Communication, networking, and distributed systems

Milestone report and presentation

Due on **Tuesday Nov 2 at 4pm**

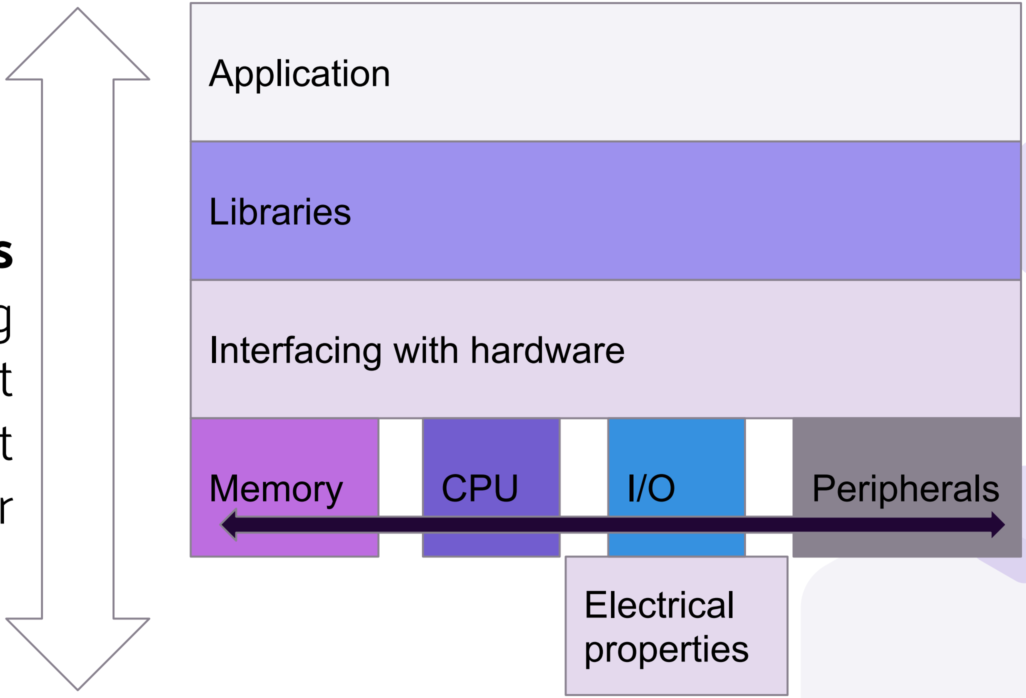
Short (10 min) presentation and demo at lab time

- Flexible to time conflicts

Peer reviews of artifacts

Review so far... embedded systems as systems

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



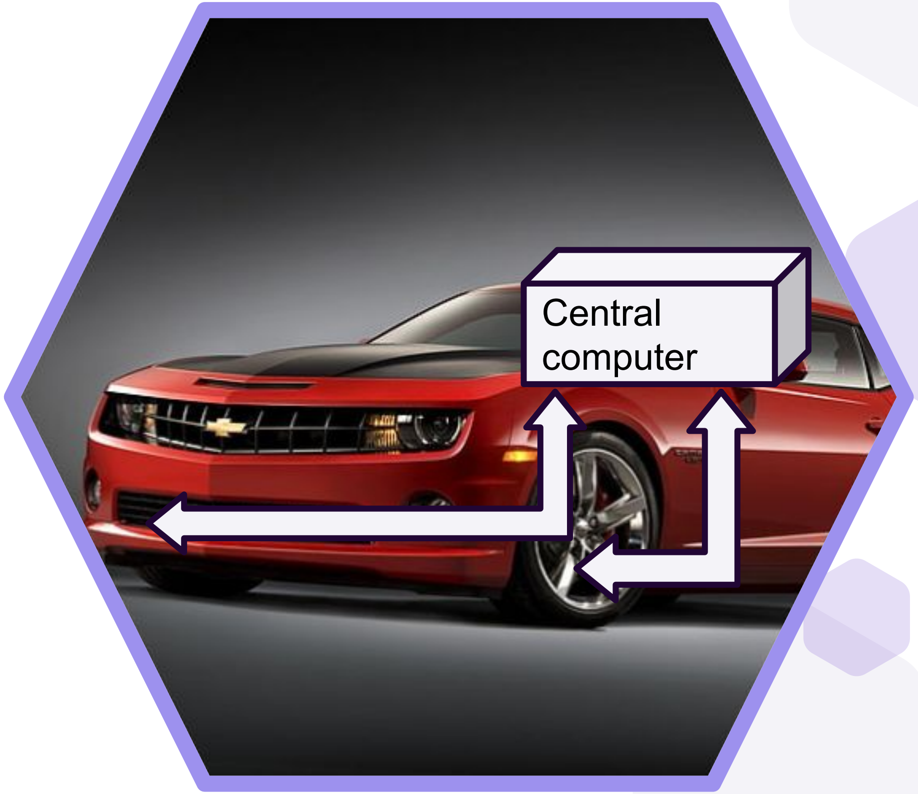
Today

- Distributed systems
    - How they communicate
    - Challenges
    - Protocols

