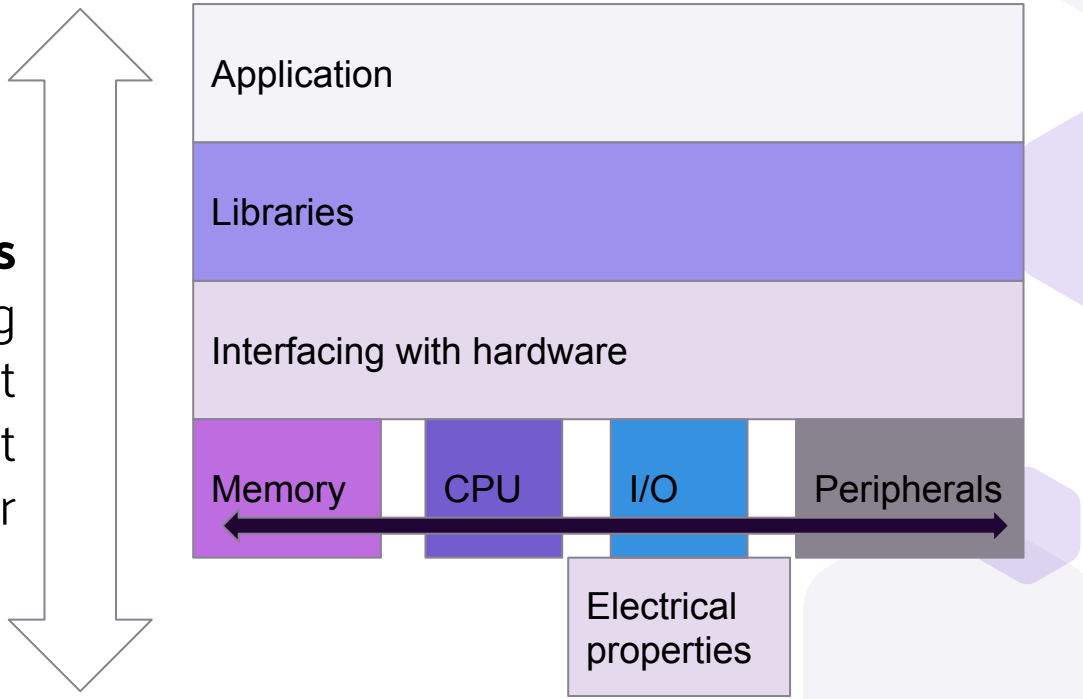
20: Distributed systems

Review so far… embedded systems as systems

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



Today

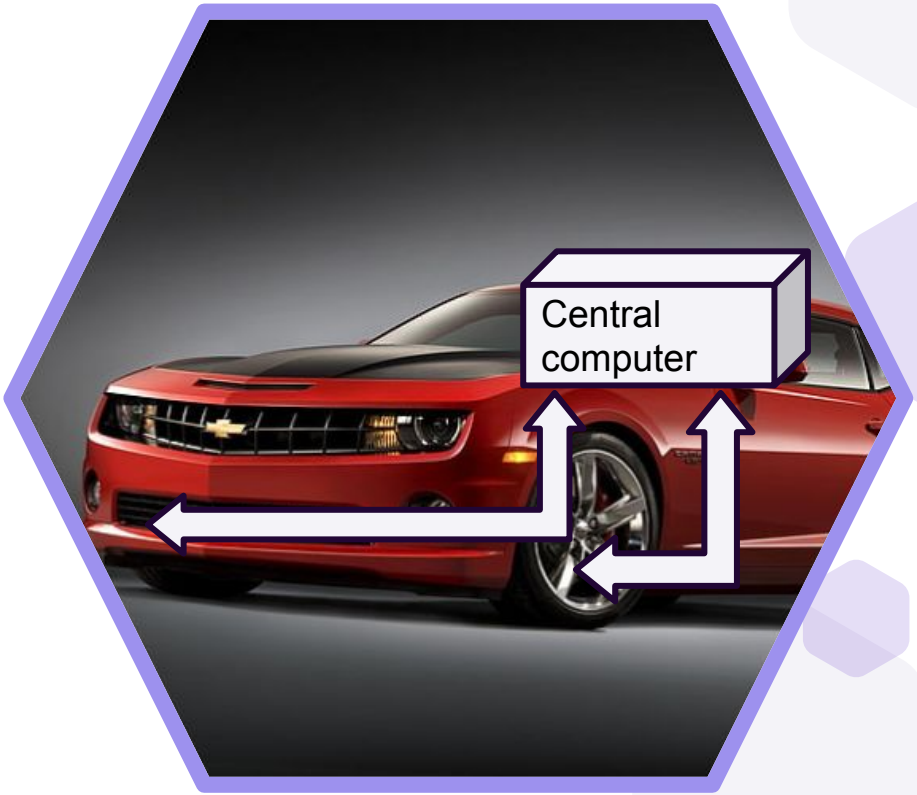
