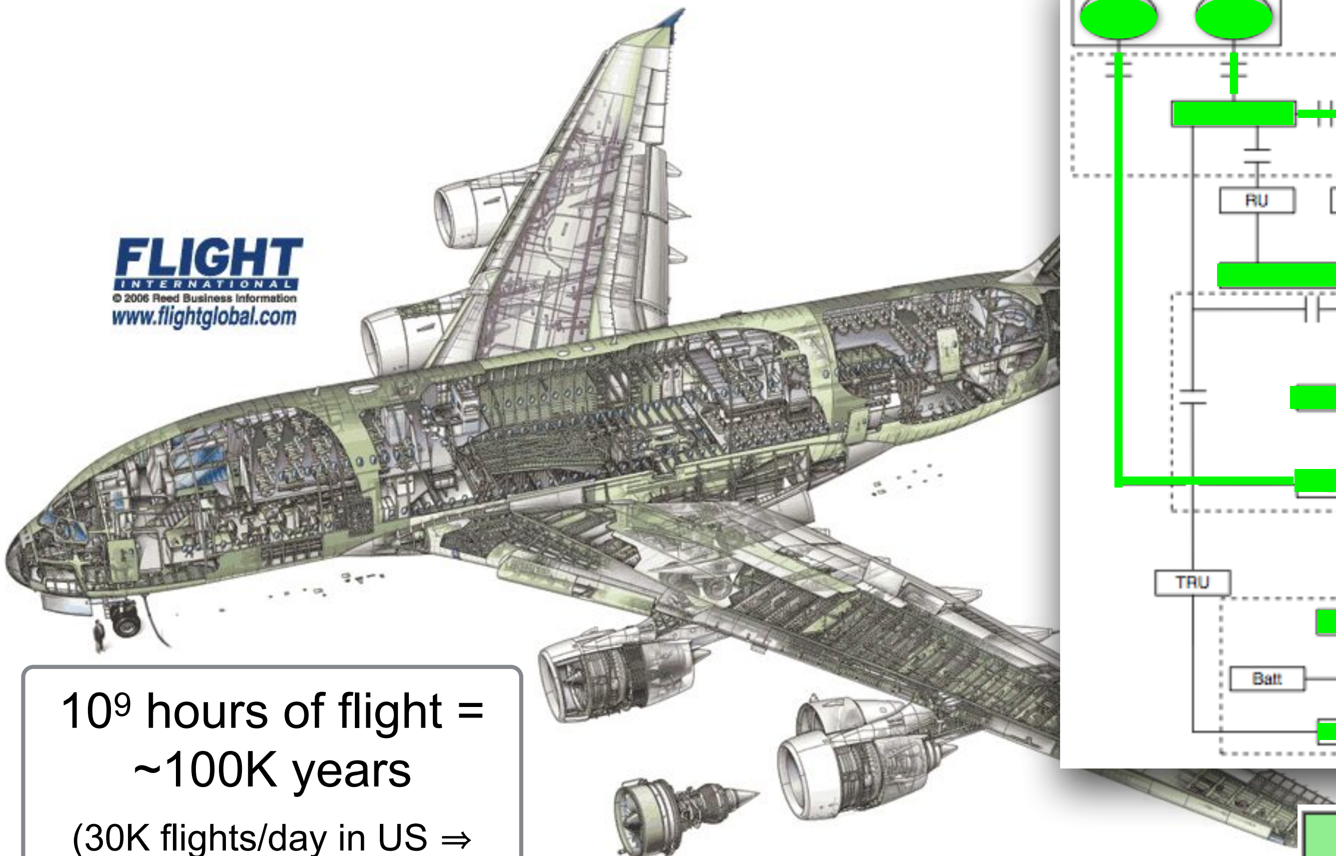
- Collection of processes participate in a database transaction
- Each process has to decide whether transaction should be committed or aborted
- Agreement: no two processes should decide on different values
- Validity: if any process aborts, all must abort
- Weak termination: if there are no failures,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