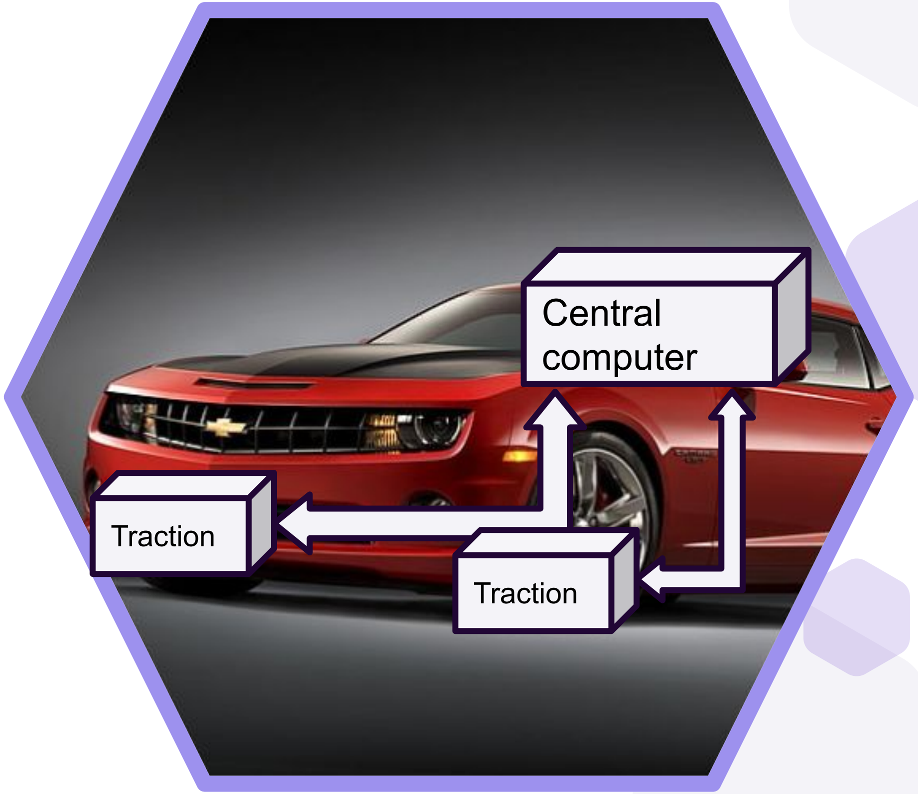
Cars -- then

Not a computer

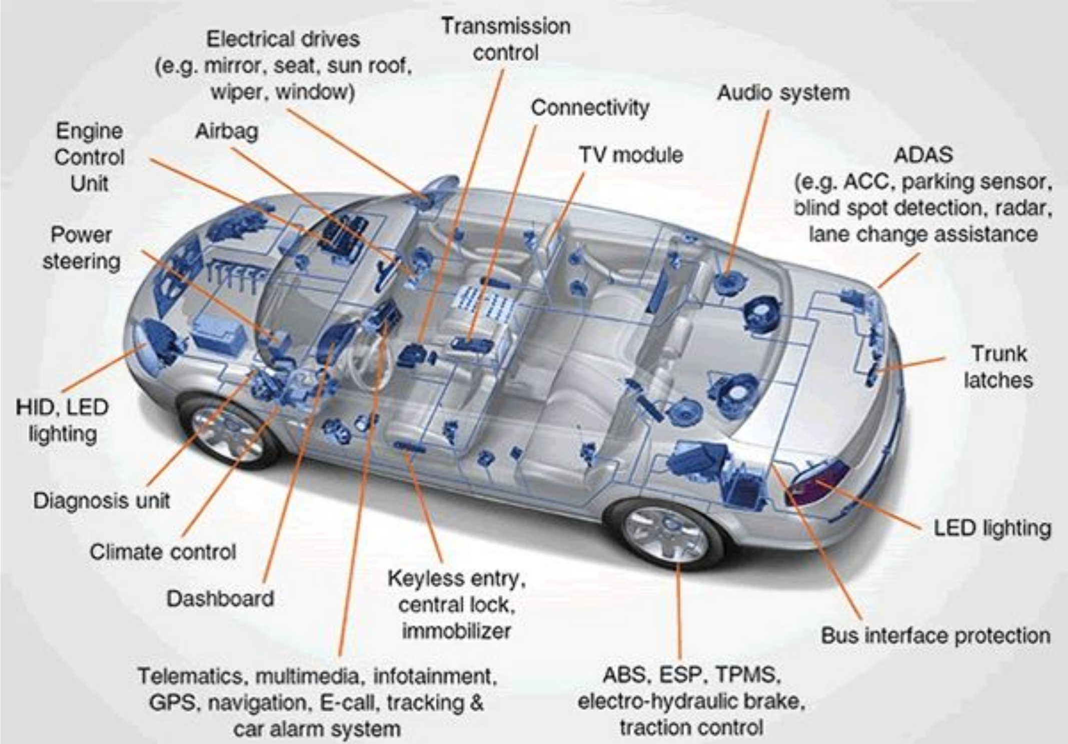
Central computer?



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?

