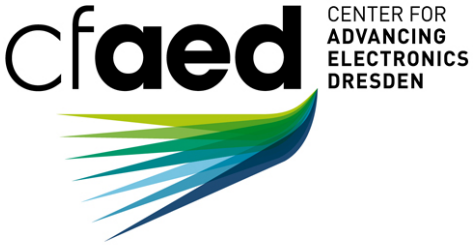
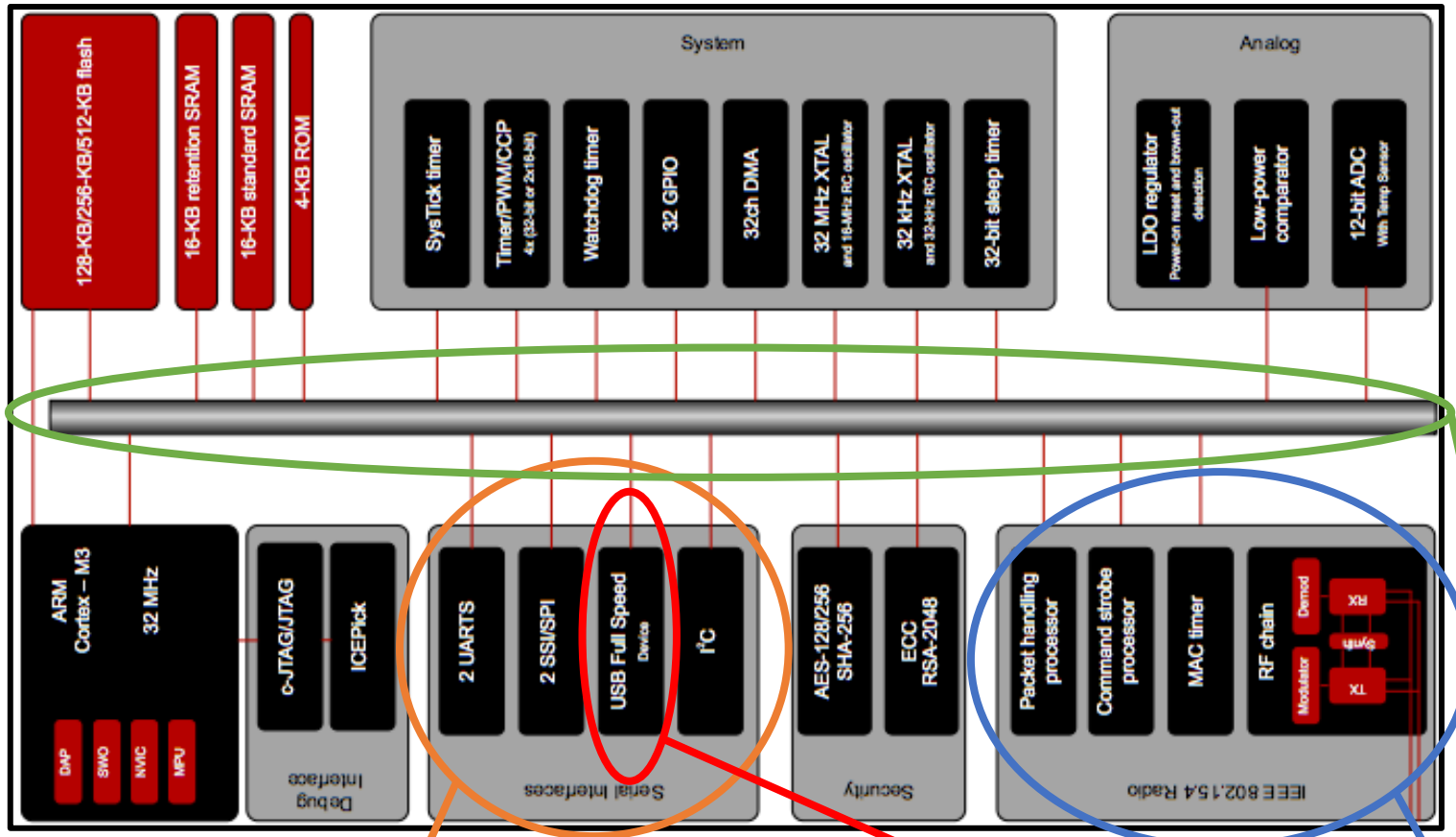
Networked Embedded Systems WS 2016/17

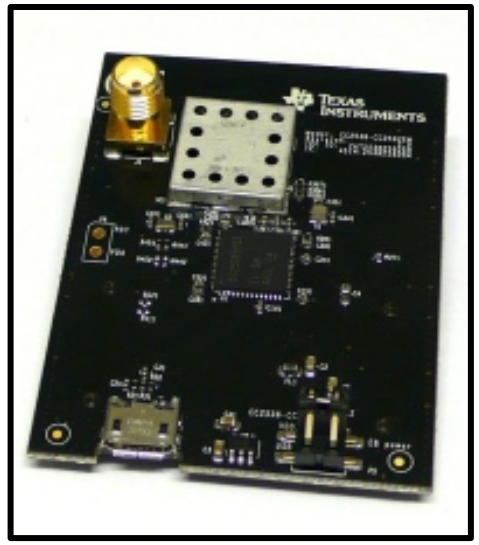
Lecture 3: Communication

Marco Zimmerling



Communication on CC2538 System on Chip (SoC)





Advanced high-performance bus (AHB)

Serial buses

USB 1.x (for programming)

Wireless radio

Goal of Today's Lecture

- Communication with peripherals: serial buses
- Communication between microcontrollers: interconnects
- Communication between devices: low-power wireless

Goal of Today's Lecture

- Communication with peripherals: serial buses
- Communication between microcontrollers: interconnects
- Communication between devices: low-power wireless

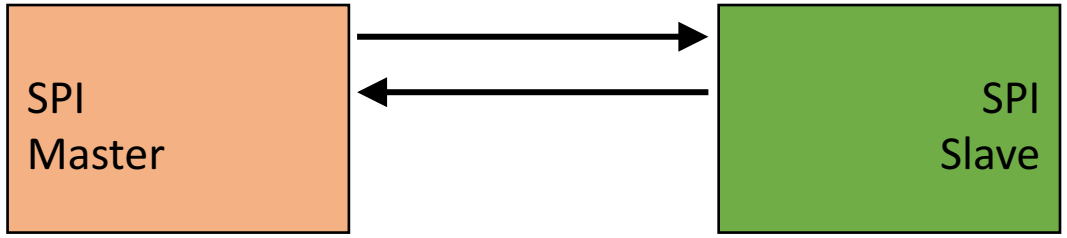
Serial Buses on an Embedded Platform

- Universal asynchronous receiver/transmitter (UART)
  - In embedded systems often realized via *bit banging*: setting and sampling pins and configuring signal timing, levels, and synchronization purely in software
  - A dedicated UART handles these tasks and provides (buffered) data transfers
- Serial peripheral interface (SPI)
  - Widely used to connect peripherals (*e.g.*, LCD, sensor, radio) to microcontroller
- Inter-integrated circuit (I^2C) – pronounced "I-squared-C"
  - Used to connect lower-speed peripherals (*e.g.*, power switch, access to low-speed analog-to-digital converters and clocks, configuring displays and speakers)
  - Requires fewer pins and signals than SPI (only two pins), but at most 3.4 Mbps
- Inter-IC sound (I^2S) – pronounced "I-squared-S"
  - Used for digital audio communication via pulse code modulation (PCM)

SPI: Overview

- Provides synchronous serial communication over short distances
  - Serial: One bit at a time rather than multiple bits in parallel
  - Synchronous: transmission of each bit is aligned with a common clock signal
- First described by Motorola in late 1980s (only hardware operation)
- No standard/patent/license, no definition of software protocol
  - Good: widely adopted, de-facto standard for connecting peripherals to a microcontroller (memory cards, real-time clocks, cameras, sensors, ADC/DAC)
  - Bad: lack of standard tools due to many different variants (*e.g.*, word sizes)
- Very simple hardware interfacing and software implementation

SPI: Capabilities

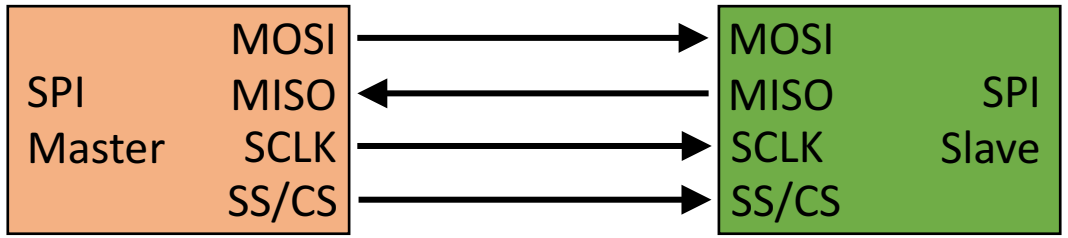


- Always *full-duplex communication*
  - Communication in both directions simultaneously
  - Transmitted (or received) data may not be meaningful
- High transmission speeds
  - 10s of MHz clock frequencies are not uncommon
  - SPI itself incurs no overhead (no in-band addressing, no error control), but connected peripheral device may do (*e.g.*, addressing), reducing the goodput
- *Master-slave architecture* with multiple slaves
  - Master selects slave before initiating transmission
- Typically 8-bit word size, but other sizes are also possible and common
  - Examples: 16 bit for touchscreen controllers, 12 bit for ADCs and DACs

SPI: Wiring and Responsibilities

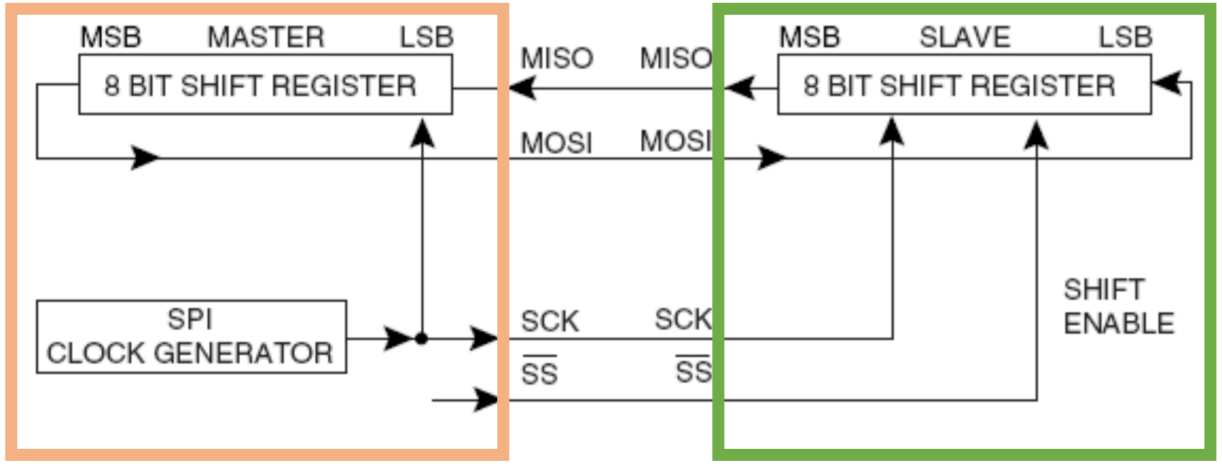
- Typically 4 wires
  - Master-Out, Slave-In (*MOSI*): carries data out of master to slave
  - Master-In, Slave-Out (*MISO*): carries data out of slave to master

  } *both are active during every transmission*
  - System Clock (*SCLK*): synchronizes data transfer
  - Slave Select/Chip Select (*SS/CS*): unique line to select each slave



- Responsibilities of master
  - Generate SCLK
  - Assert SS/CS line (*e.g.*, to select one out of multiple slaves)
  - Initiate data transfer (also if it only acts as receiver)