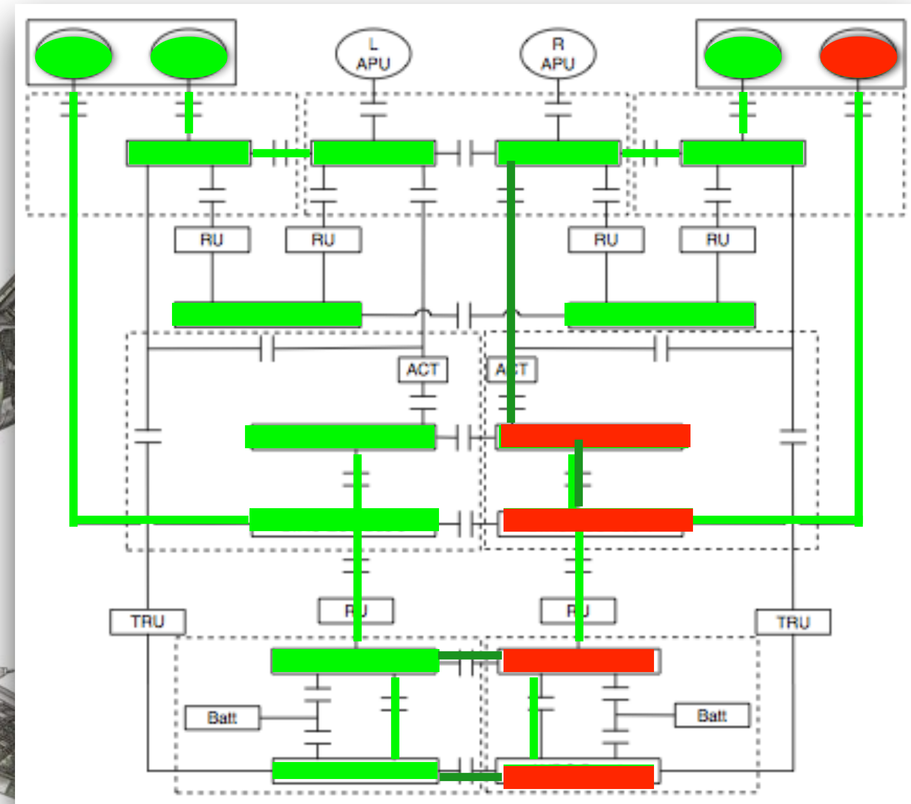


Block Chain

- 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)

Hazard Classification	Development Assurance Level	Maximum Probability per Flight Hour
Catastrophic	A	10^{-9}
Hazardous	B	10^{-7}
Major	C	10^{-5}
Minor	D	--
No Effect	E	--

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

What Goes Wrong: ZA002, Nov 2010

Official Word from Boeing: ZA002 787 Dreamliner fire and smoke details

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

Boeing 787 Dreamliner ZA002 at Paine Field on January 27, 2010 before its first flight.

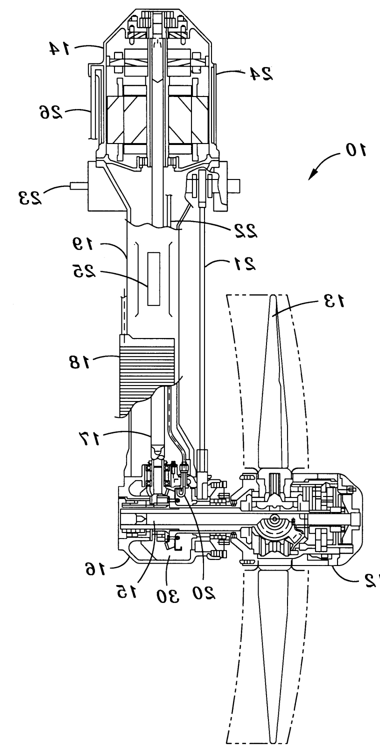
For the last day there are been bits and pieces of information coming from Boeing, inside sources and different media outlets on ZA002's sudden landing due to reported smoke in the cabin. Boeing has just released an official statement putting some of the rumors to rest and explaining what they know of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

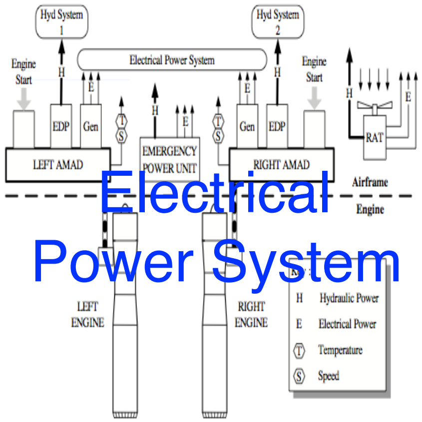
After their initial inspection, it appears that a power control panel in the rear of the electronics bay will need to be replaced. They are checking the surrounding areas for any additional damages. At this time, the cause of the fire is still being investigated and might take a few days until we have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