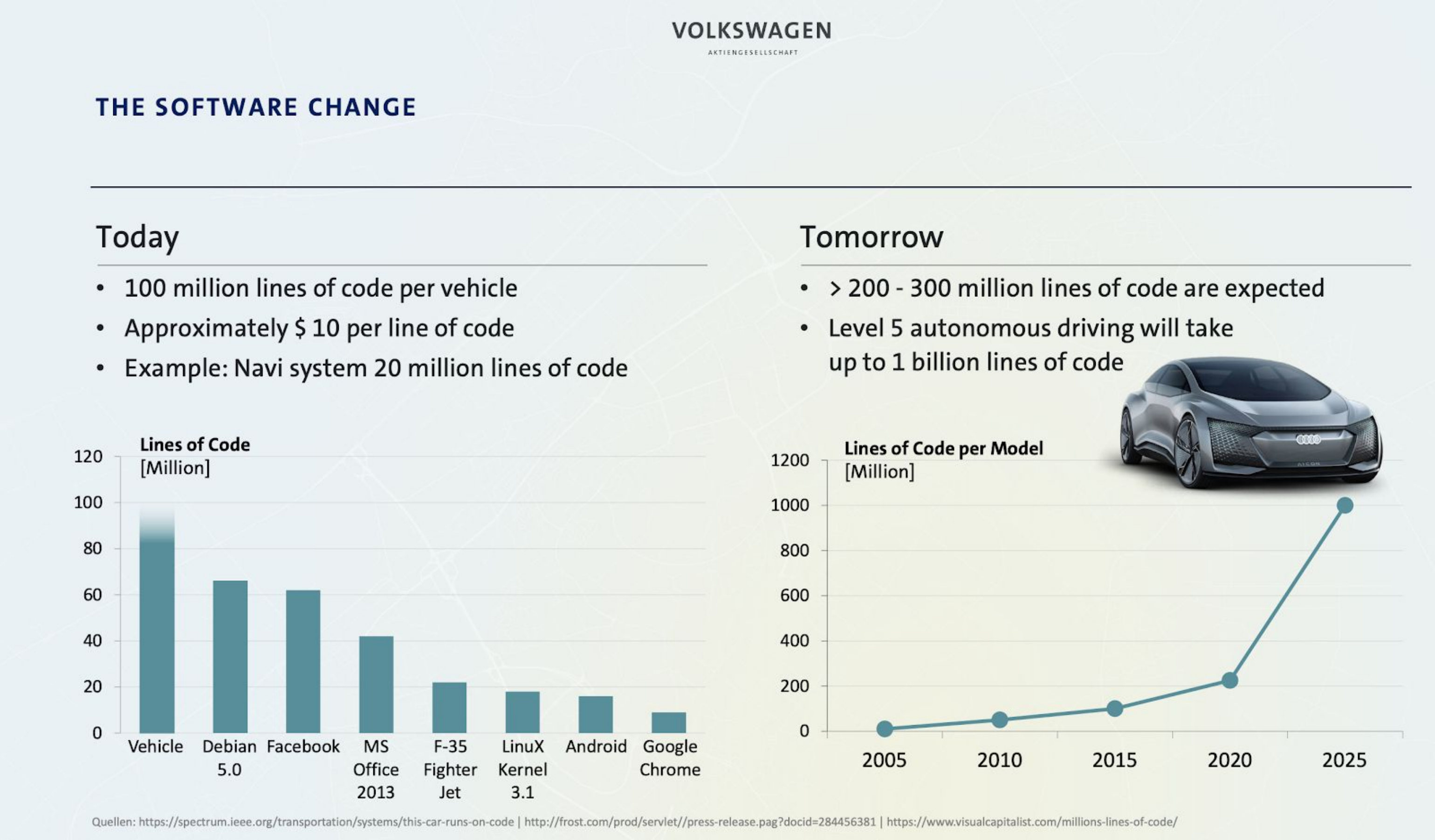


Image source

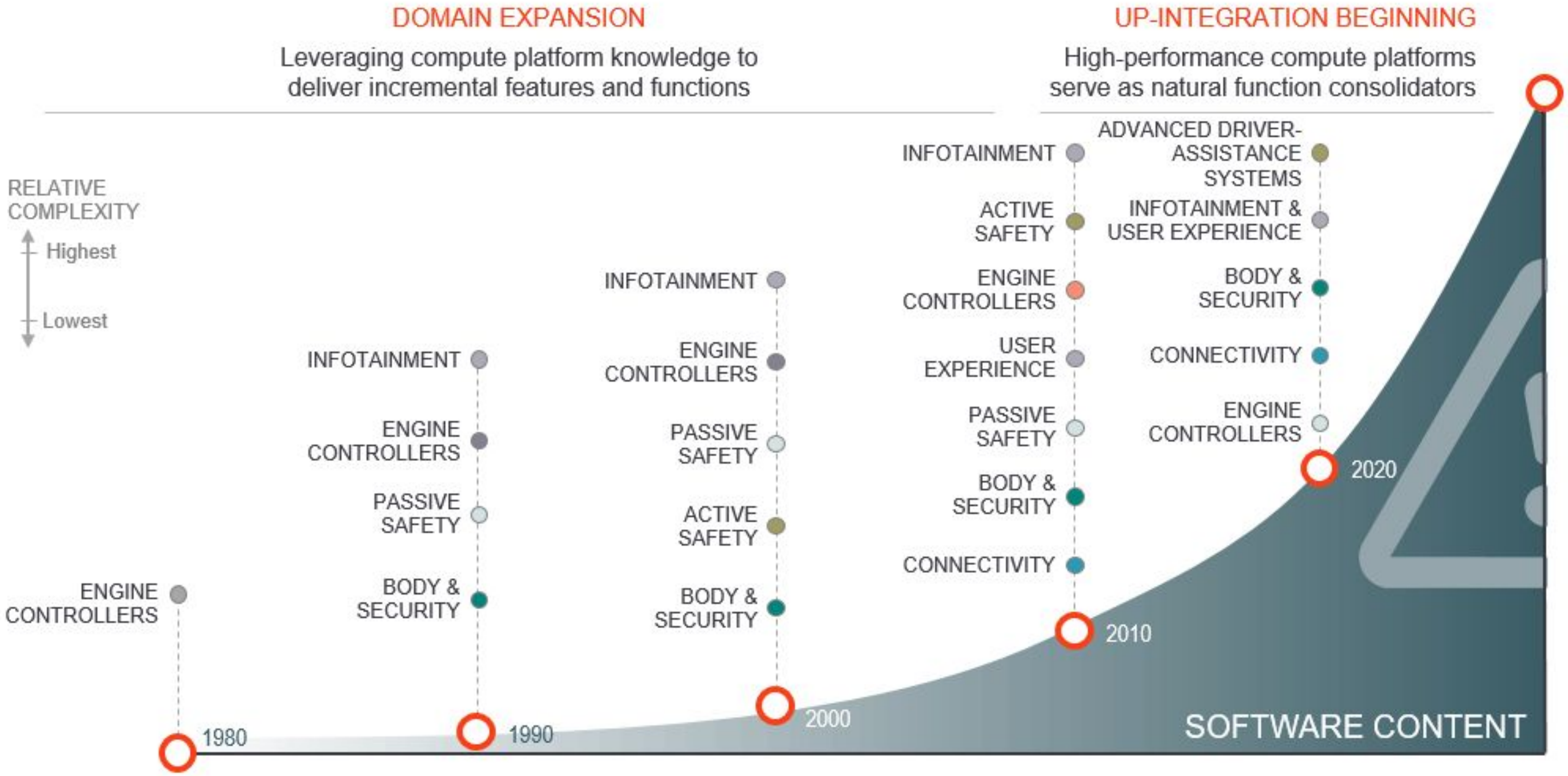
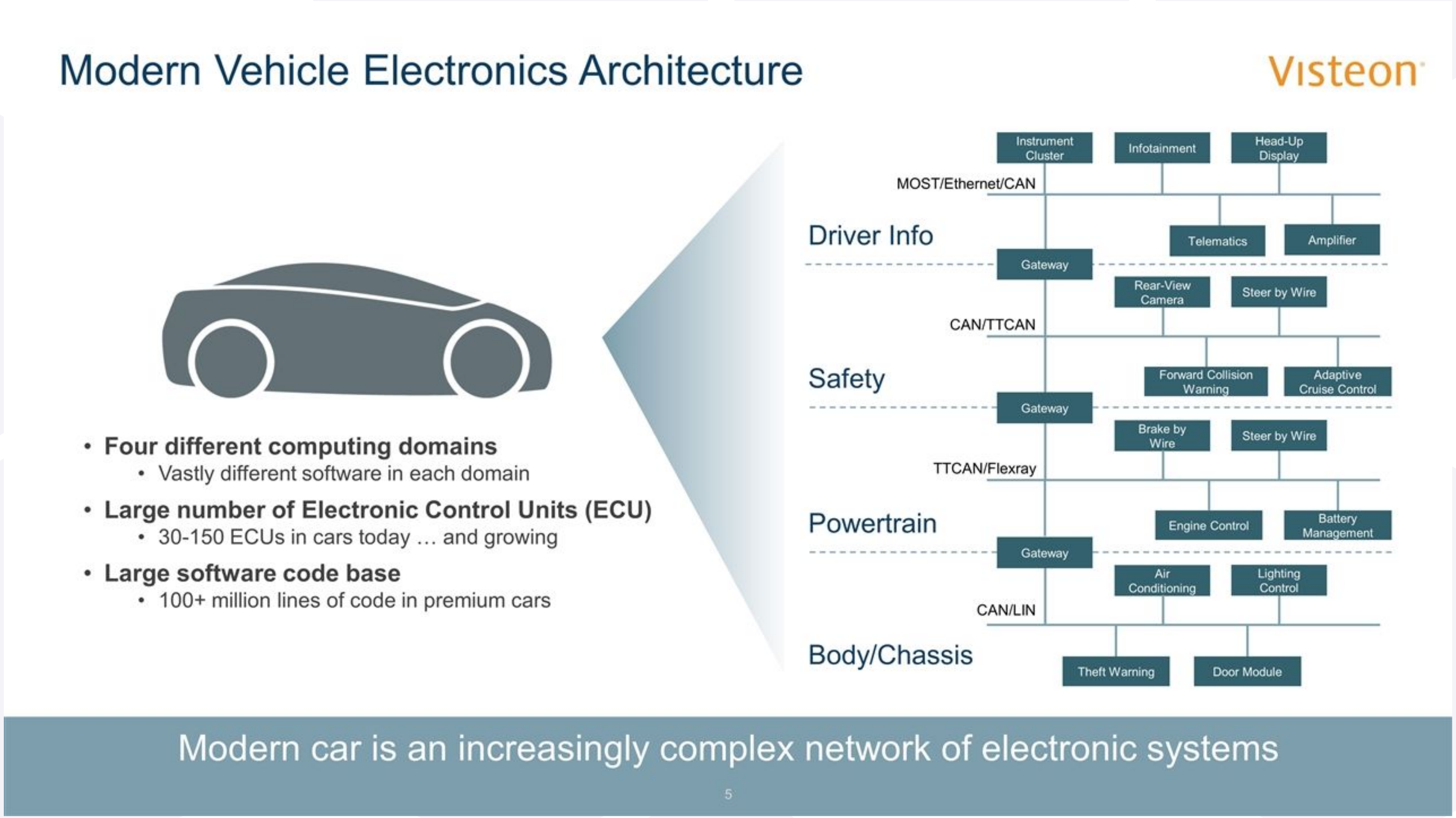