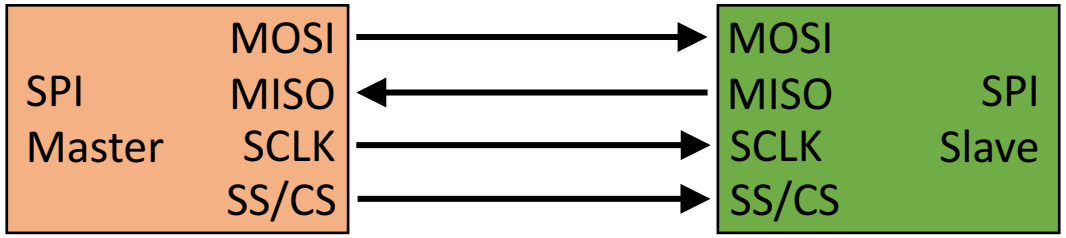
SPI: Data Transfer via Shift Registers



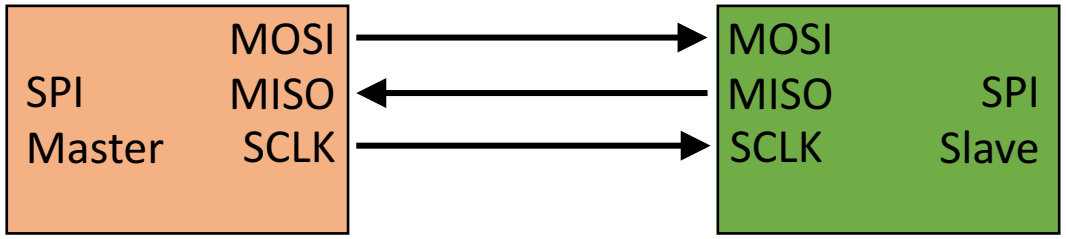
- One shift register per master or slave for both sending and receiving
- Master shifts out data to slave, and shifts in data from slave
  - Received data are available in the same register as transmitted data
- Typically the most-significant bit (MSB) comes first
  - Some microcontrollers (*e.g.*, Atmel AVR) allow to configure in software whether the MSB or the least-significant bit (LSB) comes first

SPI: Configurations with a Single Slave

- *4-wire master-slave*

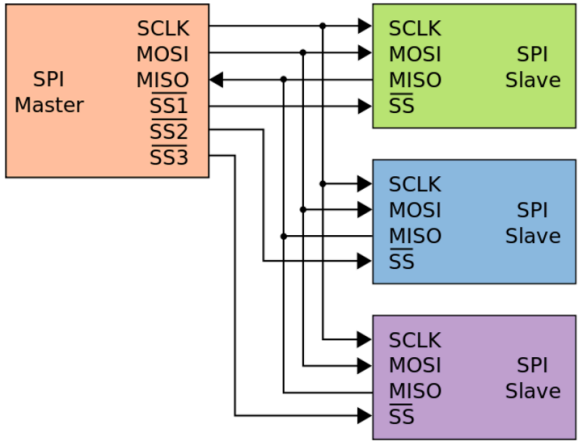


- *3-wire master-slave*
  - SS/CS connected to ground (*i.e.*, slave is always selected due to active-low)

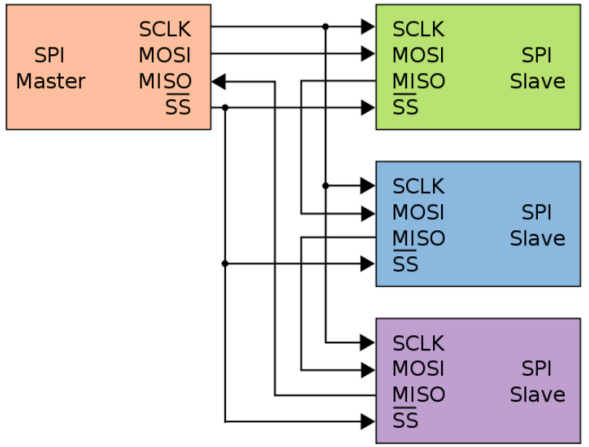


SPI: Configurations with Multiple Slaves

- Master and multiple *independent slaves*
  - Independent SS/CS line for each slave
  - MISO pins must be tristate
    - High (1), low (0), high-impedance (neither 0 nor 1)
  - Typical configuration



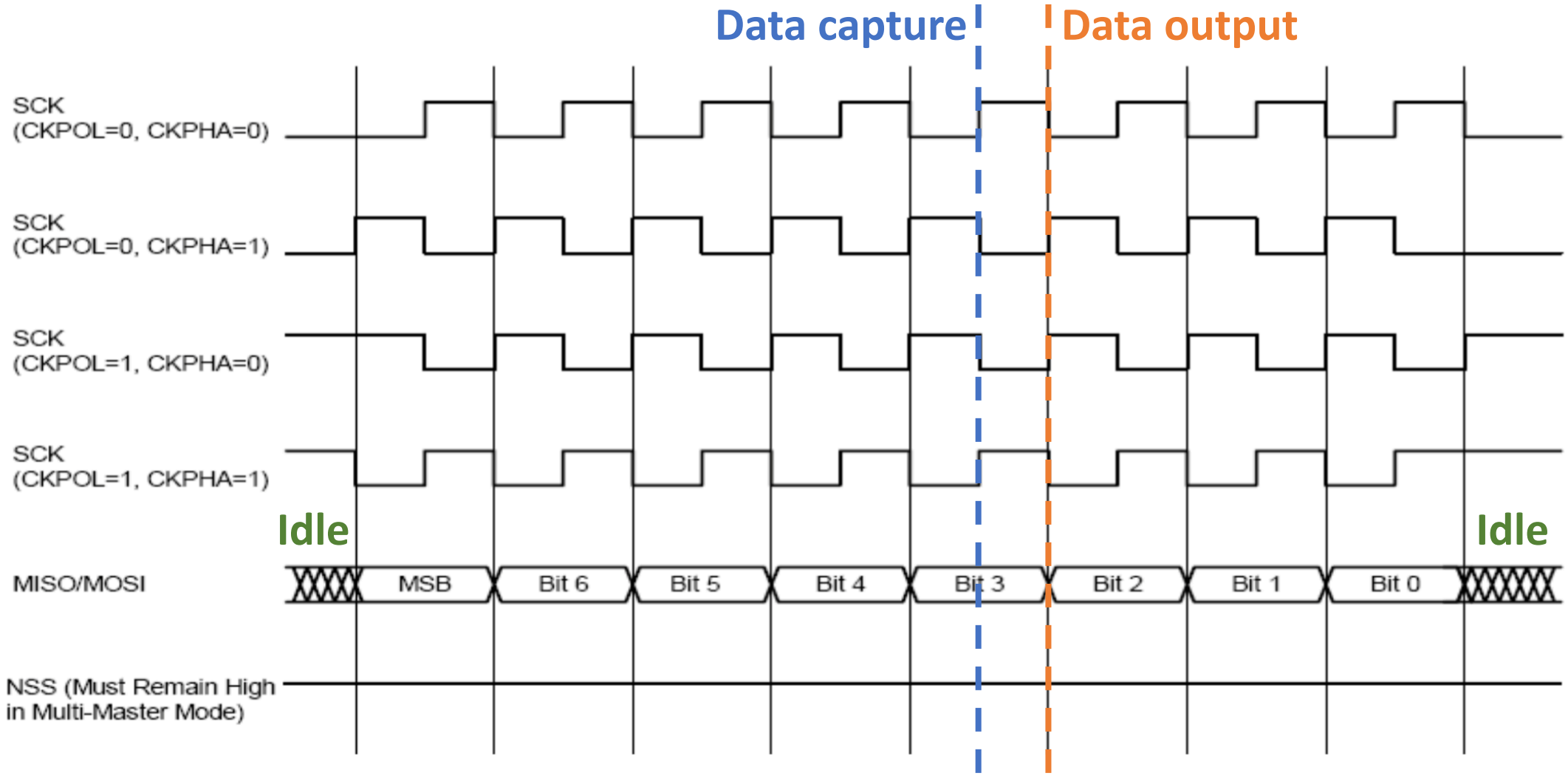
- Master and multiple *daisy-chained slaves*
  - First slave output connect to second slave input, etc.
  - Slaves send copy of received data, with a small delay
  - Example: JTAG for on-chip debugging



SPI: Modes

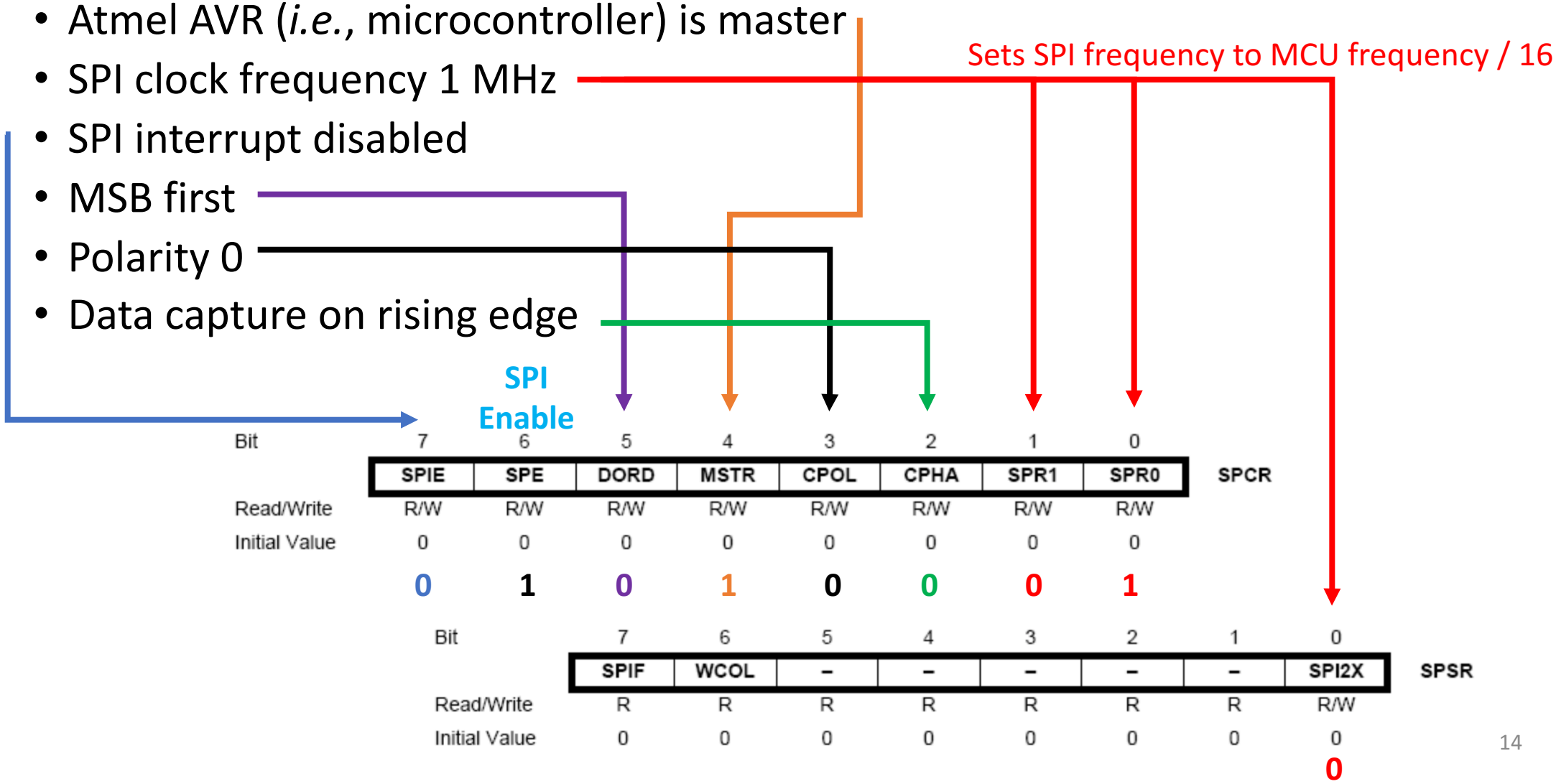
- Four modes determined by possible combinations of
  - *Polarity*, as determined by CPOL configuration bit
  - *Phase*, as determined by CPHA configuration bit
- CPOL = 0: clock signal is 0 in idle state and 1 in active state
  - CPHA = 0: data captured on rising edge and output on falling edge
  - CPHA = 1: data captured on falling edge and output on rising edge
- CPOL = 1: clock signal is 1 in idle state and 0 in active state
  - CPHA = 0: data captured on falling edge and output on rising edge
  - CPHA = 1: data captured on rising edge and output on falling edge