Distributed systems

- How they communicate
- Challenges
- Protocols

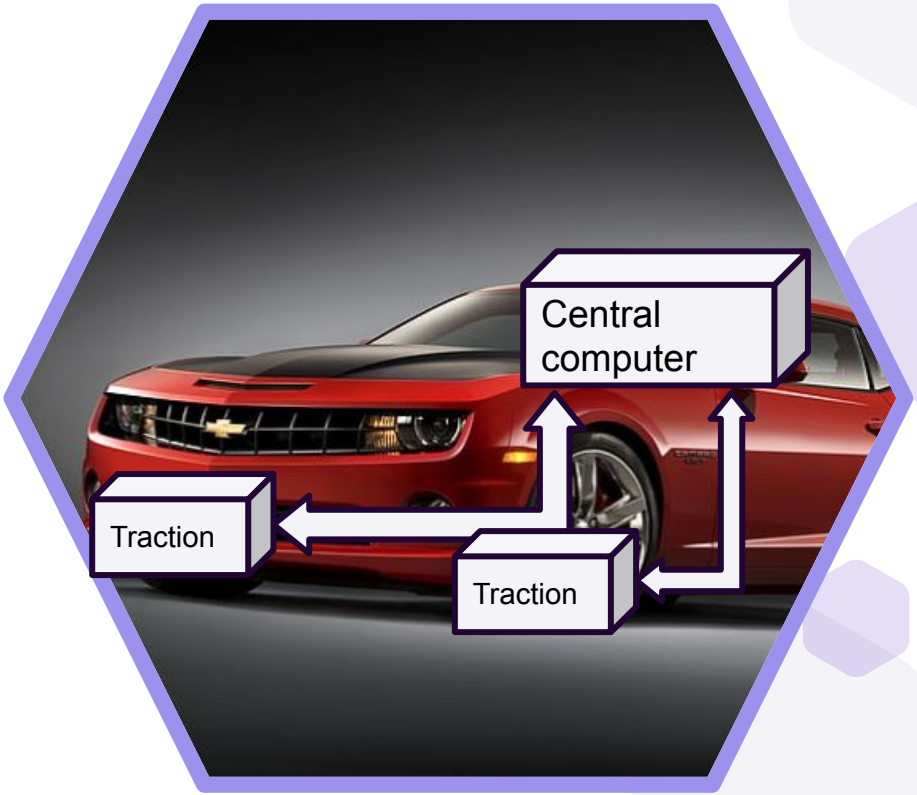
Cars -- then

Not a computer

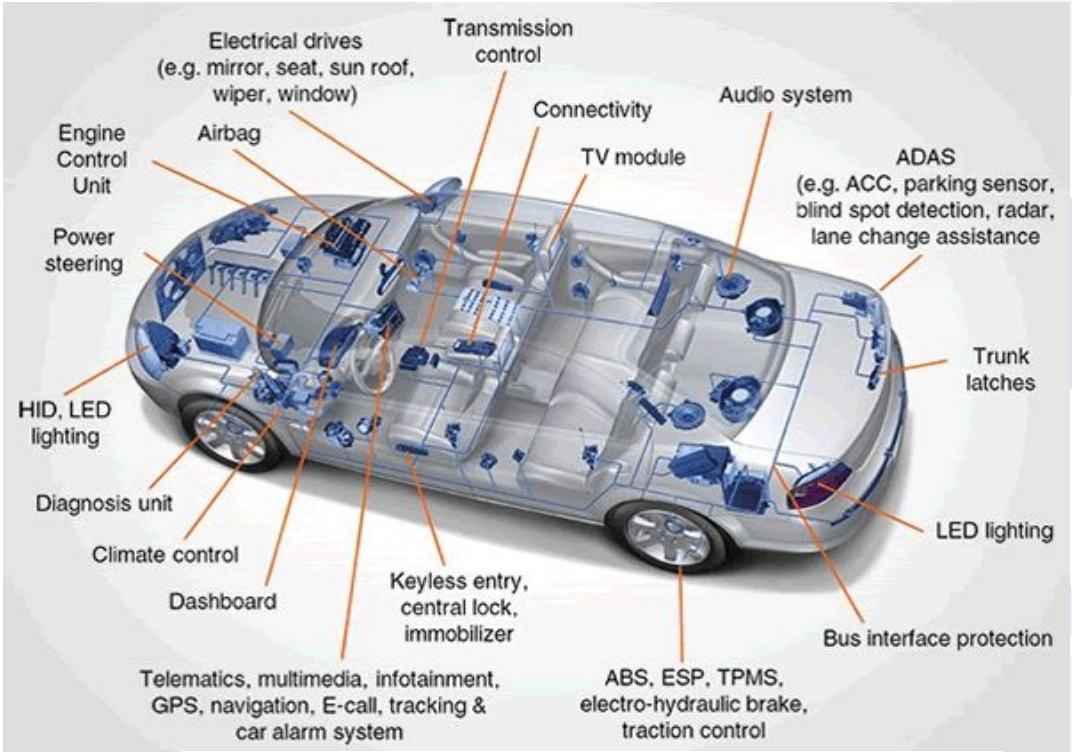
Central computer?