Image source



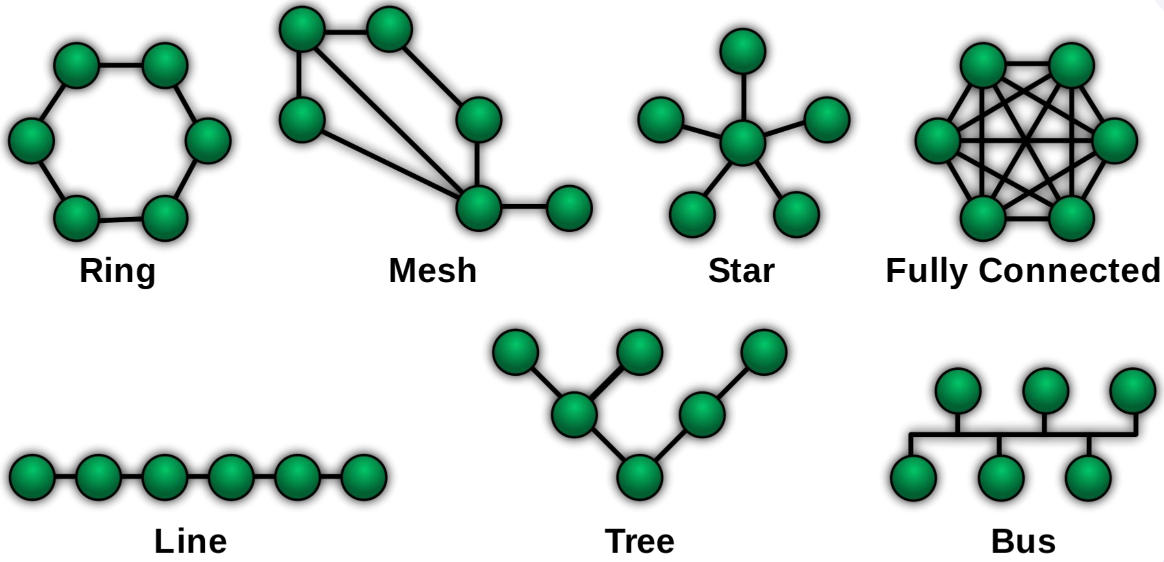
[Image source](#)

Distributed systems

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

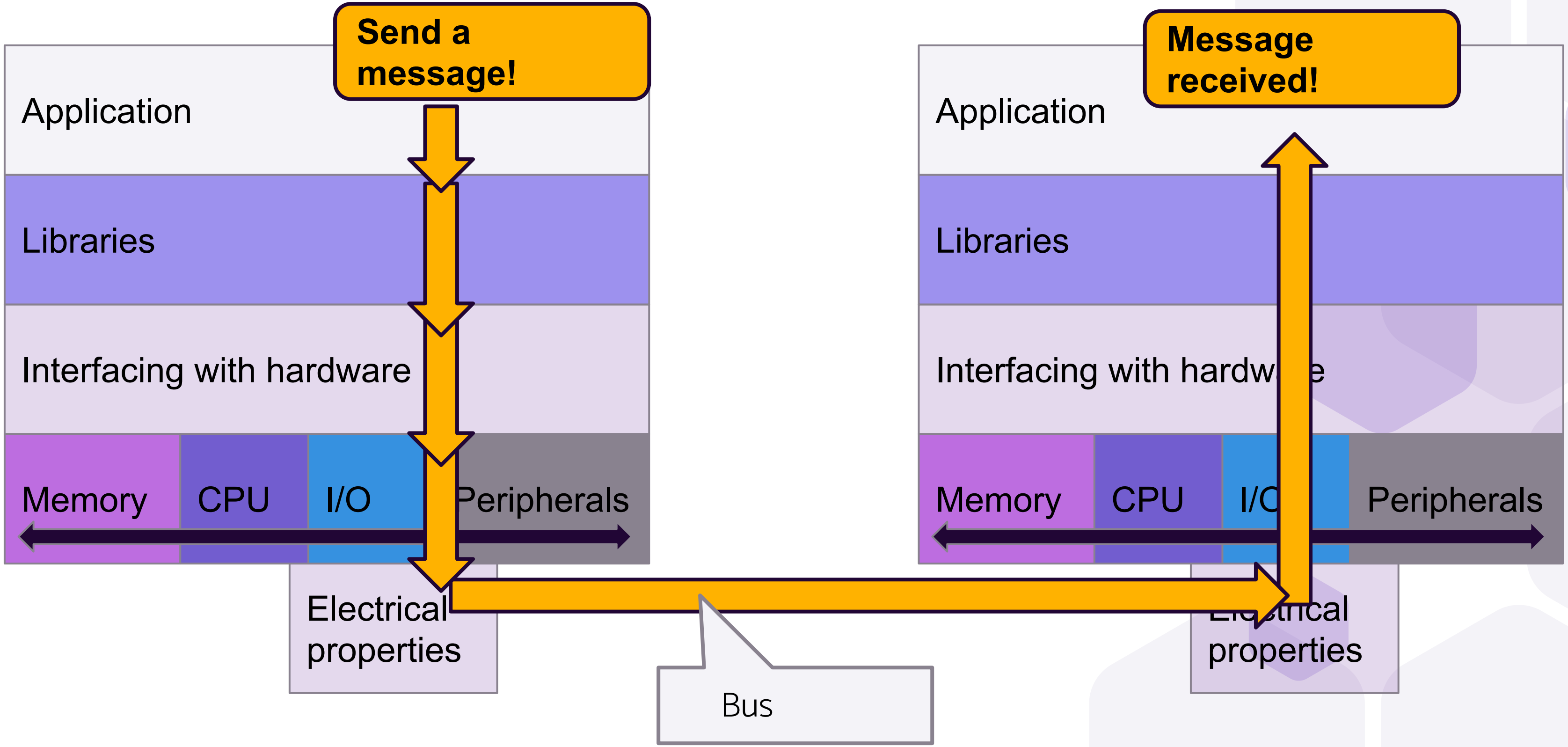
Multiple products working together (smart home) **or even** a single product with multiple components

Ways to distribute systems



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

Image source

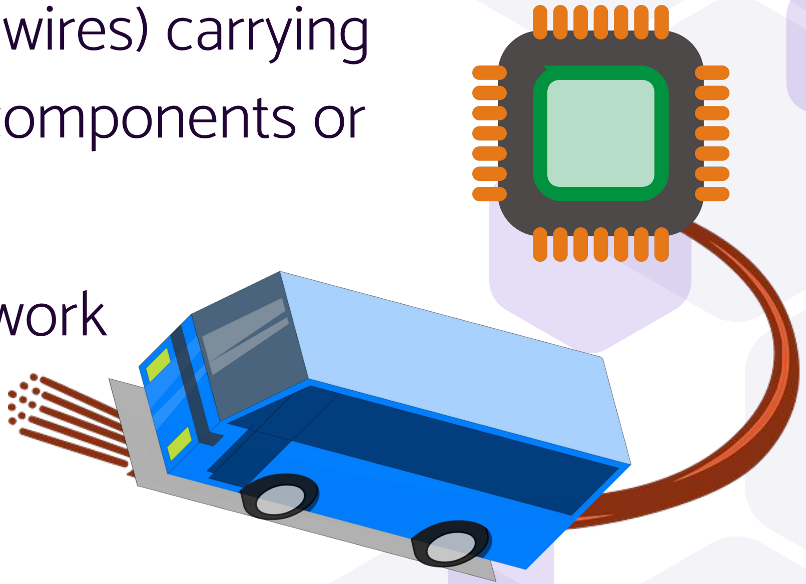


Send a message!
Application
Libraries
Interfacing with hardware
Memory
CPU
I/O
Peripherals
Electrical properties
Message received!
Application
Libraries
Interfacing with hardware
Memory
CPU
I/O
Peripherals
Electrical properties
Bus