SPI: One Transfer Cycle for Different Modes



SPI: Example Configuration on Atmel AVR

- **Enable SPI** transfer on Atmel AVR running at 16 MHz such that
  - Atmel AVR (*i.e.*, microcontroller) is master
  - SPI clock frequency 1 MHz
  - SPI interrupt disabled
  - MSB first
  - Polarity 0
  - Data capture on rising edge



SPI: Pros and Cons

- Pros
  - Fast point-to-point communication (no defined maximum clock frequency)
  - Easily allows for streaming, even in both directions (arbitrary message size)
  - Simple software implementation (*e.g.*, no in-band addressing)
  - Simple hardware interfacing, typically lower power requirements than I^2C
  - Broadly supported
- Cons
  - Out-of-band “addressing” via SS/CS makes multiple slaves more complex
  - No acknowledgment by slave (master has no clue)
  - No flow control, so master must know slave speed (and adapt to it)
  - No error detection or recovery protocol defined

Goal of Today's Lecture

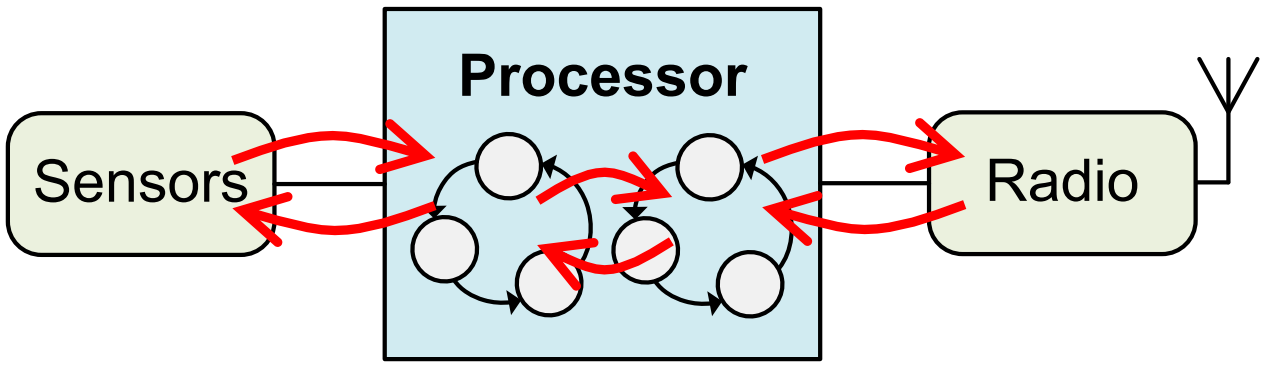
- Communication with peripherals: serial buses
- Communication between microcontrollers: interconnects
- Communication between devices: low-power wireless

Trend Toward Multi-processor Platforms (1)

- Real-world networked embedded applications become increasingly more sophisticated, featuring both heavy and light tasks.
- Characteristics of *heavy tasks* (*e.g.*, complex data preprocessing)
  - Highly performance demanding
  - Typically perform compute operations
  - Long but infrequently
- Characteristics of *light tasks* (*e.g.*, wireless communication)
  - Not performance demanding
  - Typically perform I/O operations
  - Short but frequently

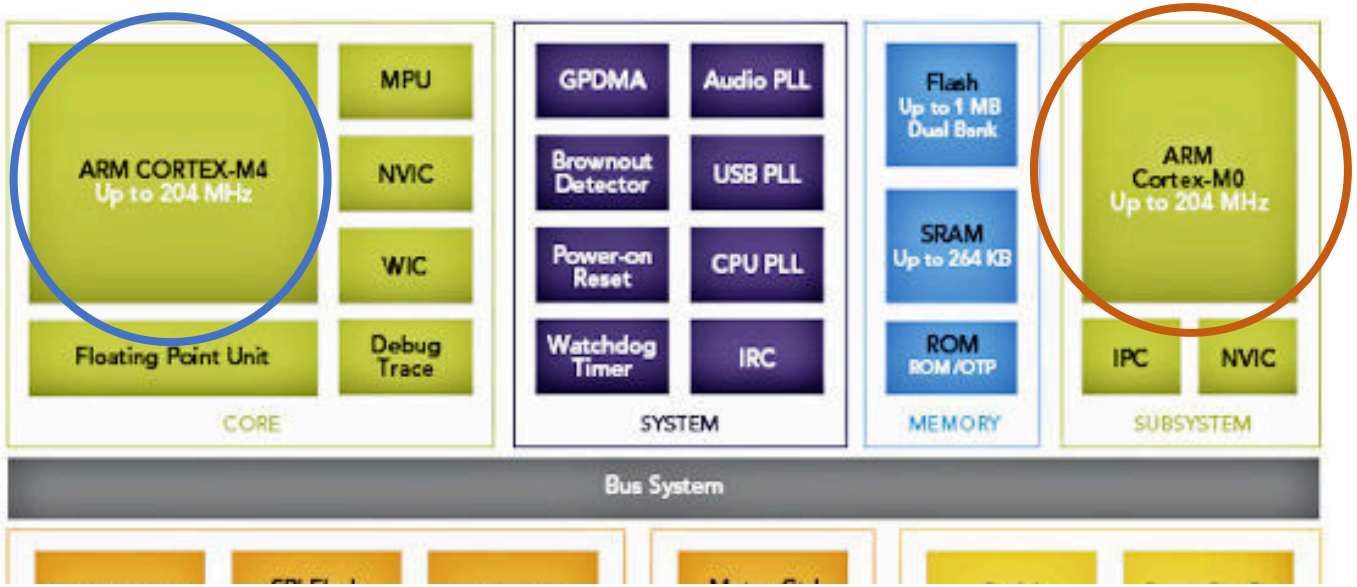
Trend Toward Multi-processor Platforms (2)

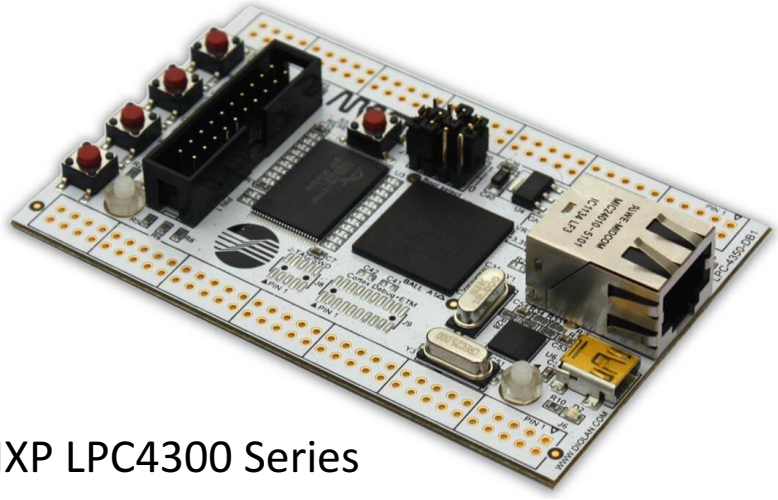
- Using a single powerful core that runs at high frequencies and has rich architectural features offers poor energy efficiency for light tasks.
  - *High energy overhead in entering and exiting active power state* (frequent light tasks make this overhead very prominent in the overall energy budget)
  - *High idle power* (light tasks are not compute-intensive, so core spends many short periods in idle state; not enough time to switch to lower-power state)
  - *Over-provisioned performance* (lowest possible frequency and active power are limited by architecture & fabrication process, still over-provisioning to light tasks)
- Moreover, it is *extremely difficult to provide real-time guarantees*, especially when tasks are triggered by unpredictable external events.



Trend Toward Multi-processor Platforms (3)

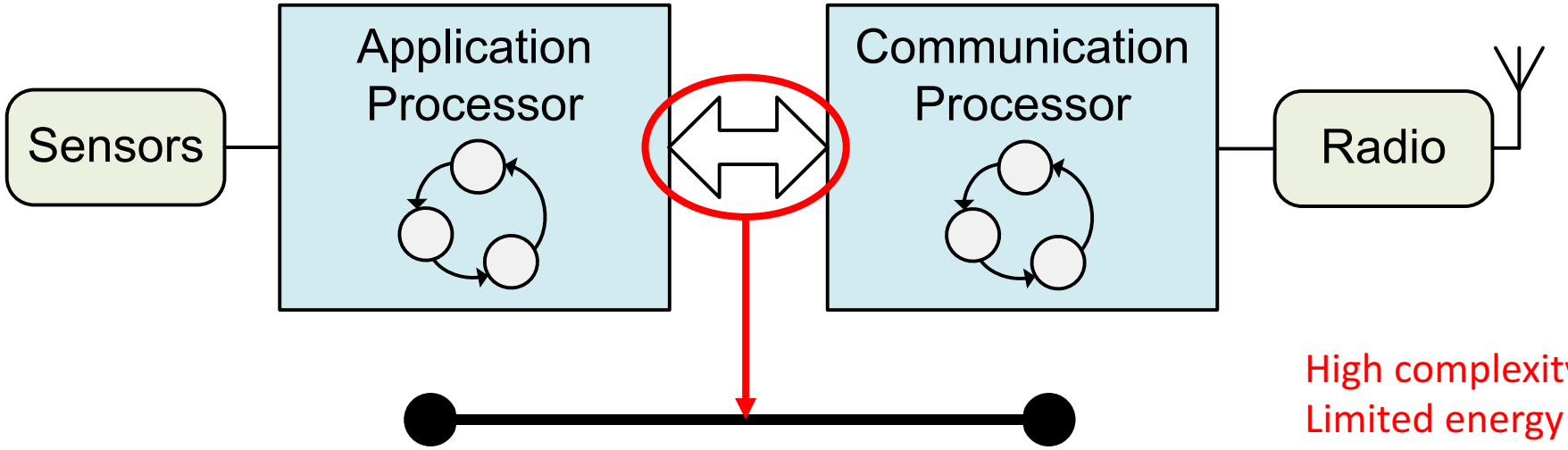
- Exploit *hardware heterogeneity* for best performance-power tradeoff
  - High-performance, high-power core for heavy tasks (*e.g.*, high-rate sensor data acquisition and compute-intensive local preprocessing)
  - Low-performance, low-power core for light tasks (*e.g.*, executing wireless communication stack and interacting with the radio chip)





NXP LPC4300 Series

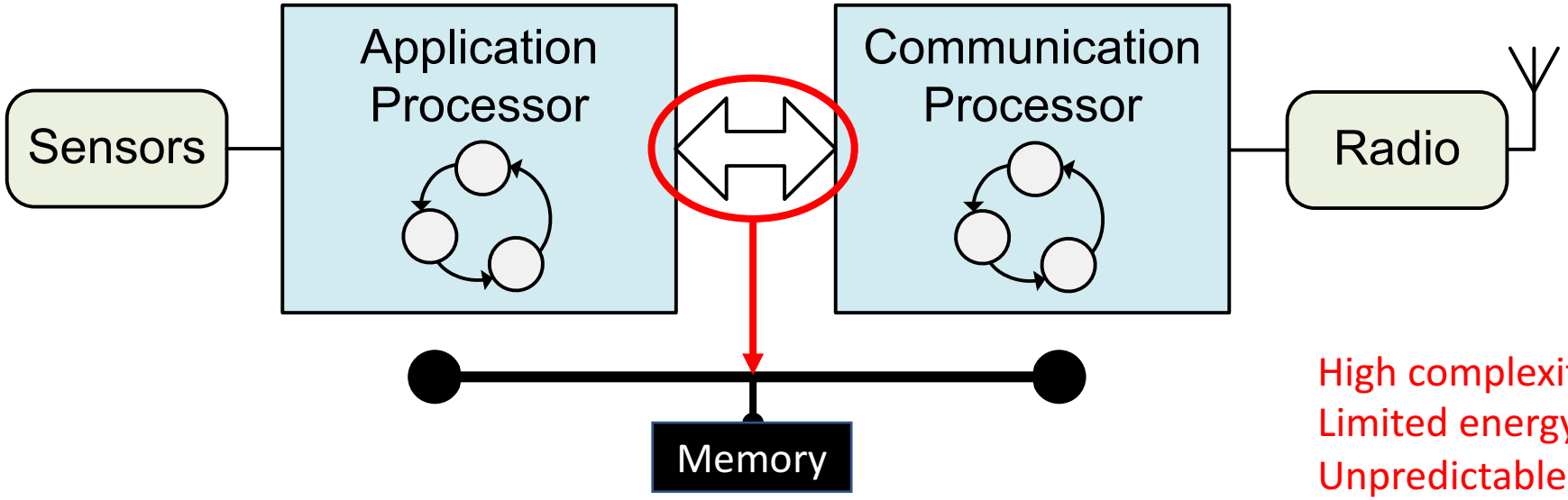
Interconnect: Shared Bus (*e.g.*, AHB, SPI, I^2C)