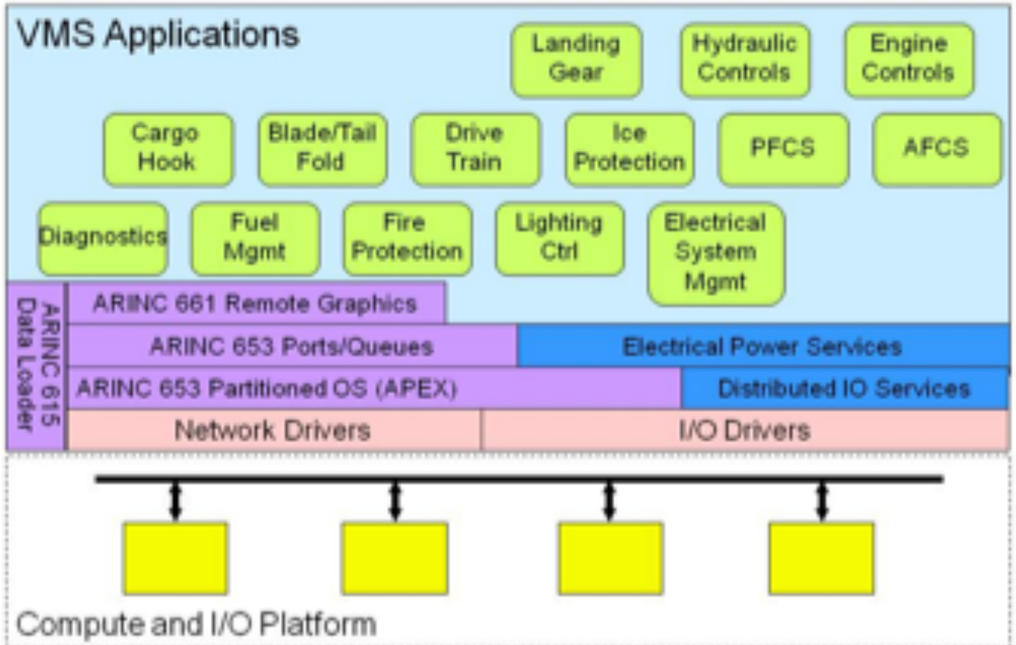
Loss of primary electrical power => cockpit goes "dark"

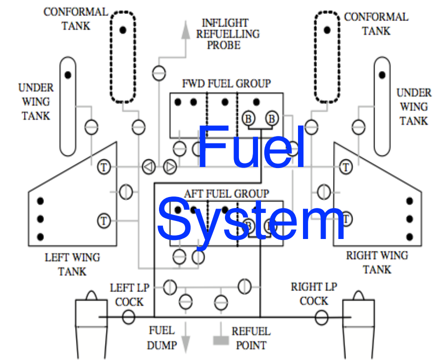
Ram Air Turbine (RAT) deployed and allows safe landing