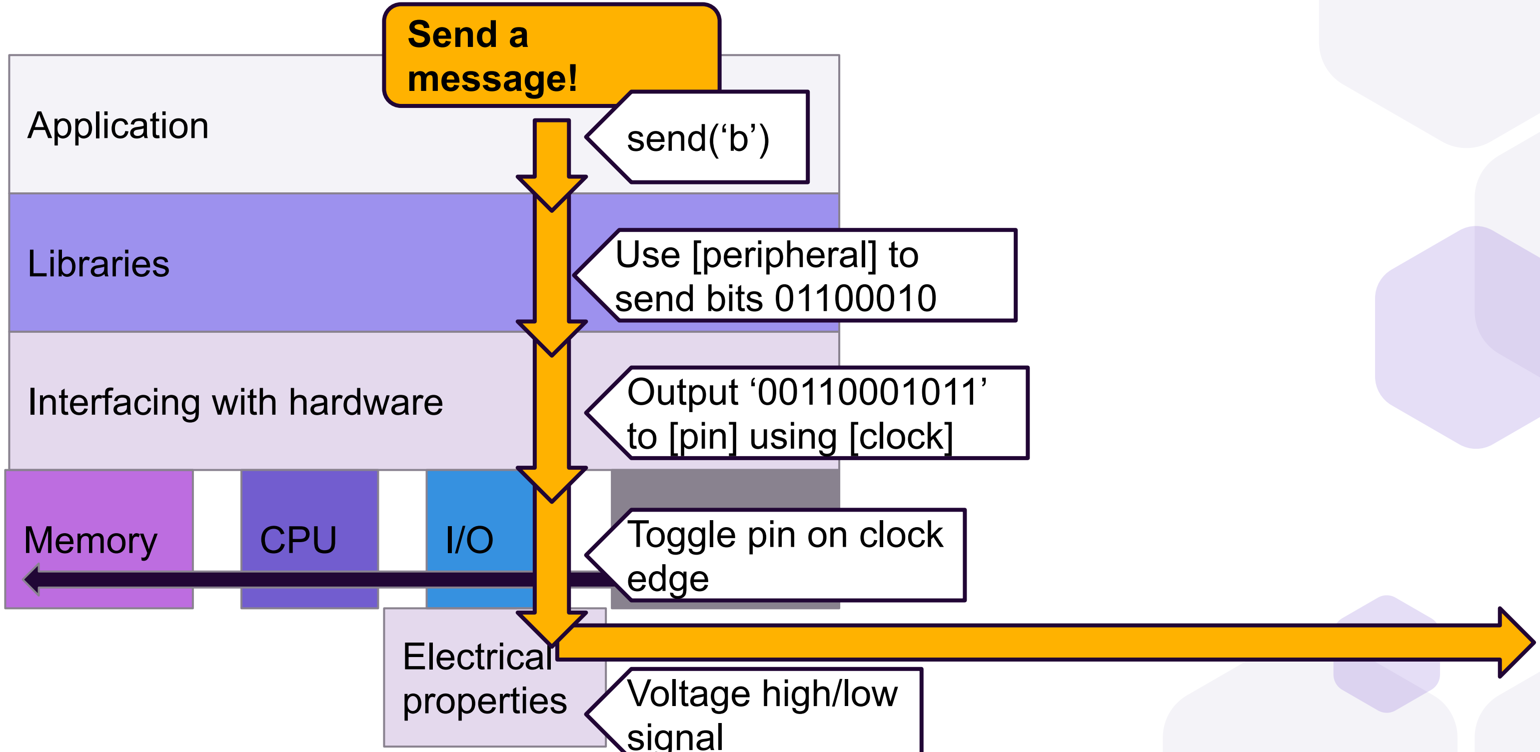
Bus

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, embedded network, multiplexed wire



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

Challenges

Design considerations

Synchronization

Control flow and data flow

Reliability

Bandwidth

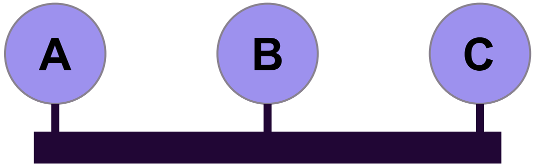
Two computers send two different messages almost simultaneously. How do you determine which happened first?

Synchronization - Keeping time

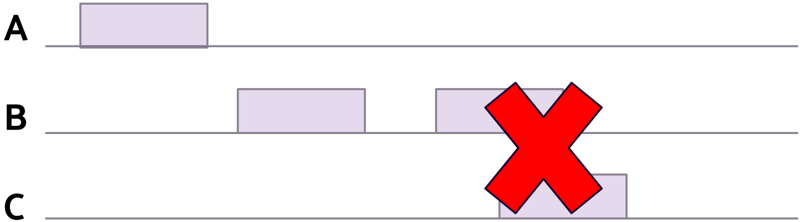
- Synchronize to centralized computer
  - Cristian's algorithm, Berkeley algorithm
- Distributed clock synchronization
  - NTP - network time protocol
- Logical clocks (keep track of causality rather than absolute time)
  - Lamport's logical clocks, vector clocks

Control and data flow - Collisions

Consider a bus topology



Consider messages being sent:



How would you avoid collisions?

Coordination on centralized system

Controller tells peripherals when it is time to transmit:

Polling: controller goes around asking peripherals if they have something to transmit

TDMA (time division multiplex access): controller sends time coordination

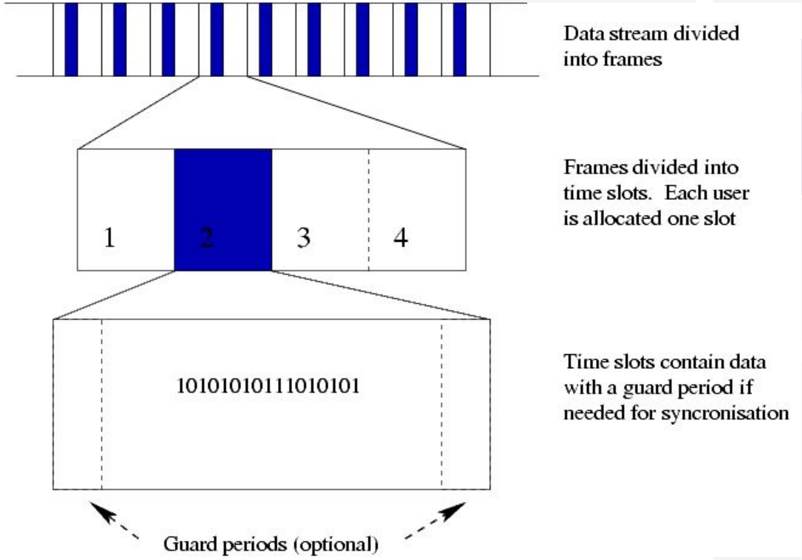


Image source

Token passing

Nodes coordinate ownership of “token” that says they can send

- Centralized system - controller passes token out
- Fully distributed system - token is passed around (e.g. round robin - “token ring”)