High complexity
Limited energy efficiency
Unpredictable timing

- Each processor must manage communication state locally
- Couples processors in time, power, and clock domains
  - Time: both need to engage in communication at the same time (synchronous)
  - Power: both need to be in active power state at the same time
  - Clock: both need to agree on a common clocking frequency

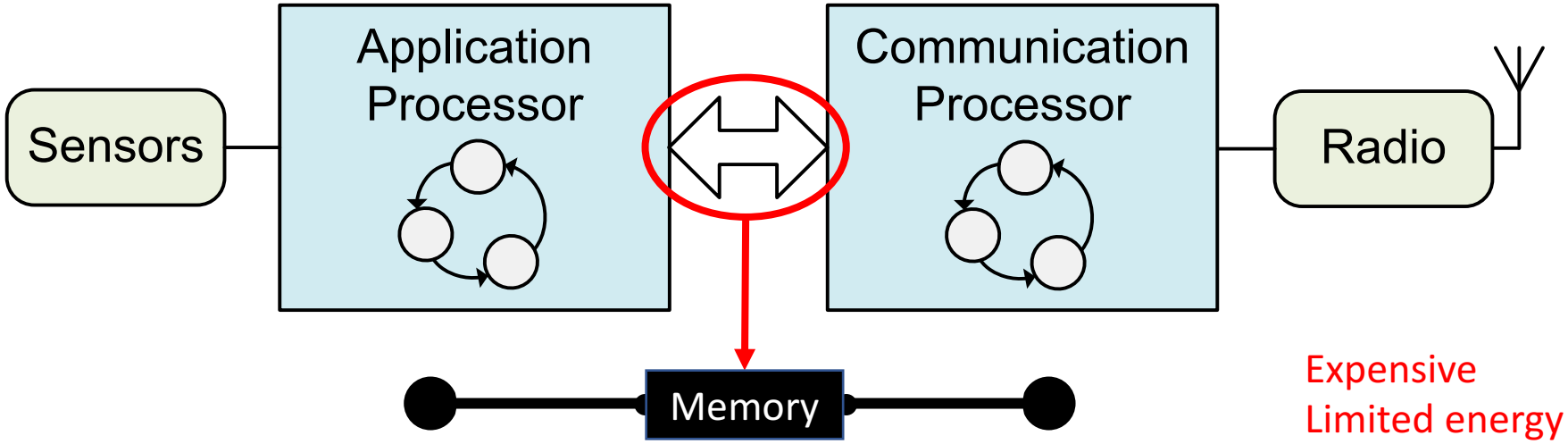
Interconnect: Shared Memory



High complexity
Limited energy efficiency
Unpredictable timing

- Each processor must manage communication state locally
  - Semaphores to handle concurrent reads and writes on shared variables
  - Spinlocks make execution time of reads and writes highly non-deterministic
- Still couples processors in time, power, and clock domains

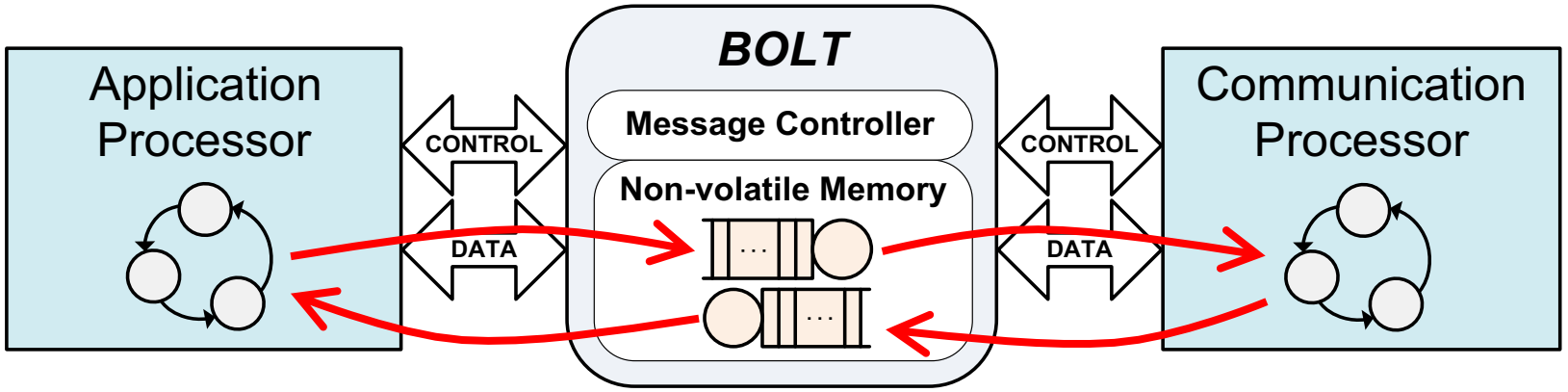
Interconnect: Dual-ported Memory



Expensive
Limited energy efficiency

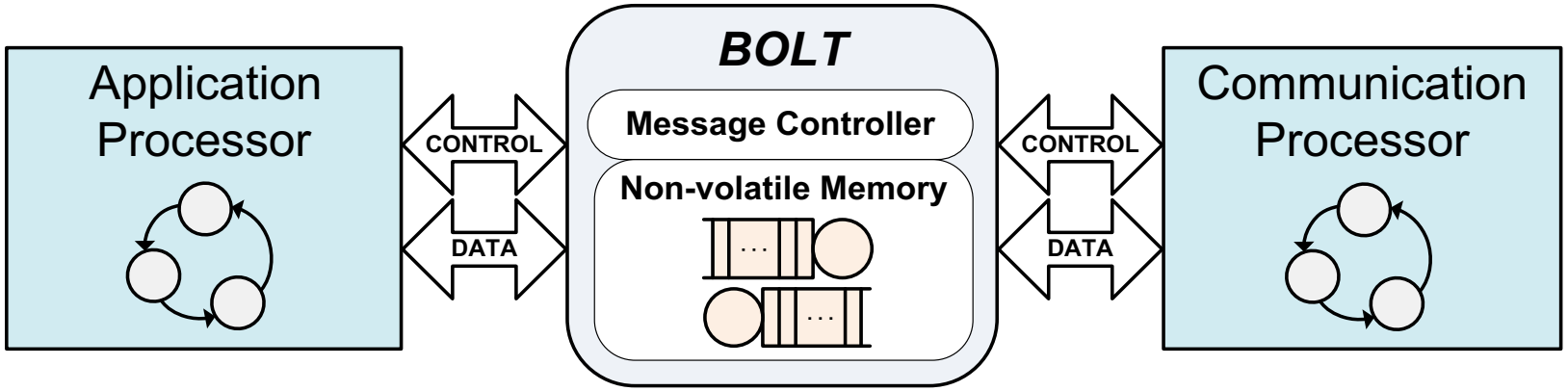
- Supports nearly current reads and writes on shared variables
- Typically expensive (extra circuitry) and based on volatile memory technology (*e.g.*, VRAM, SRAM), so needs to be powered all the time
- Still couples processors in clock domain

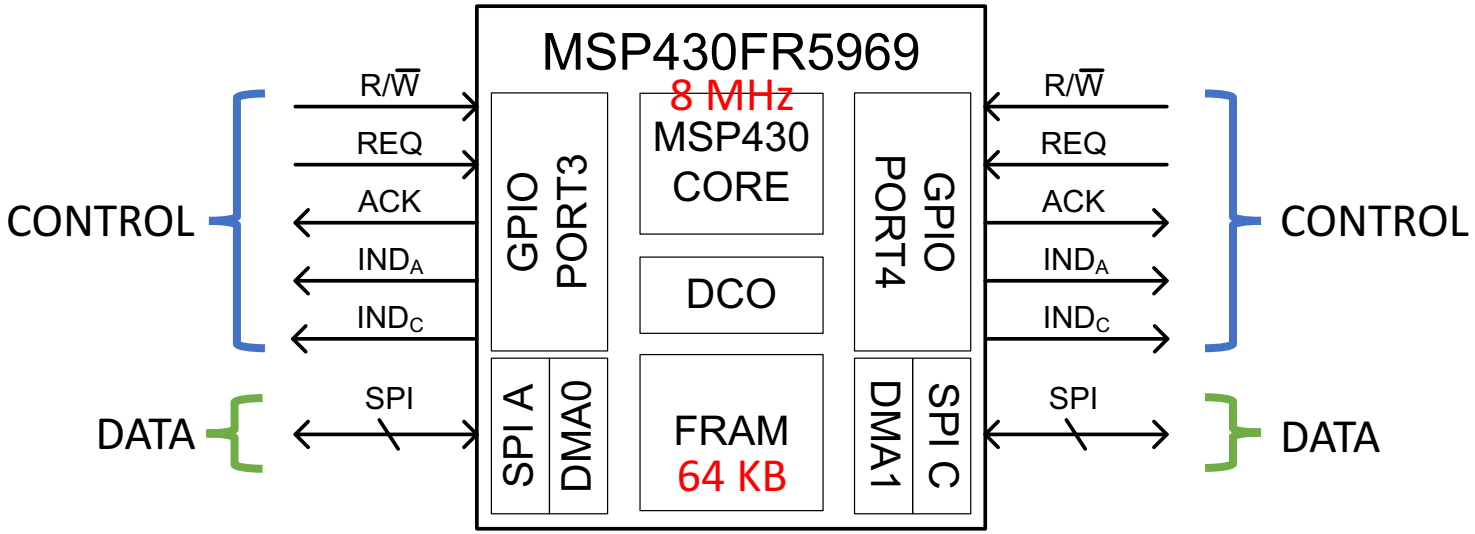
Bolt: Stateful Processor Interconnect



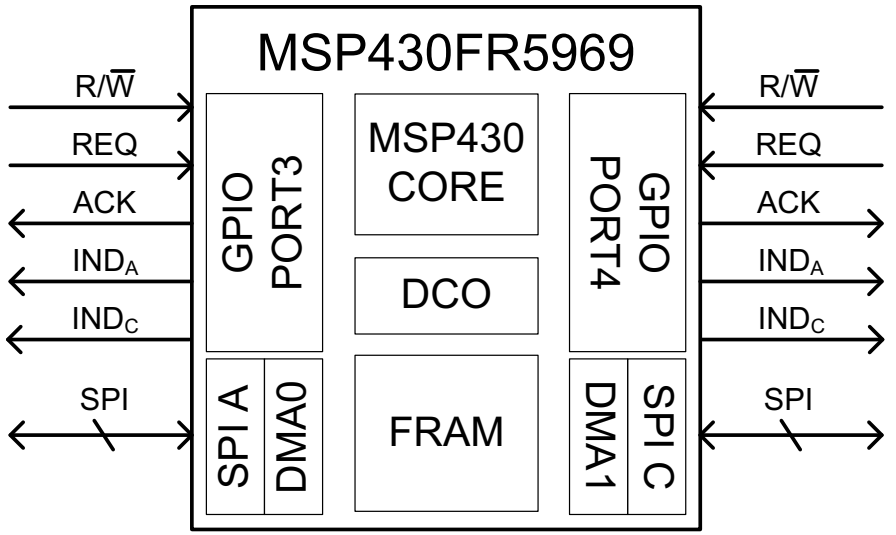
- *Decoupling* through asynchronous message passing (FIFO semantics)
  - Time: each processor can independently read and write messages
  - Power: each processor can independently switch power states
  - Clock: each processor can transfer messages at its preferred clock frequency
- *Predictability* through bounded execution times of reads and writes
- *Composability* in terms of both hardware and software components
  - Using several components together does not change their individual properties