Localized computation?



Remember this from lecture 1?



*Thomas Scannel, "*Automotive Connectivity Evolves to Meet Demands for Speed & Bandwidth*", 2017*

What are the pros and cons of engineering something to be made up of multiple computers?

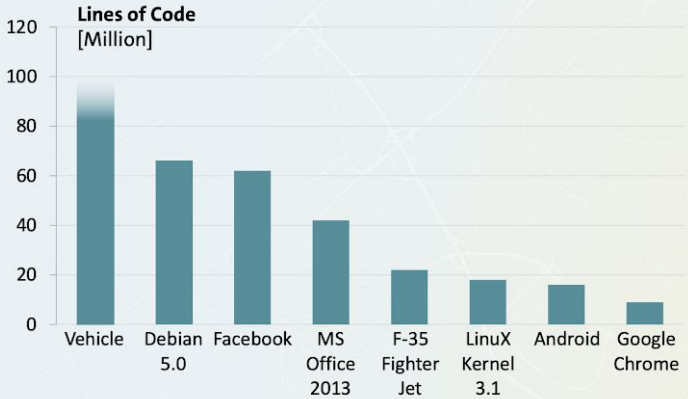
No single point of failure
Could be more secure (if you have good separation between components)
con: synchronization (increased complexity)
Increased need for intensive integration testing
    But can distribute engineering workload
Specialize hardware for tasks
Increased parallelization
Supply chain issue!
Expensive to repair?
Might save on wiring if its distributed

VOLKSWAGEN
AKTIENGESELLSCHAFT

THE SOFTWARE CHANGE

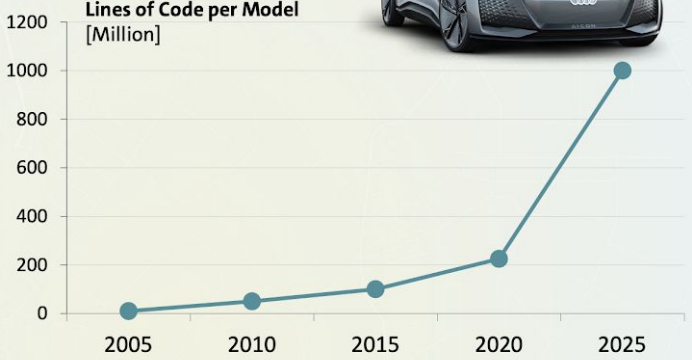
Today

- 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



Quellen: https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code | http://frost.com/prod/servlet//press-release.pag?docid=284456381 | https://www.visualcapitalist.com/millions-lines-of-code/