CSMA

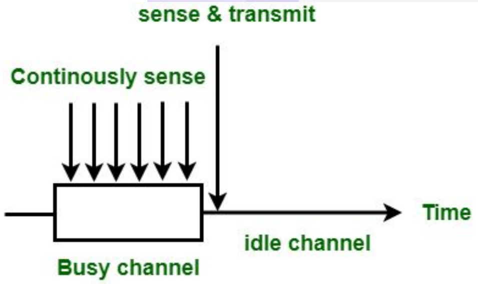
Carrier Sense Multiple Access: coordinate on fully distributed system

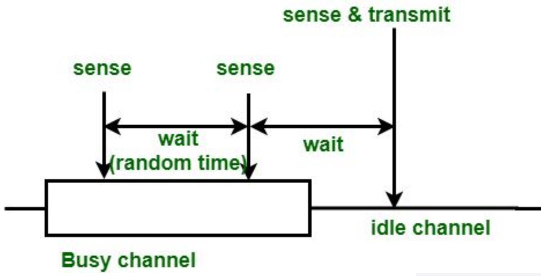
- Check if transmission line is busy before sending

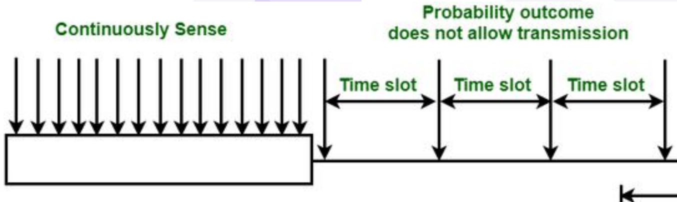
Multiple kinds:

- Check constantly (*persistent* CDMA)
- Wait before checking again (*non-persistent* CDMA)
- Transmit with probability p (*p-persistent* CDMA)

CSMA/CD (collision detection) - immediately stop transmitting when collision detected







[Image source]()

Binary countdown

Each node has an arbitration ID

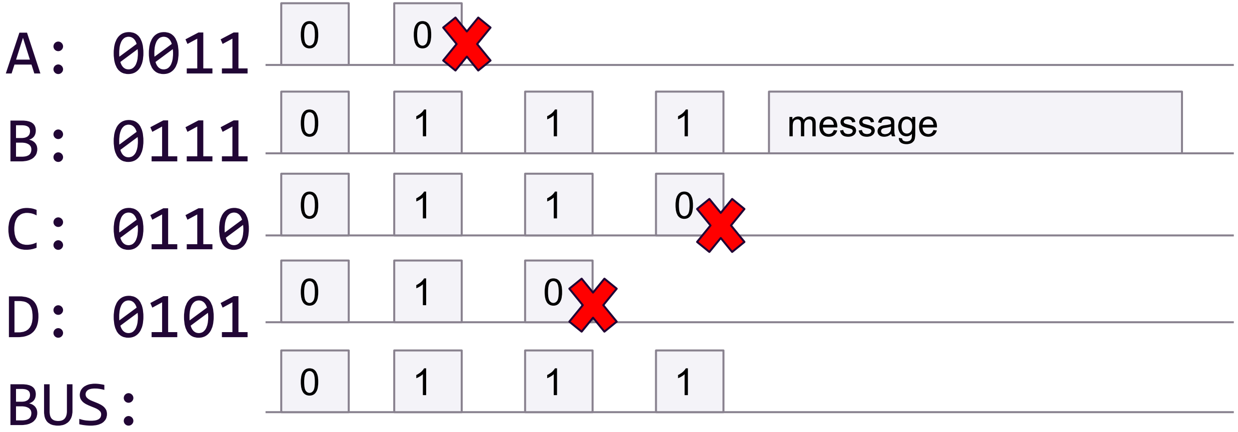
When a node wants to send, it broadcasts the first bit of the ID

If other nodes want to send at the same time, they also send their first bit

Bus is an OR of all bits

1-bit dominates, so nodes that send 0 back off

Binary countdown example



Reliability - because signals aren't perfect

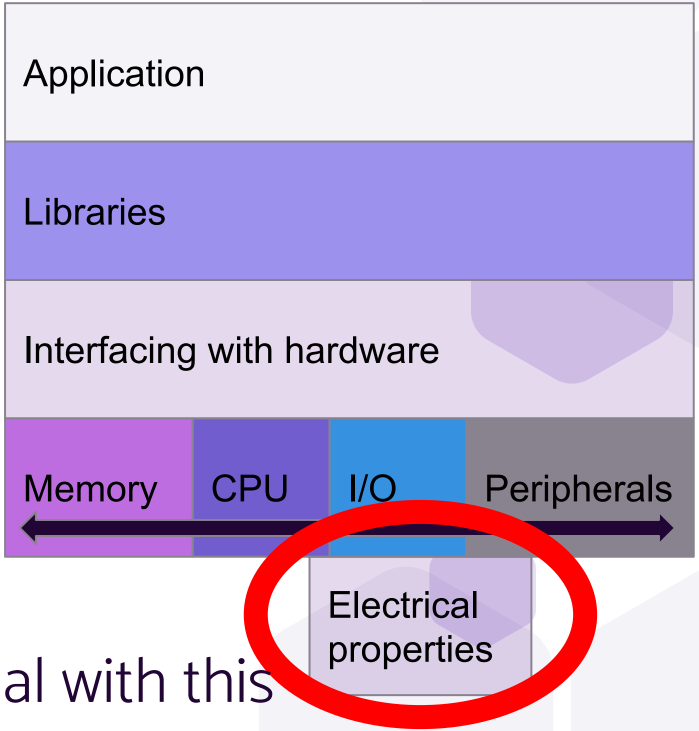
Bits are just high/low voltage signals on a wire sent w.r.t a clock

Clock may not be perfect

Wire may be noisy (electrical interference)

Wireless has even more dangers

ACKs are one way we've seen to deal with this



Checksums

A computation on the data that describes something about the contents of the data

Example: parity (number of 1s odd or even)

0110 has parity 0

0111 has parity 1

Sender sends data and checksum

Receiver receives data and compares computed checksum with received checksum

Say a sender sends 7 data bits followed by a parity bit. Which of these messages will be rejected by the receiver?

```
A: 0000 0000
B: 1110 0000
C: 1111 0101
```

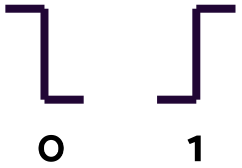
Reliability: RZ/NRZ

Return to zero/no return to zero

NRZ: signal can output the same value for arbitrary time

RZ: signal must have an edge every once in a while

Manchester encoding:



What are the tradeoffs between NRZ and RZ?

Differential drivers for noisy signals

Assumption is that noise is somewhat correlated for two signals sent at the same time on two adjacent wires

Subtract the signals to reduce noise

Specific protocols

What a message looks like:

Start bit(s)	Header	Data	Error detection	End bit(s)

Serial protocols: message sent as a sequence of bits on one wire

UART

Start bit: 0	Data (7-9 bits)	Parity bit	End bit: 1

Universal Asynchronous Receiver-Transmitter

Two components communicating

Each has transmit (TX) and receive (RX) line

Do not need synchronized clock (just both components at same frequency

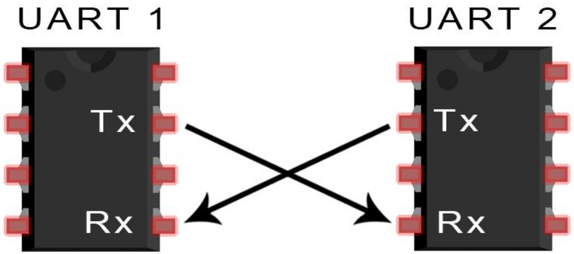


Image source

Problems with UART?