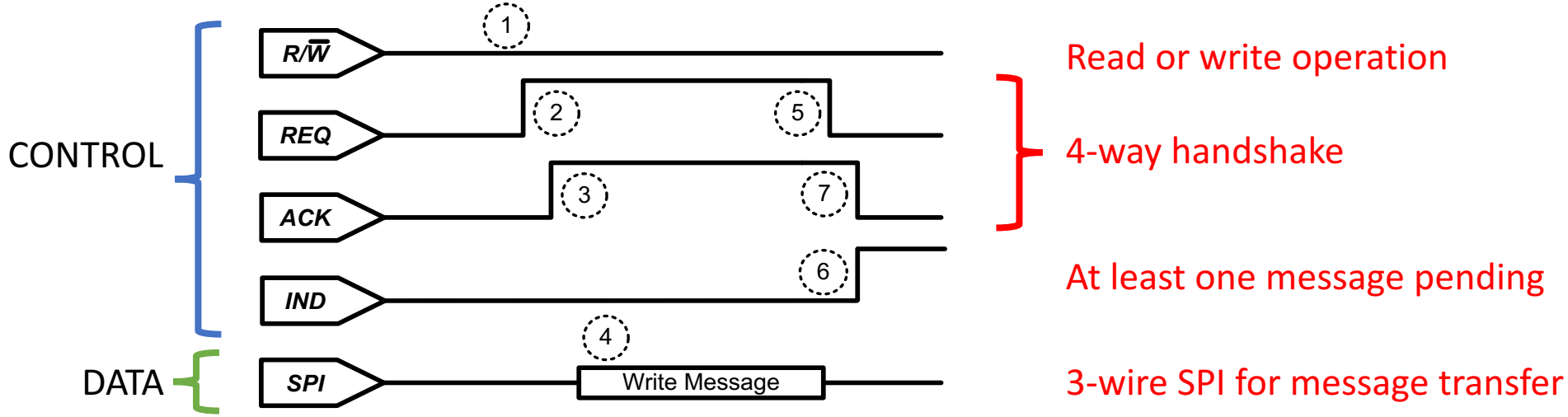
Bolt: Avoiding Bus Contention



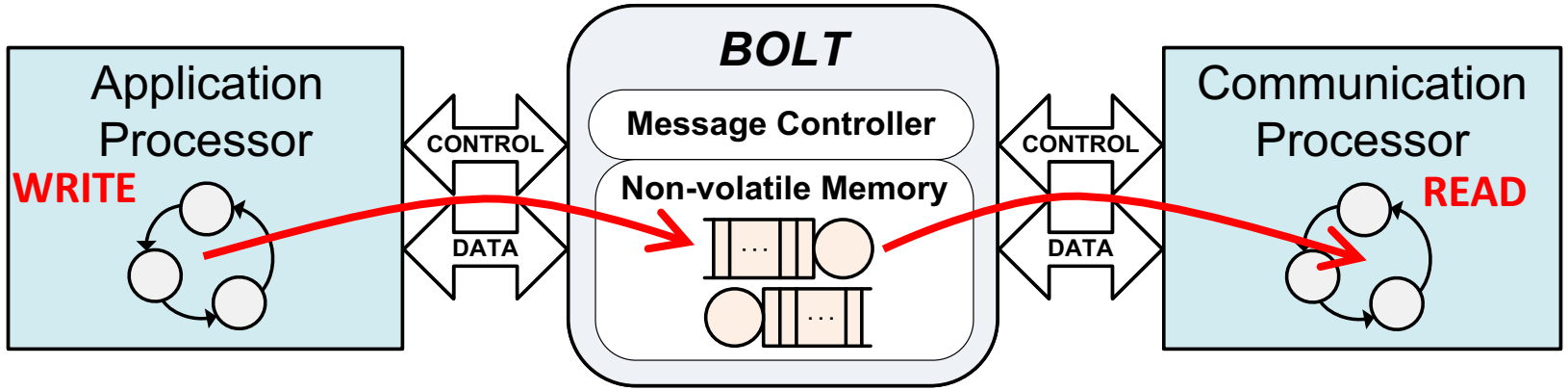


Bolt: Asynchronous Message Passing



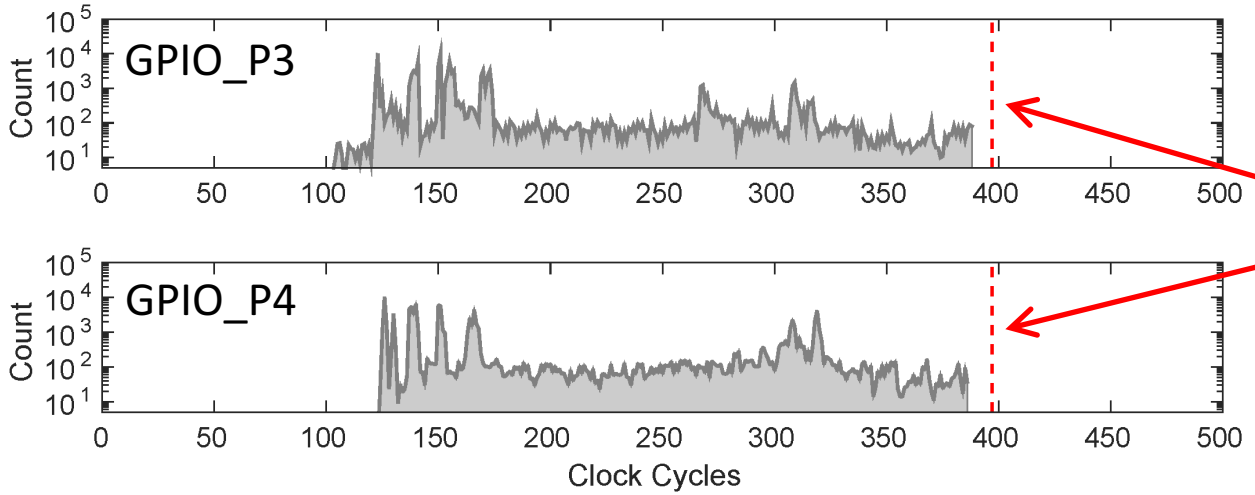


Bolt: Bounding Unavoidable Interference



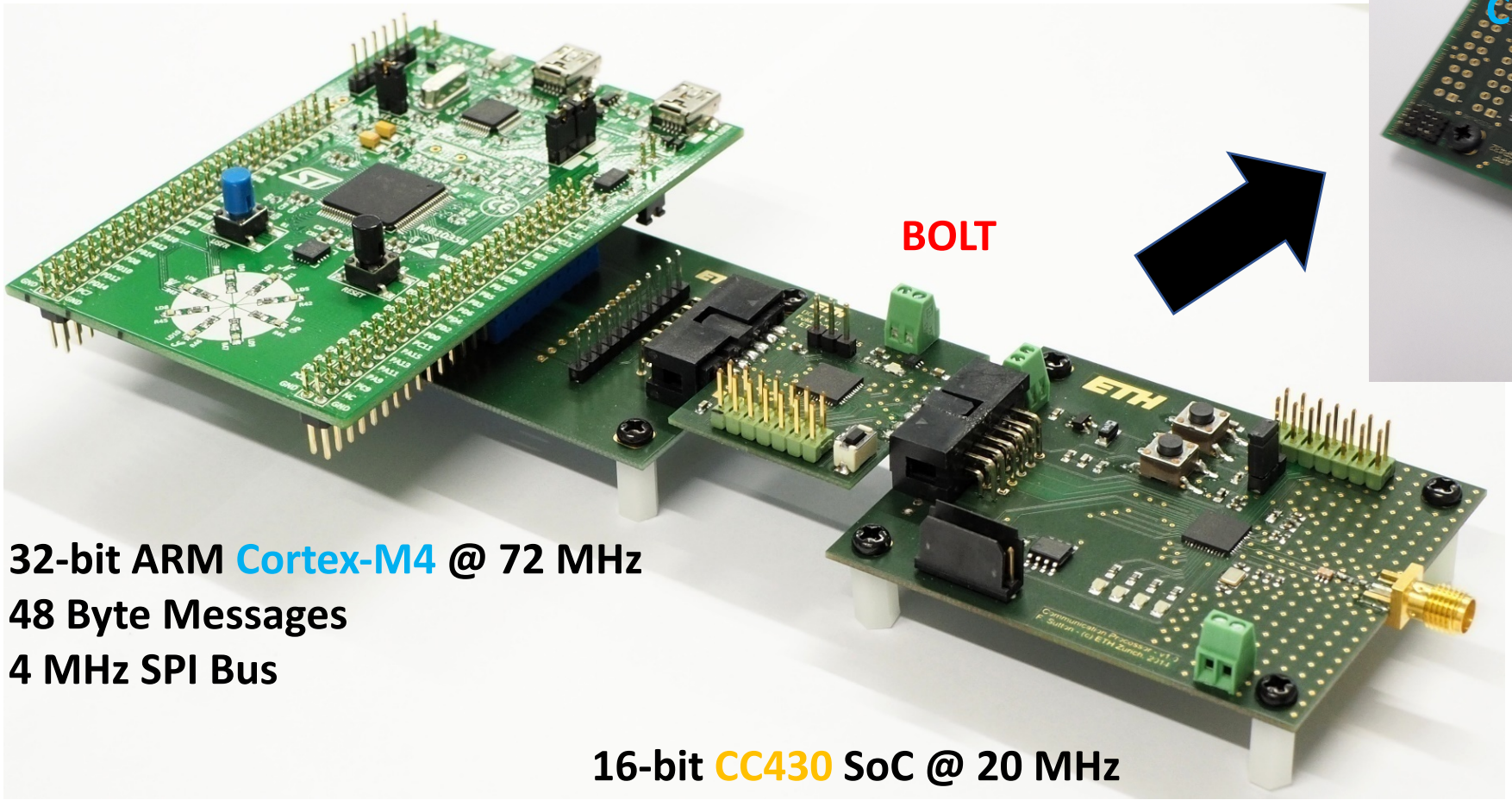
- Concurrent message operations cause unavoidable interference, but it is still possible to tightly bound the execution time of reads and writes.

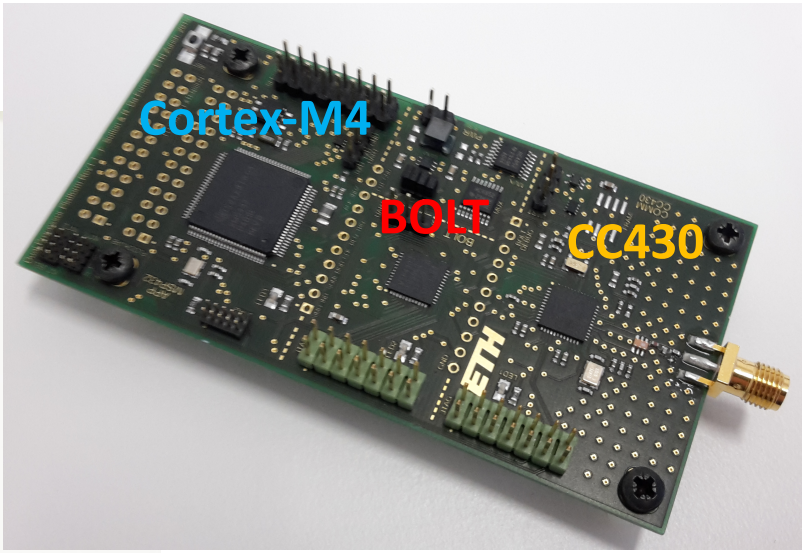
Execution time measurements



Worst-case execution times determined through formal analysis (model checker)

From Prototype to Dual-processor Platform (DPP)