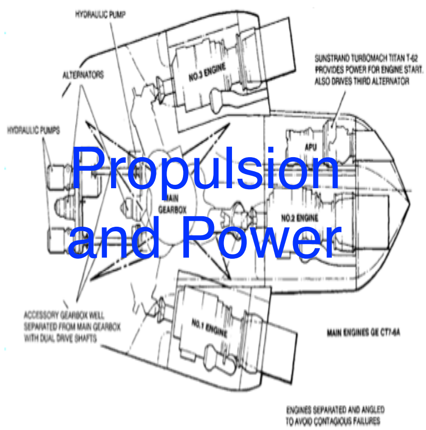


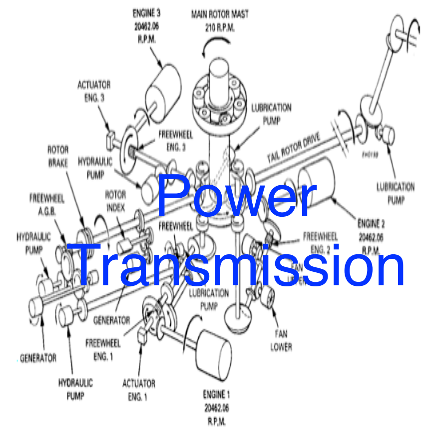
Aircraft Vehicle Management Systems



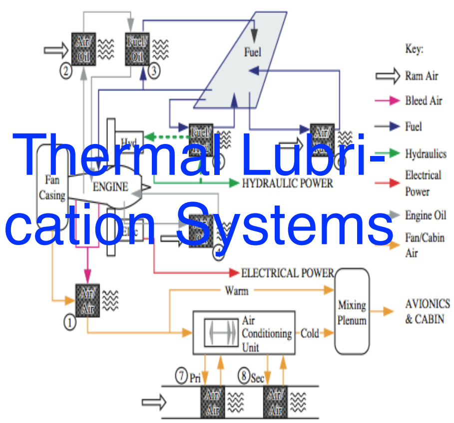


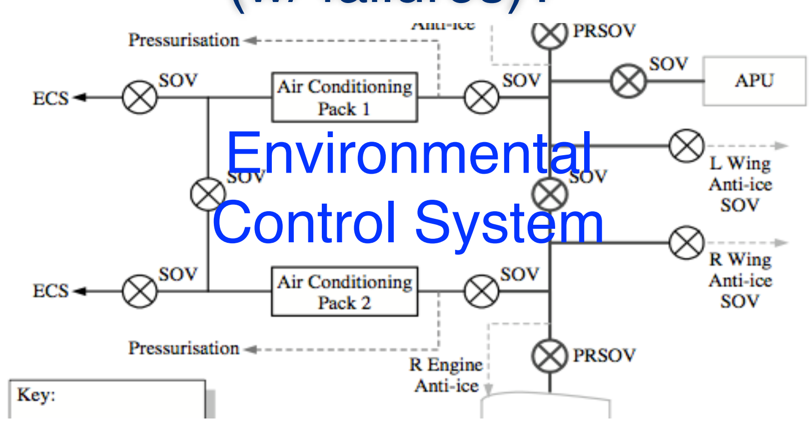


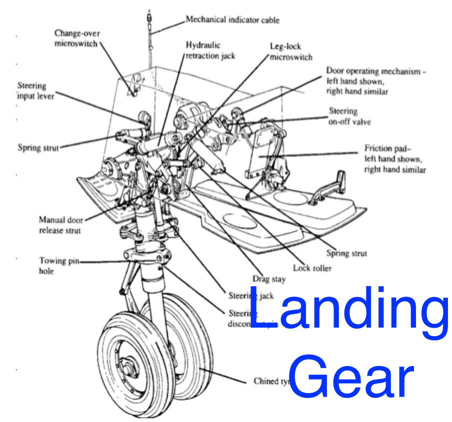




How do we design software-controlled systems of systems to insure safe operation across all operating conditions (w/ failures)?