SPI

Serial peripheral interface

Controller sends clock to peripheral and transmits with clock

Transmits clock for longer so peripheral can respond

Multiple peripherals: chip select line

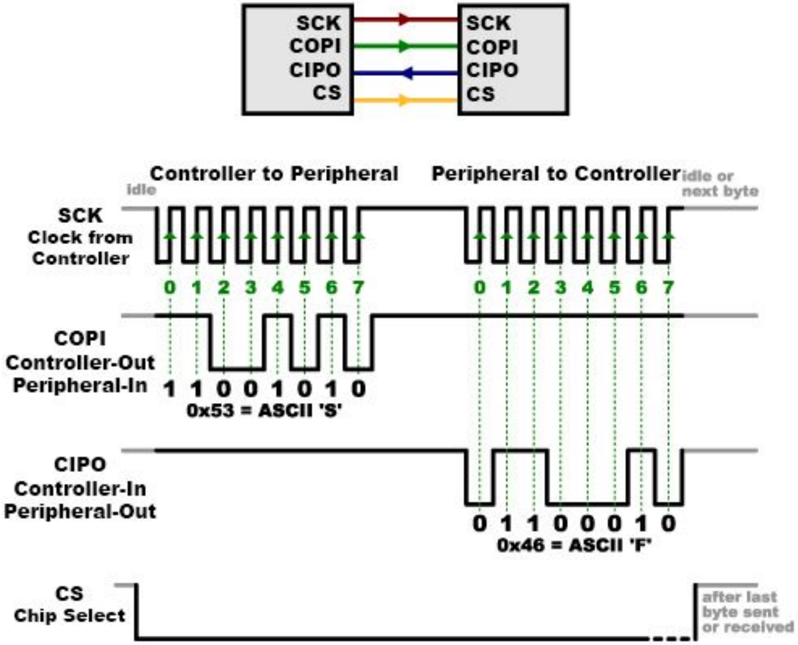


Image source

Problems with SPI?

I2C

Start bit	Address (7 or 10 bits)	R/W bit	Data (8 bits)	ACK bit	Data (8 bits)	ACK bit	...	End bit

Inter-integrated circuit

Controller uses address to select which peripheral it is communicating with

Timing of SDA/SCL means this protocol supports multiple controllers

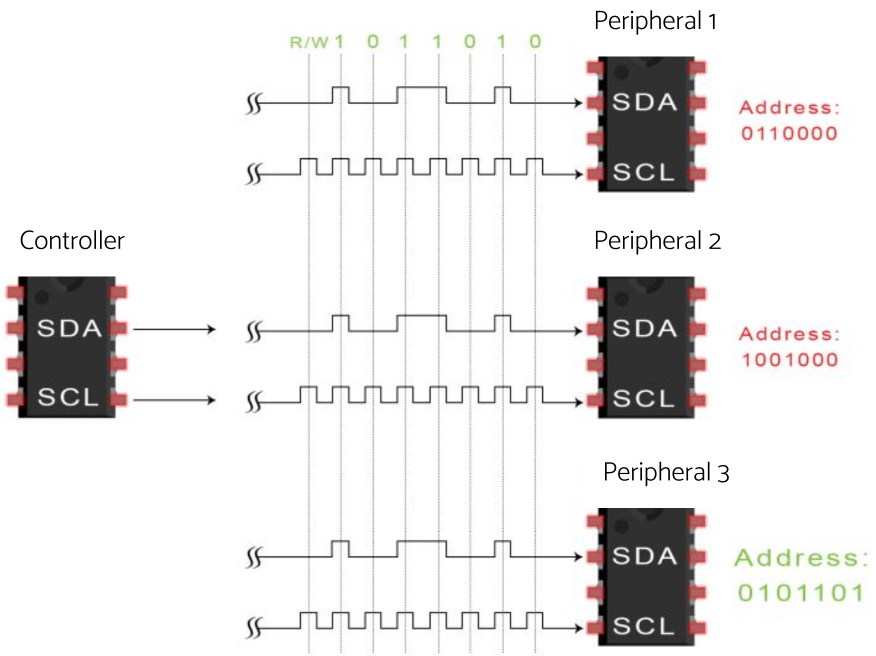


Image source

Problems with l2c?