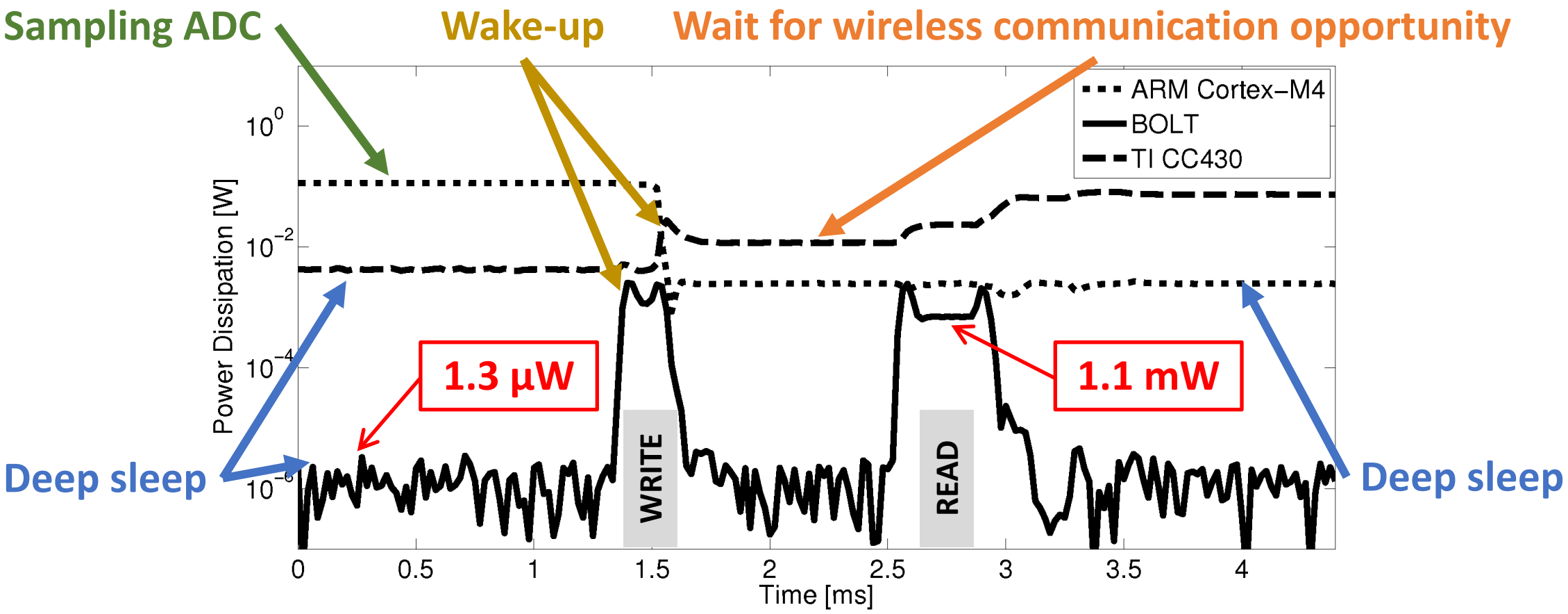
32-bit ARM Cortex-M4 @ 72 MHz
48 Byte Messages
4 MHz SPI Bus

16-bit CC430 SoC @ 20 MHz
24 Byte Messages
2 MHz SPI Bus

Up to 3.3 Mbps throughput

http://www.bolt.ethz.ch/

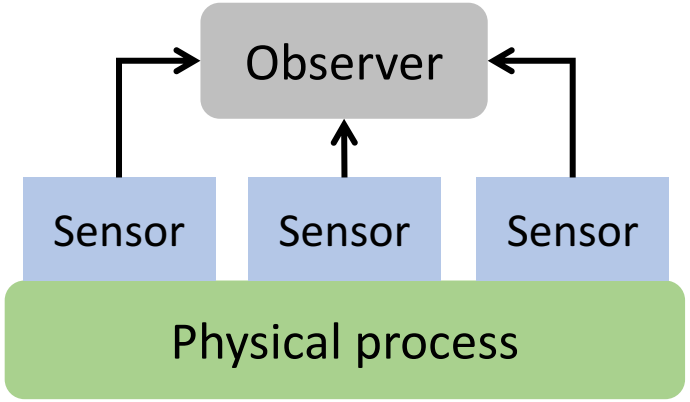
Bolt: Power Decoupling

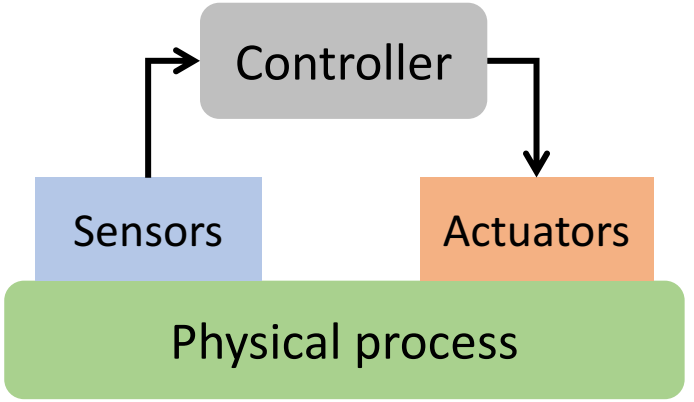


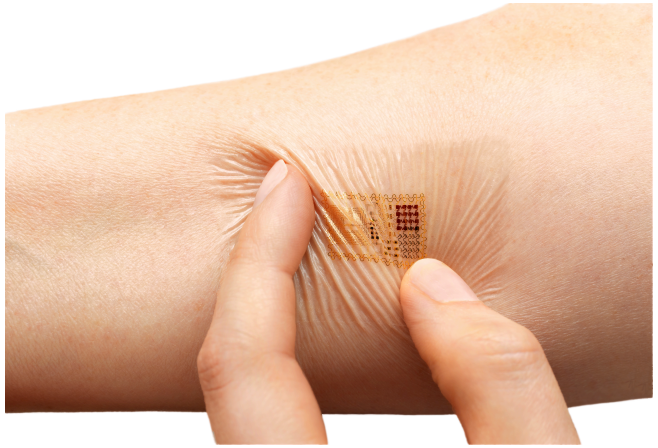
Goal of Today's Lecture

- Communication with peripherals: serial buses
- Communication between microcontrollers: interconnects
- Communication between devices: low-power wireless

Low-power Wireless Application Space







Monitoring / Data collection
("sense and send")

Cyber-physical systems (CPS)
("sense and react")

Real deployments reported in the academic literature

~2002

~2012

time

Application Requirements

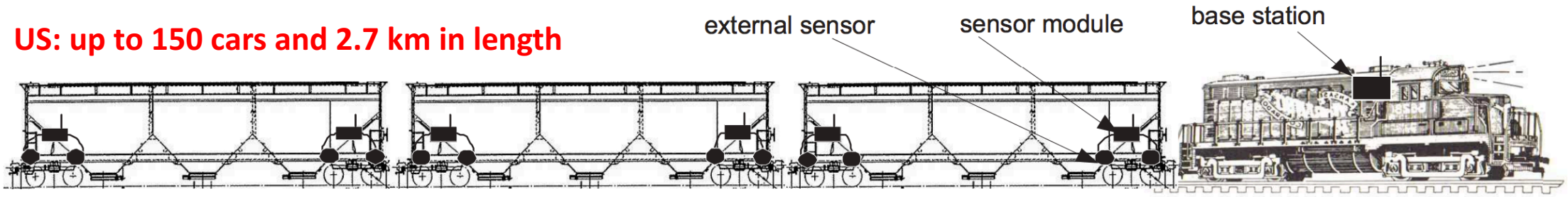
- Long *network lifetime*
  - Typically defined as time until the first node runs out of energy
  - Need to operate autonomously for several years

- High *end-to-end reliability*
  - Defined as number of packets received over those sent (packet delivery ratio)
  - Enable accurate data analyses and/or control decisions

- Low and/or bounded *end-to-end latency*
  - Defined as time needed to transmit packet from source to destination
  - Enable timely data analyses and/or real-time control decisions

Example: Freight Railroad Train Monitoring

US: up to 150 cars and 2.7 km in length



- Goal: timely visibility into health and status of trains to improve safety and resource utilization while reducing maintenance costs
  - Derailments due to cracked wheels are a major concern
  - More than 1.4 million railroad cars in the US, most without on-board power
  - Use wireless sensors to detect or prevent cracks (predictive maintenance)
- Network lifetime at least as long as regular maintenance cycle (5 years)
- End-to-end reliability and latency directly impact application utility

Important Application Characteristics

- Communication pattern

one-to-one
(data query)

many-to-one
(data collection)

one-to-many/-all
(reprogramming)

many-to-many
(distributed control)

- Traffic pattern

periodic and fixed
(environmental monitoring)

periodic but changing
(habitat monitoring)

aperiodic
(event detection)

- Mobility pattern (○ static node, ◌ mobile node)

all nodes static

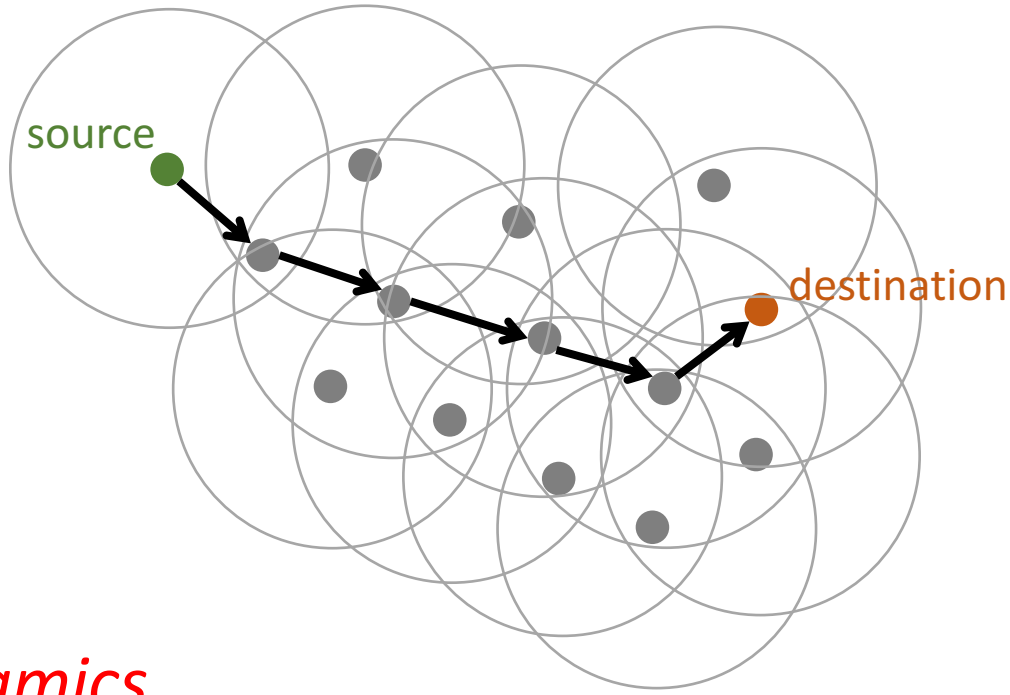
mobile sources

mobile sink(s)

all nodes mobile

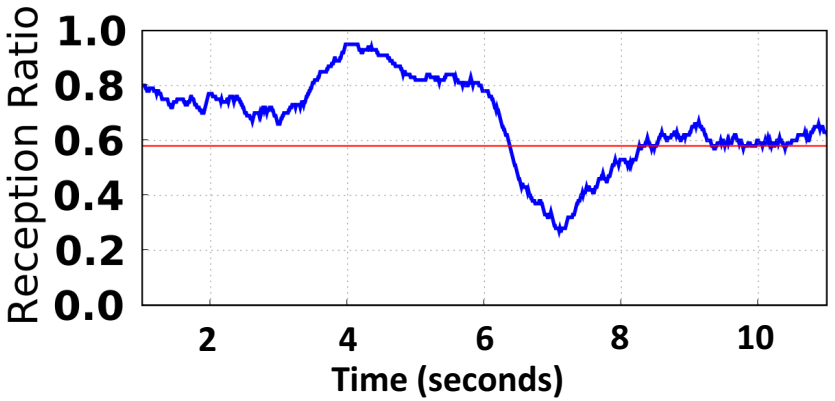
Challenges in Low-power Wireless

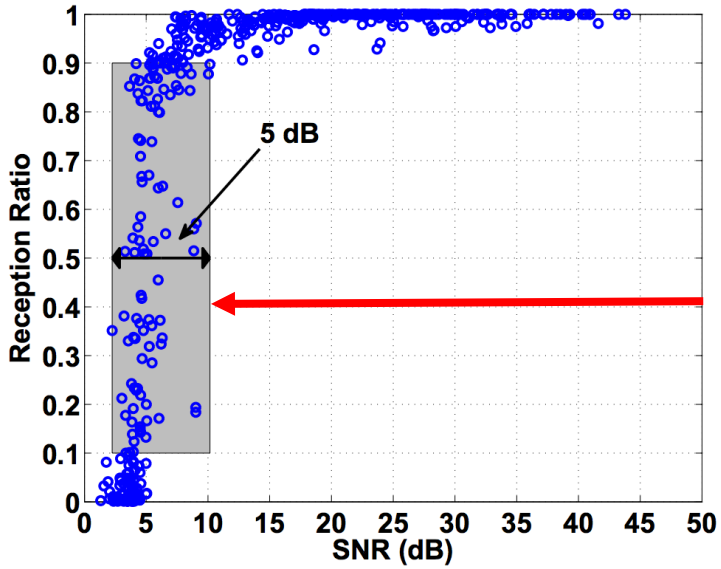
- Limited resources
  - Radio: 250 kbps
  - MCU: 8-32 bit, 4-72 MHz
  - Memory: 4-32 kB RAM
  - Energy: batteries and/or harvesting
- *Multi-hop communication*
  - Because radio range is <100 m
- *Unpredictable link and network dynamics*
  - Fading (variation in signal attenuation due to multi-path or obstacles)
  - Interference (*e.g.*, from Wi-Fi, microwaves, baby monitors)
  - Node mobility and/or failures
  - Environmental factors (*e.g.*, varying ambient temperature)