Image source

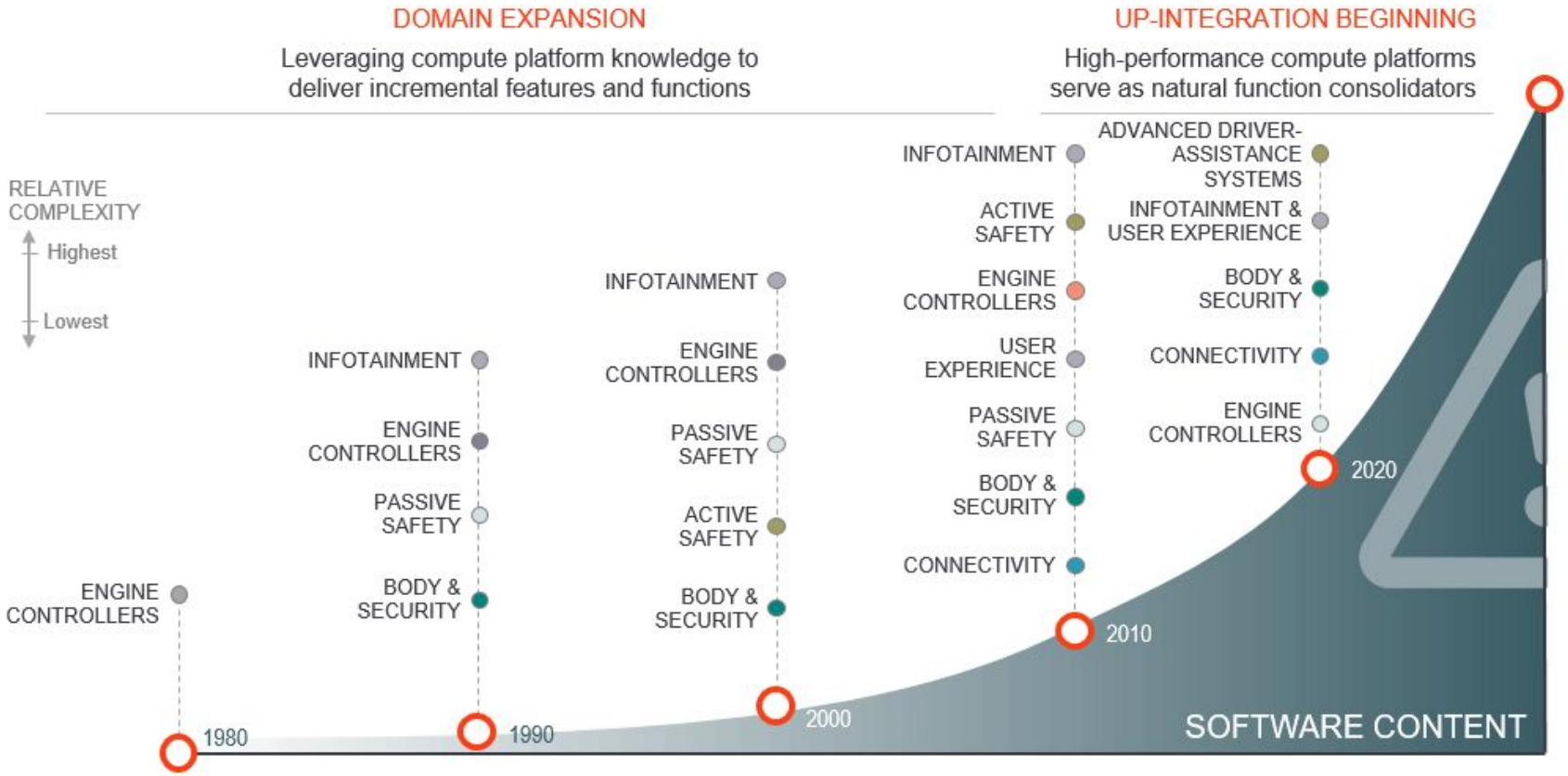
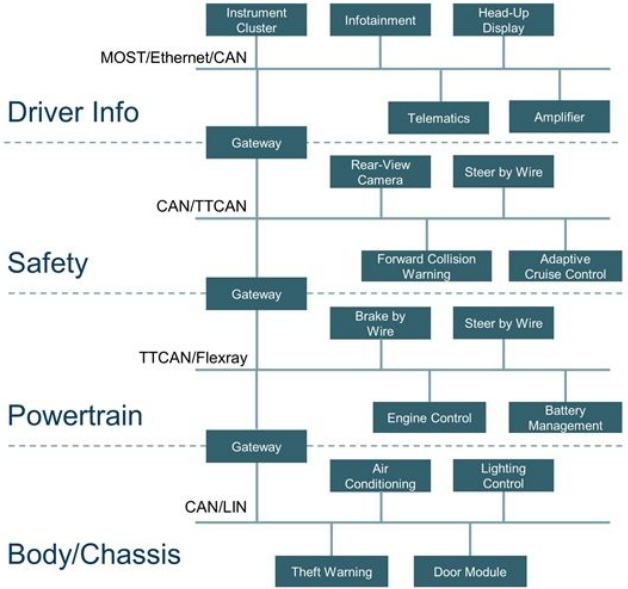


