20: Distributed systems
Review so far... embedded systems as systems

Studying **systems** means studying how all these fit together and affect each other.
Today

Distributed systems

How they communicate

Challenges

Protocols
Cars -- then

Not a computer
Central computer?
Localized computation?
Remember this from lecture 1?

Thomas Scannel, “Automotive Connectivity Evolves to Meet Demands for Speed & Bandwidth”, 2017
What are the consequences of engineering something to be made up of multiple computers?
**THE SOFTWARE CHANGE**

**Today**
- 100 million lines of code per vehicle
- Approximately $10 per line of code
- Example: Navi system 20 million lines of code

**Tomorrow**
- > 200 - 300 million lines of code are expected
- Level 5 autonomous driving will take up to 1 billion lines of code

---

**Lines of Code [Million]**

<table>
<thead>
<tr>
<th>Vehicle</th>
<th>Debian 5.0</th>
<th>Facebook</th>
<th>MS Office 2013</th>
<th>F-35 Fighter Jet</th>
<th>LinuX Kernel 3.1</th>
<th>Android</th>
<th>Google Chrome</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>80</td>
<td>60</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

**Lines of Code per Model [Million]**

- 2005: 0
- 2010: 0
- 2015: 0
- 2020: 120
- 2025: 120


**Image source**
Modern Vehicle Electronics Architecture

- **Four different computing domains**
  - Vastly different software in each domain
- **Large number of Electronic Control Units (ECU)**
  - 30-150 ECUs in cars today … and growing
- **Large software code base**
  - 100+ million lines of code in premium cars

Modern car is an increasingly complex network of electronic systems
Distributed systems

Tasks are spread across multiple computers working together to achieve a goal

Multiple products working together (smart home) or even a single product with multiple components
Ways to distribute systems

Sometimes centralized (controller + peripheral nodes), sometimes fully distributed
Send a message!

send('b')

Use [peripheral] to send bits 01100010

Output ‘00110001011’ to [pin] using [clock]

Toggle pin on clock edge

Voltage high/low signal

Message received!

Message format

Bus
Bus

A connection (wire or collection of wires) carrying data between different computer components or different computers

Sometimes refers to a specific network technology (e.g. CAN bus)

Might also see: serial bus, databus, embedded network, multiplexed wire
Message format (basic structure)

<table>
<thead>
<tr>
<th>Start</th>
<th>Header</th>
<th>Data</th>
<th>Error correction/detection</th>
<th>End</th>
</tr>
</thead>
</table>


Challenges

Design considerations
Synchronization
Control flow and data flow
Reliability
Bandwidth
Two computers send two different messages almost simultaneously. How do you determine which happened (got sent) first?
Synchronization - Keeping time

Synchronize to centralized computer
  Cristian’s algorithm, Berkeley’s algorithm
Distributed clock synchronization
  NTP - network time protocol
Logical clocks (keep track of causality rather than absolute time)
  Lamport’s logical clocks, vector clocks
Consider a bus topology

Consider messages being sent:
How would you avoid collisions?