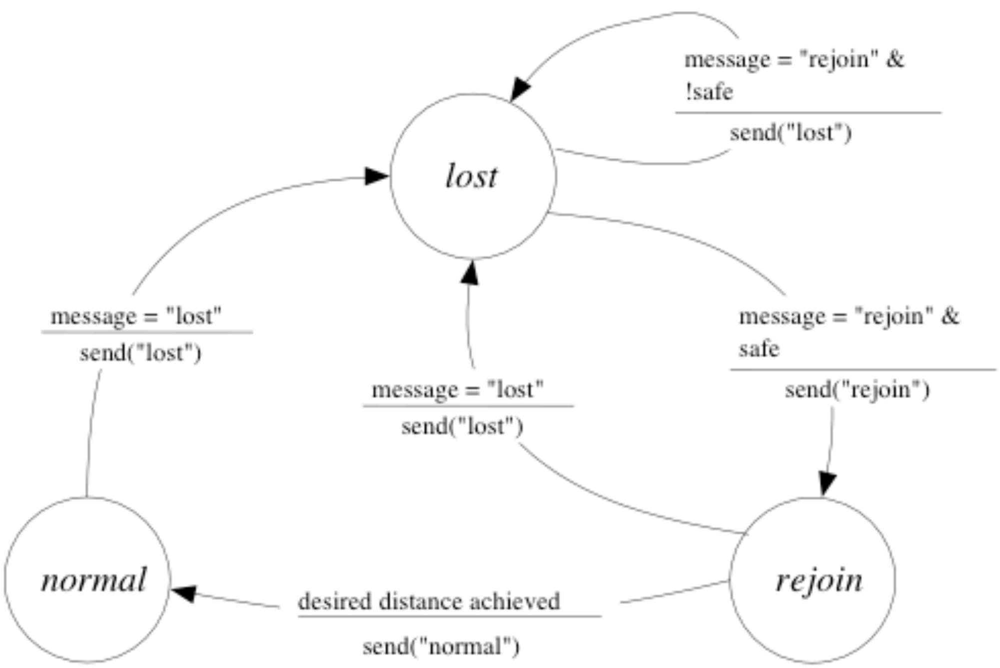


Lost Wingman Protocol Verification

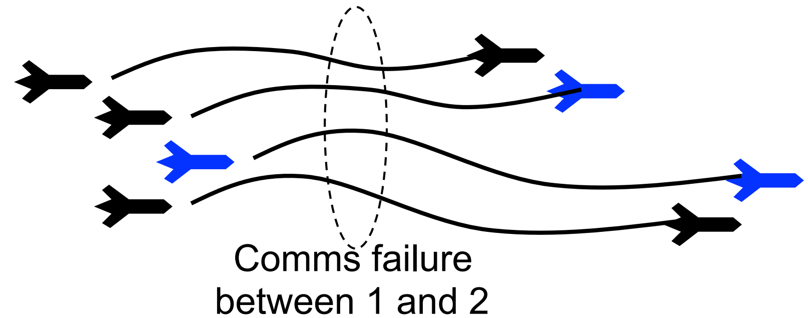
Temporal logic specification

$$\text{mode} = \text{lost} \leadsto \mathbf{stable}(d(x_l, x_f) > d_{\text{sep}})$$

- "Lost mode leads to the distance between the aircraft always being larger than d_{sep}"



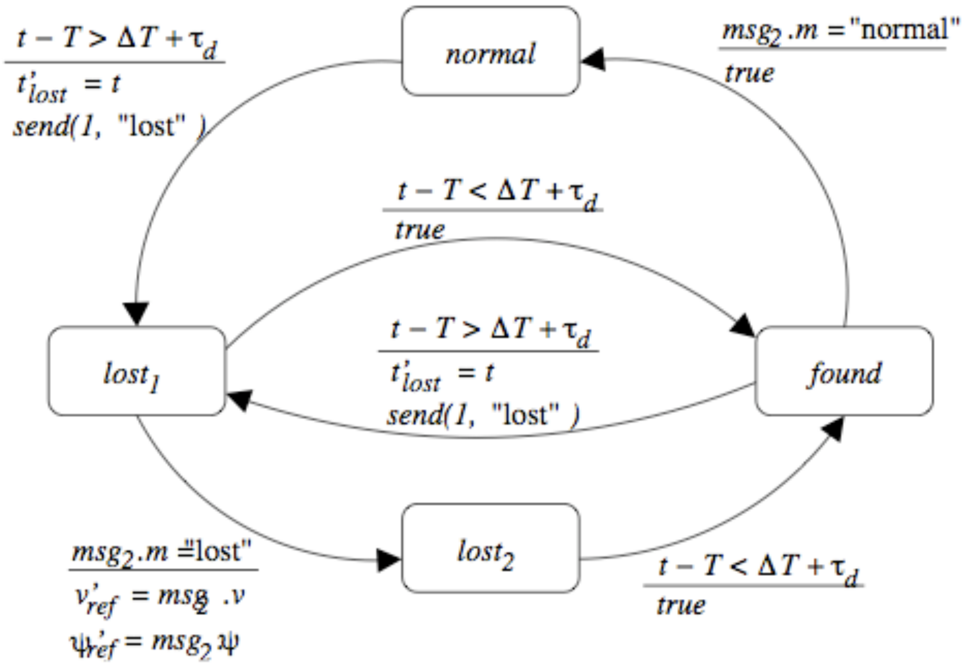
Lost wingman in fingertip formation



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



Program P_{comm}

$$\text{Initial} \quad T_s = t_0 \wedge T \in (T_s, T_s + \Delta T]$$

$$\begin{aligned} \text{Commands} \quad c_{data} &\equiv t > T \wedge data\_on : \\ & T' \in (T_s + \Delta T, T_s + \Delta T + \tau_d] \\ & \wedge T_s' = T_s + \Delta T \\ c_{msg,1} &\equiv in(1) : msg_1' = recv(1) \\ c_{msg,2} &\equiv in(2) : msg_2' = recv(2) \end{aligned}$$