Image source

Modern Vehicle Electronics Architecture

Visteon

- **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

5

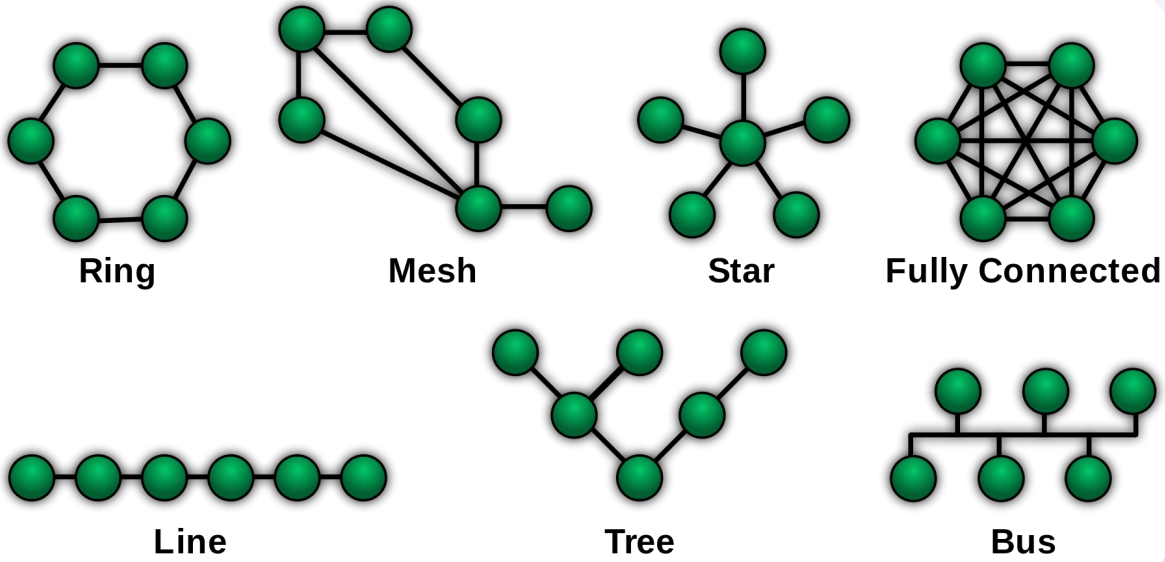
Image source

Distributed systems

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

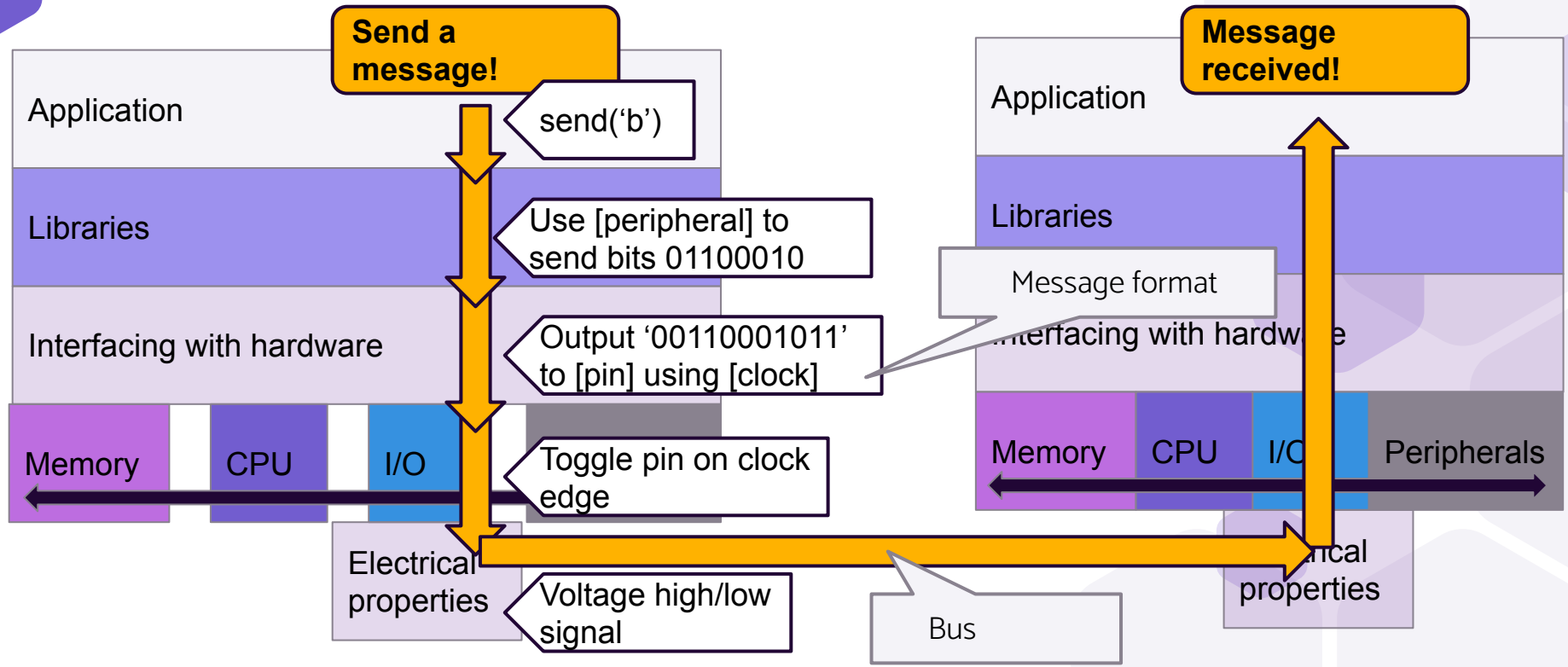
Multiple products working together (smart home) **or even** a single product with multiple components