Dynamics of Low-power Wireless Links

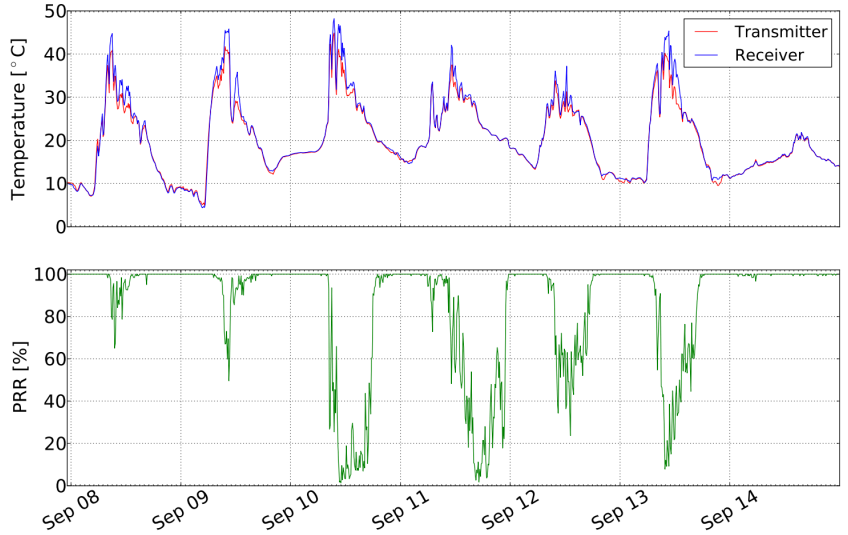
Links are only *stable for a short time* (*i.e.*, 10s of milliseconds)

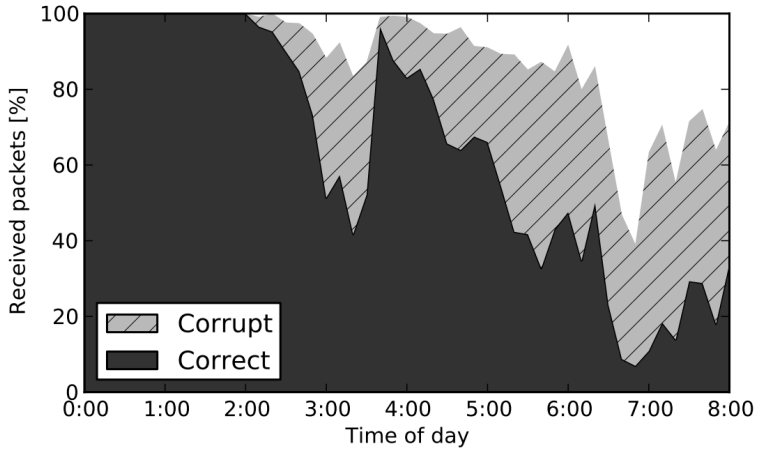




Intermediate links have highly varying packet reception rate between 10% and 90%

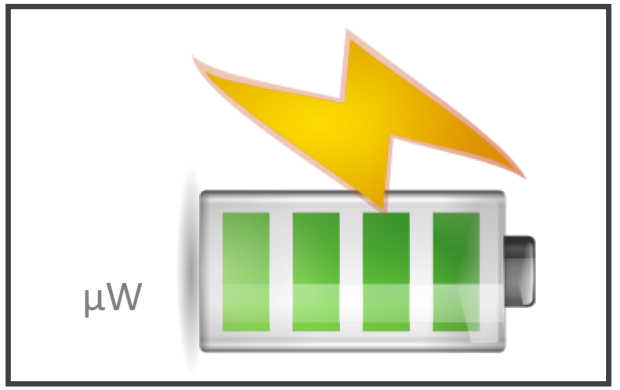
During the day, when temperature is high, *packet reception rate decreases significantly*





Communication Requirements

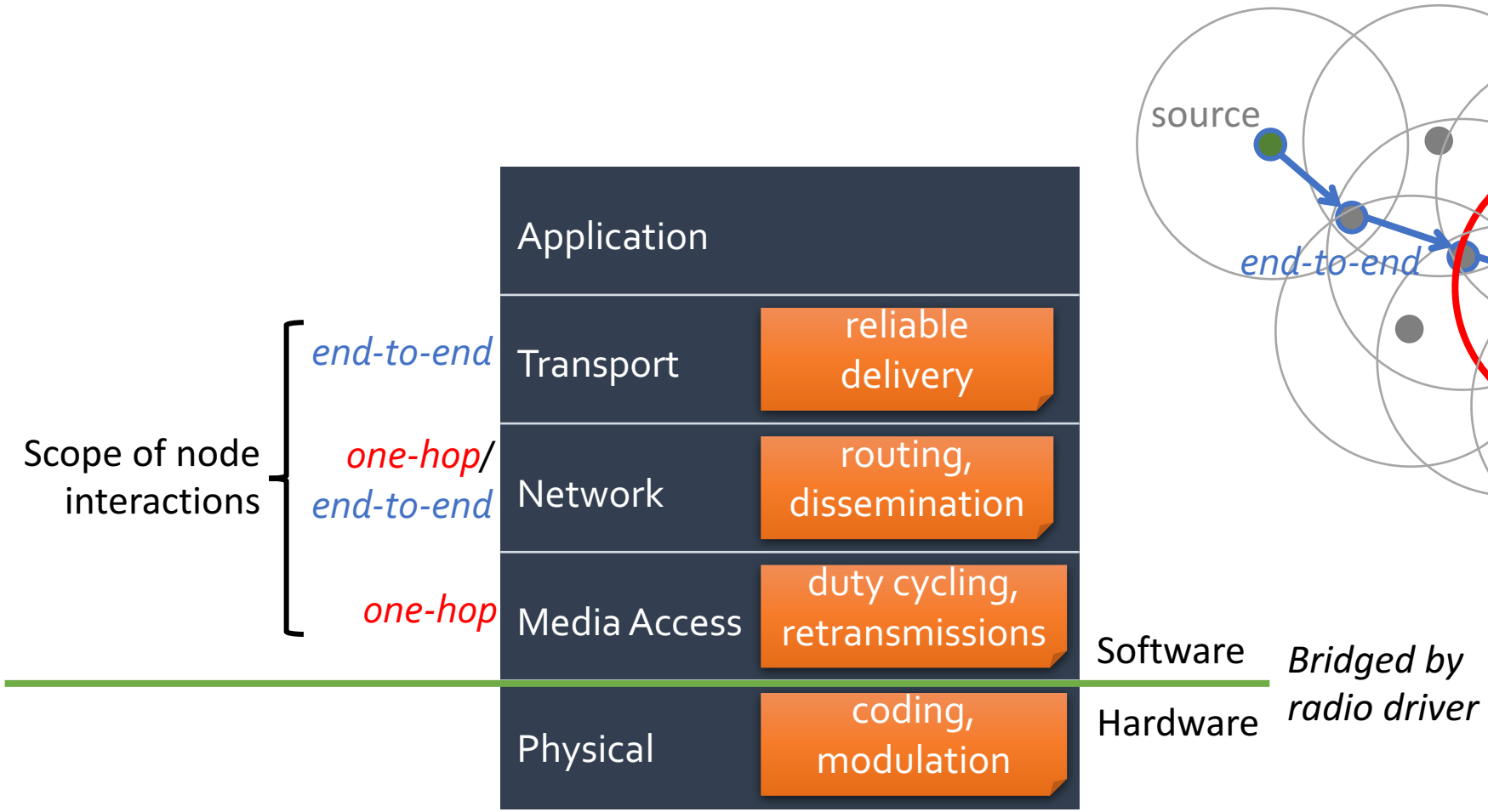
Predictable (in particular to support CPS applications)