Program T_{sm}

$$\text{Initial} \quad m_2 = n$$

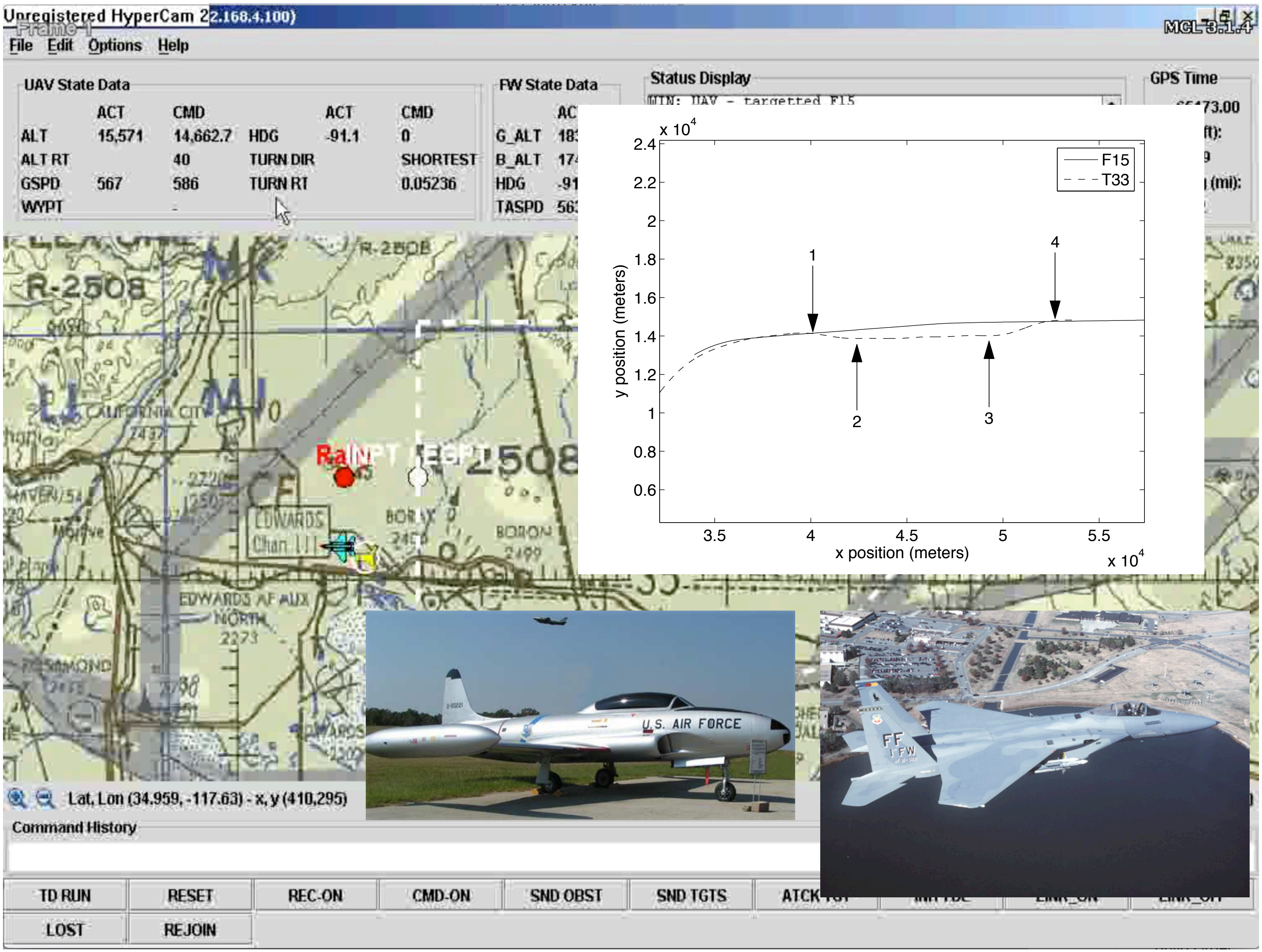
$$\begin{aligned} \text{Commands} \quad c_{lost} &\equiv m_2 \in \{n, f\} \wedge t - T > \Delta T + \tau_d : \\ & m_2' = l_1 \wedge t_{lost}' = t \\ & \wedge send(1, \text{“lost”}) \\ c_{found} &\equiv m_2 \in \{l_1, l_2\} \wedge t - T < \Delta T + \tau_d : \\ & m_2' = f \\ c_{lost2} &\equiv m_2 = l_1 \wedge msg_2.m = \text{“lost”} : \\ & m_2' = l_2 \wedge v_{ref}' = msg_2.v \\ & \wedge \psi_{ref}' = msg_2.\psi \end{aligned}$$