Ways to distribute systems



Sometimes centralized (controller + peripheral nodes), sometimes fully distributed

Image source


Send a message!
Application
send('b')
Libraries
Use [peripheral] to send bits 01100010
Interfacing with hardware
Output '00110001011' to [pin] using [clock]
Memory
CPU
I/O
Toggle pin on clock edge
Electrical properties
Voltage high/low signal
Bus
Message format
Message received!
Application
Libraries
Memory
CPU
I/O
Peripherals
properties

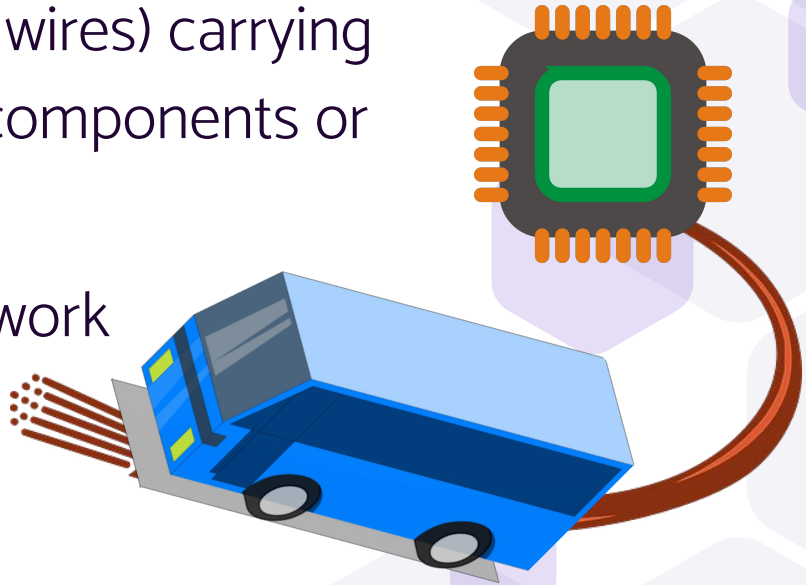
Message format (basic structure)

Start	Header	Data	Error correction/detection	End

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

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 algorithm

Distributed clock synchronization

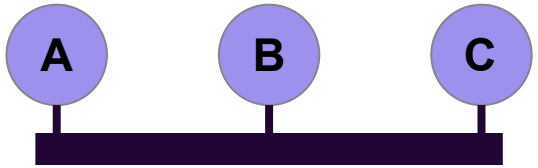
- NTP - network time protocol

Logical clocks (keep track of causality rather than absolute time)

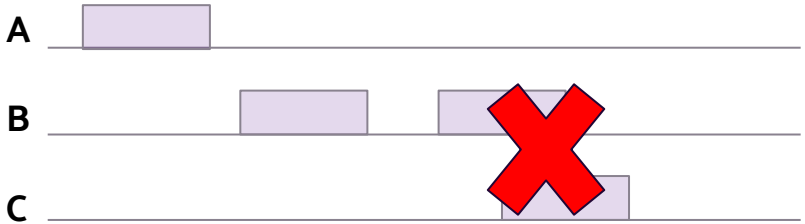
- Lamport's logical clocks, vector clocks

Control and data flow - Collisions

Consider a bus topology



Consider messages being sent:



How would you avoid collisions?