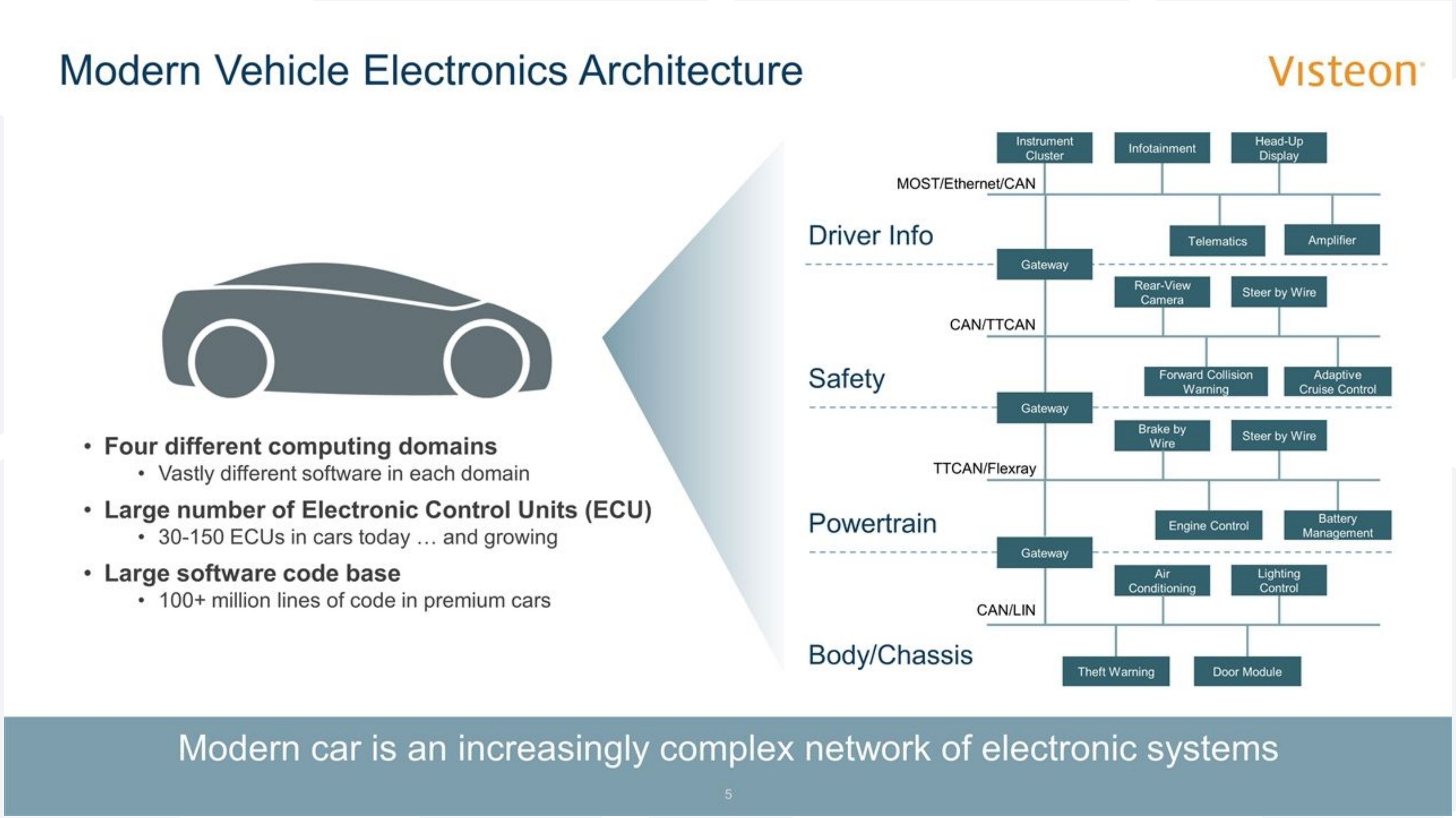


Image source

CAN

Start bit	CAN-ID (29 bits)	Transmission request bit	Control (6 bits)	Data (0-8 **bytes**)	CRC (16 bits)	ACK (2 bits)	End bit

Controller Area Network

Used for safety-critical applications (cars)

Binary countdown for arbitration

NRZ encoding with bit stuffing

We have discussed serial buses. Why are parallel buses challenging?

Keeping track of data - buffers

Way for main process and transmitter/receiver to produce/consume data at different rates

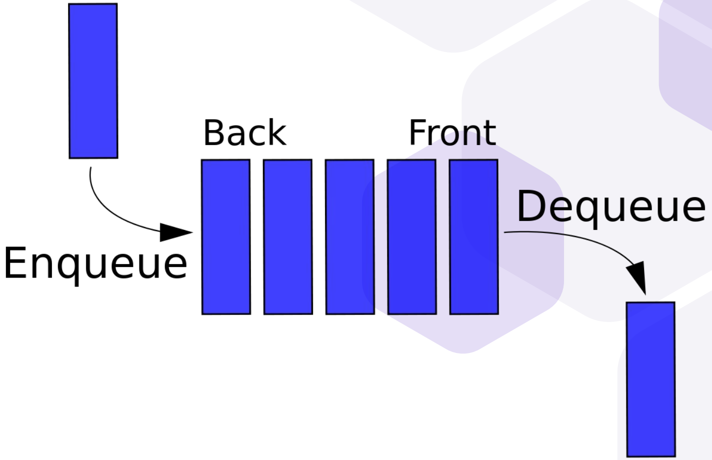
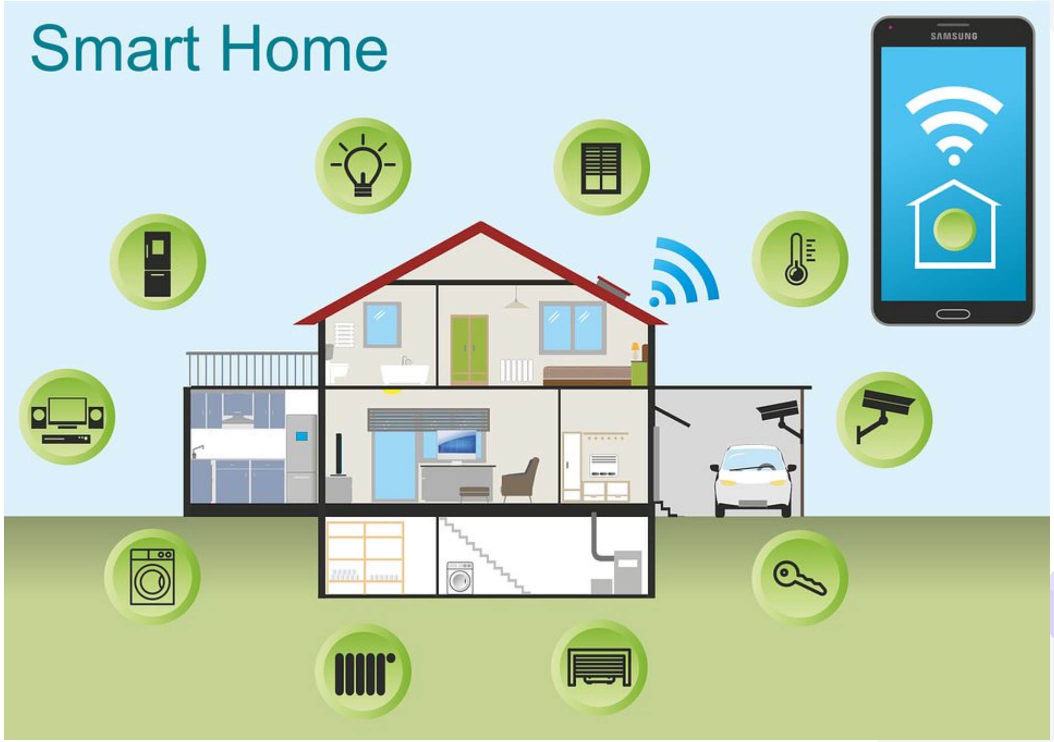


Image source

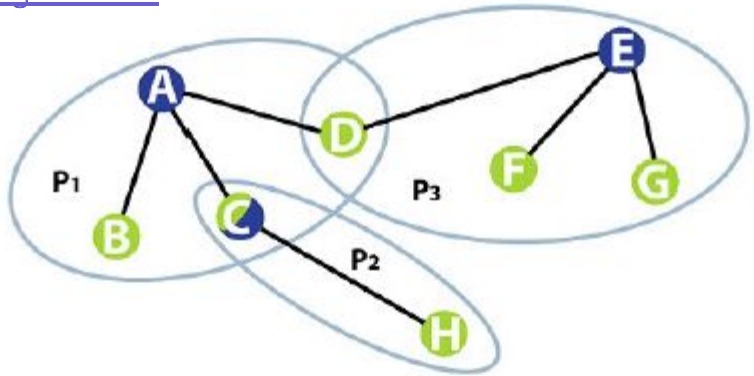
Wireless communication



What are some concerns/considerations that specifically concern wireless communication?

Bluetooth

Image source



Scatternets made up of piconets

- 1 controller, up to 7 peripherals
- Components can drop out of piconet at any time
- Elect new controller if controller leaves

Signals use frequency modulation

Frequency hop across set of frequencies in case of collision

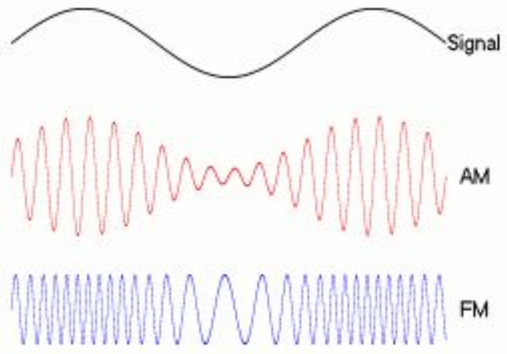


Image source

WiFi

Internet connectivity

Packets (message) transmitted on a frequency band

Use CSMA for collisions

Communicate with web servers using HTTP protocols

Summary

Study of embedded **systems** means studying how all layers affect each other

Distributed systems - multiple embedded systems talking to each other

Considerations - coordination, synchronization, reliability....