...

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)

Flight Test Results (June 2004)

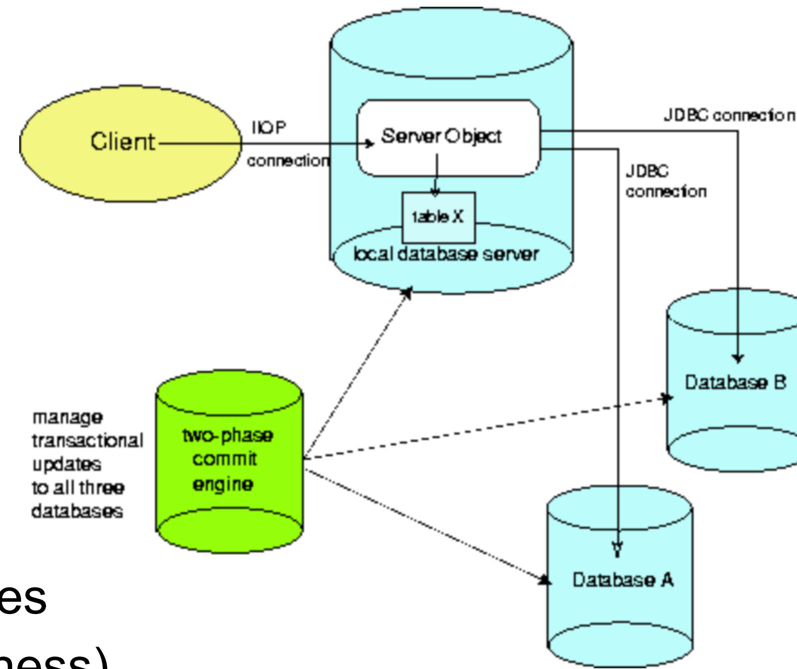


- Event 1: communications lost; T-33 executes tight turn; signals lost comms (slow link)
- Event 2: F-15 confirms communication lost message received
- Event 3: communications restored; T-33 requests rejoin (granted)
- Event 4: rejoin confirmed; return to normal operation

Summary: Introduction to Distributed Computing

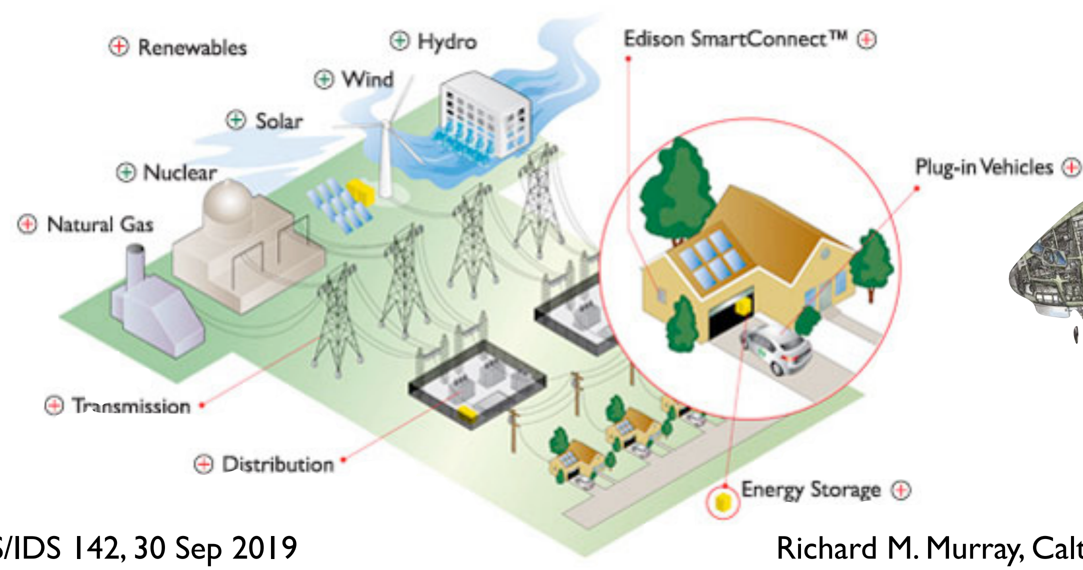
Main takeaway points

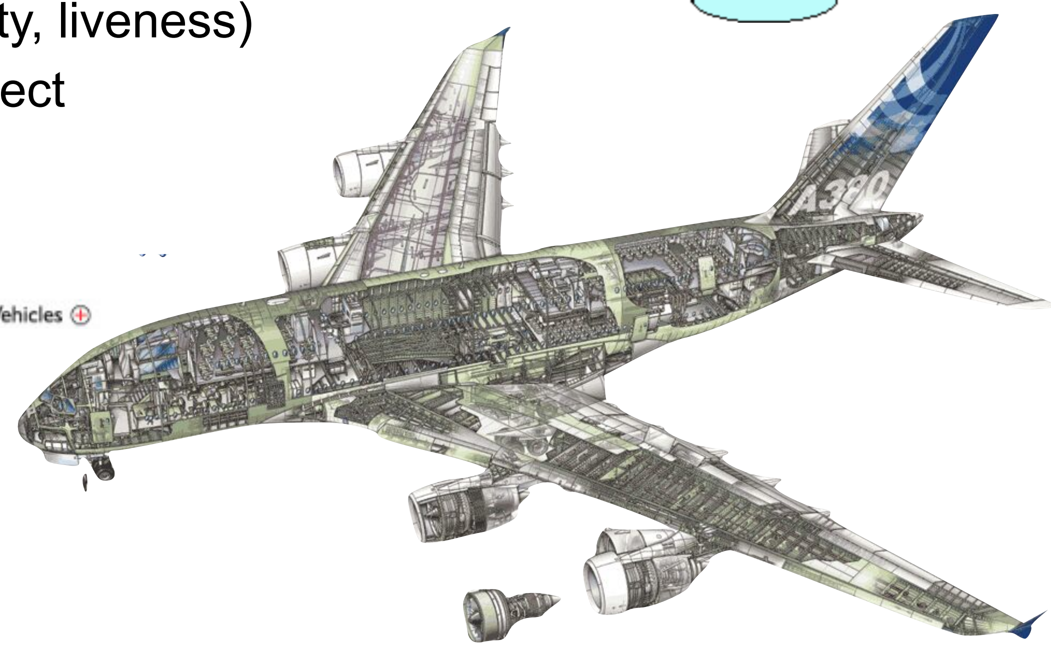
- Distributed systems (and hence distributed algorithms) are everywhere
- Debugging concurrent systems is much harder than debugging sequential programs
- For safety- (or business-) critical systems, formal proofs of correctness are key



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 (30 Sep)

- Introductory lecture, course logistics

Wednesday (2 Oct)

- Models of execution
- Finite state automata, guarded command programs
- Homework set #1 will be posted to web page (due **9** Oct)

Friday (4 Oct)

- Review of predicate calculus ($\wedge\ \vee\ \neg\ \equiv \not\equiv$), quantification $(Qi : r(i) : t(i))$
- Course ombuds announced

Homework #1 is due 9 Oct (Wed)

Online resources for the course:

- Course web page: http://www.cds.caltech.edu/cs142
- Piazza forum (Q&A): https://piazza.com/caltech/fall2019/cs142/home
- Moodle (for submitting HW): https://courses.caltech.edu → CS 142 (FA 2019)