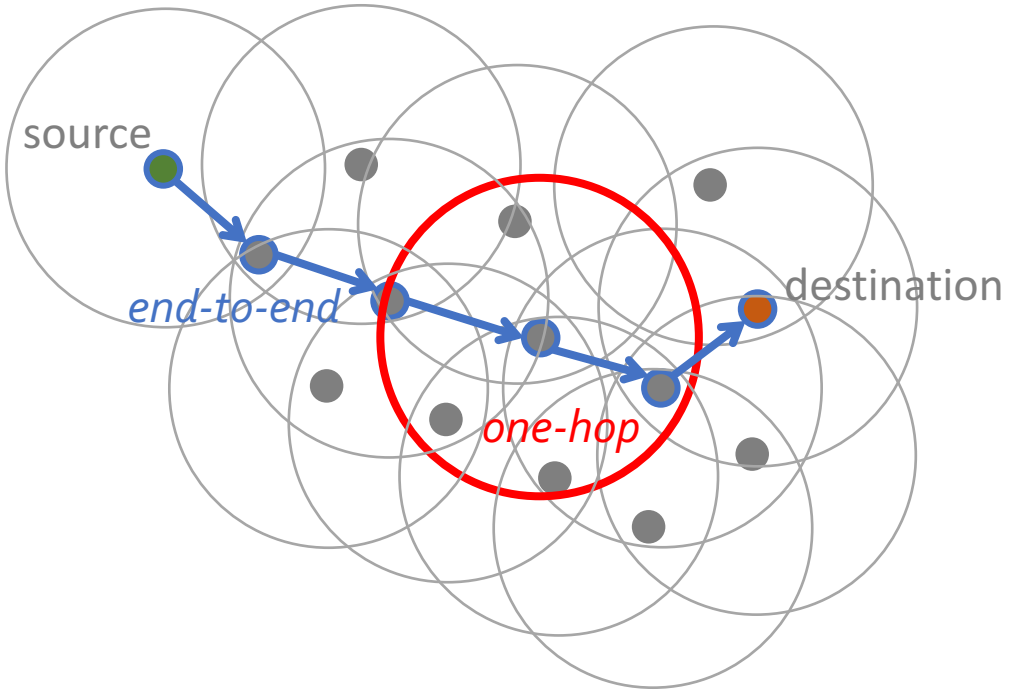


Efficient

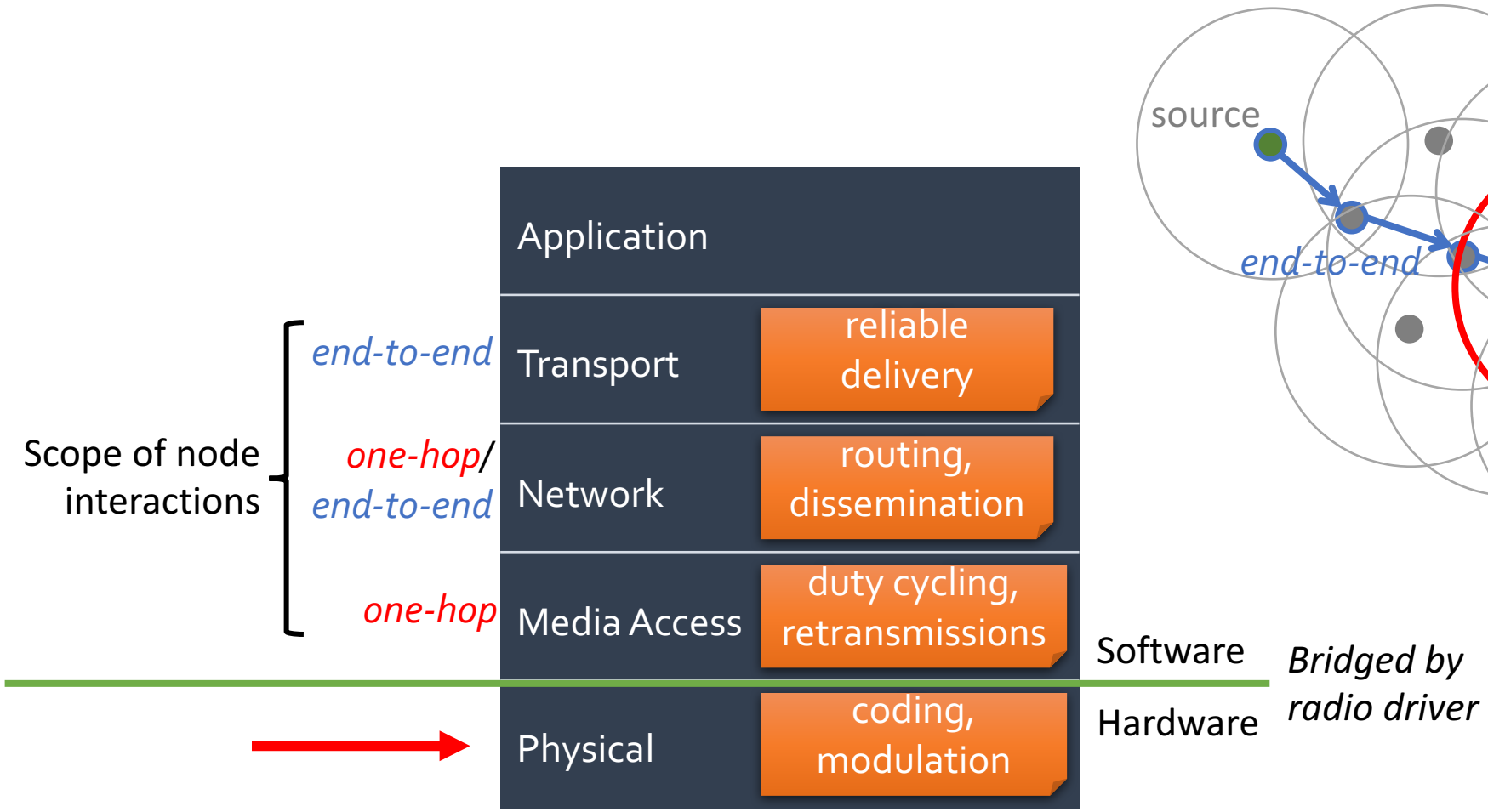
Versatile (support multiple communication patterns)
Adaptive (to changes in traffic pattern and network dynamics)

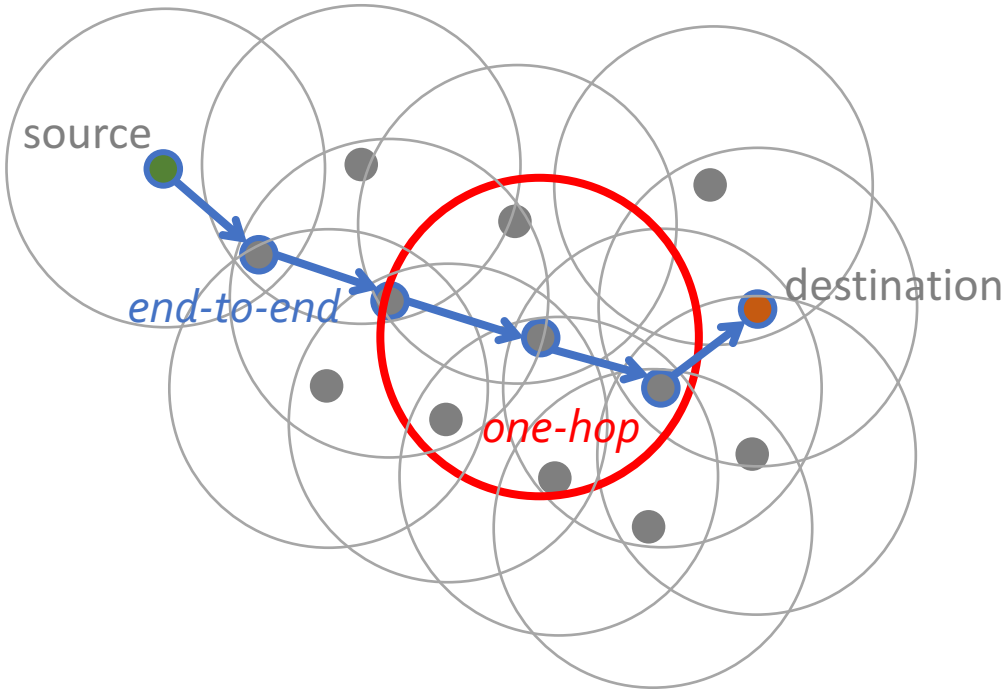
Low-power Wireless Communication Stack





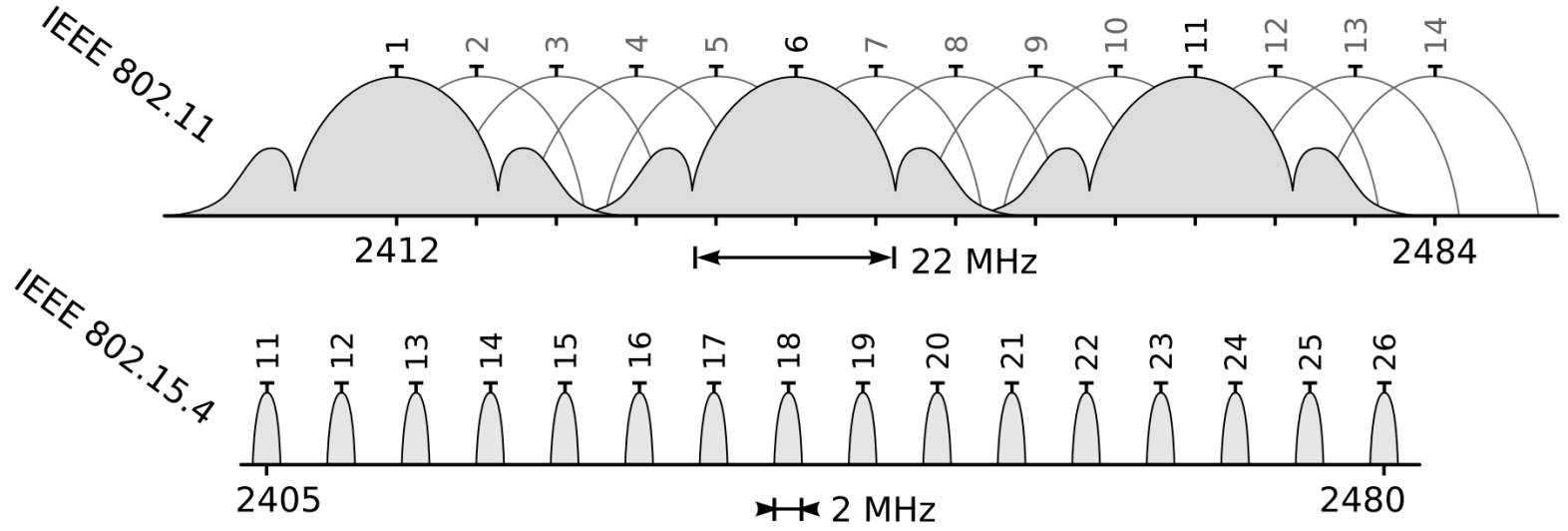
Low-power Wireless Communication Stack





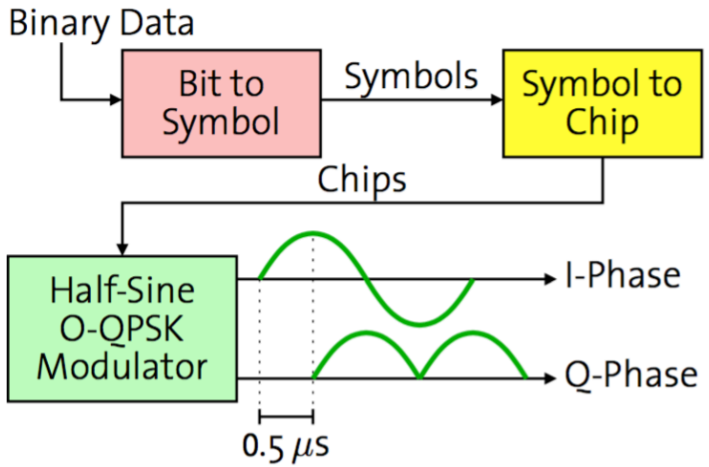
IEEE 802.15.4 Physical Layer (1)

- IEEE 802.15.4
  - Standard for low-rate wireless personal area networks, defined in 2003
  - Specifies physical and media access layers, but latter not widely adopted
  - Serves as basis for higher-layer standards (*e.g.*, ZigBee, 6LoWPAN)
- Most common 802.15.4 physical layer
  - 16 channels within 2.4 GHz industrial, scientific, and medical (ISM)
  - 250 kbps data rate, using DSSS and O-QPSK (see next slide)



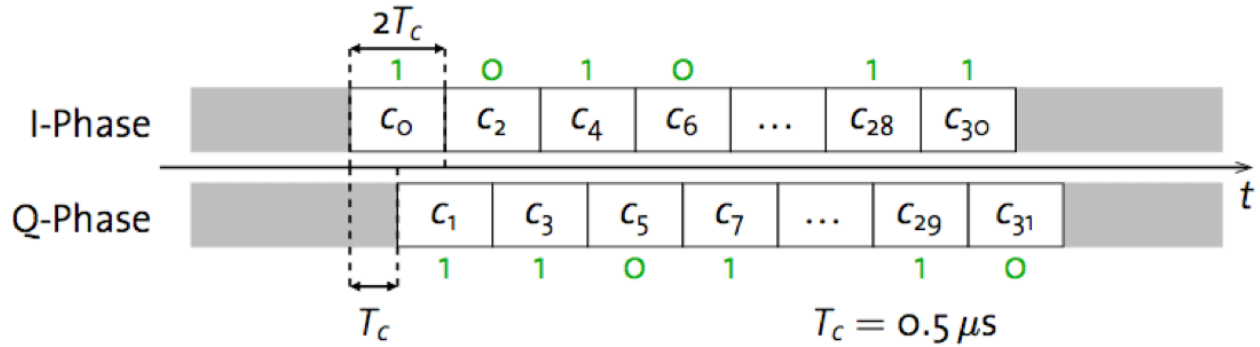
cross-technology interference

IEEE 802.15.4 Physical Layer (2)

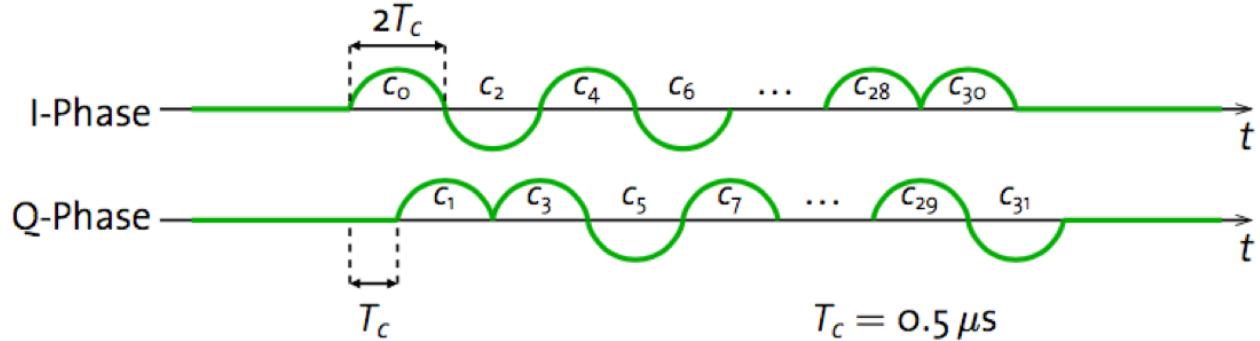


Data	Symbols	Chips ($c_0\ c_1\ \ldots\ c_{31}$)
0 0 1 0	4 →	0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1
0 1 1 1	14 →	1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0

Direct sequence spread spectrum (DSSS) adds redundancy, which helps recover correct symbol despite corrupted chips

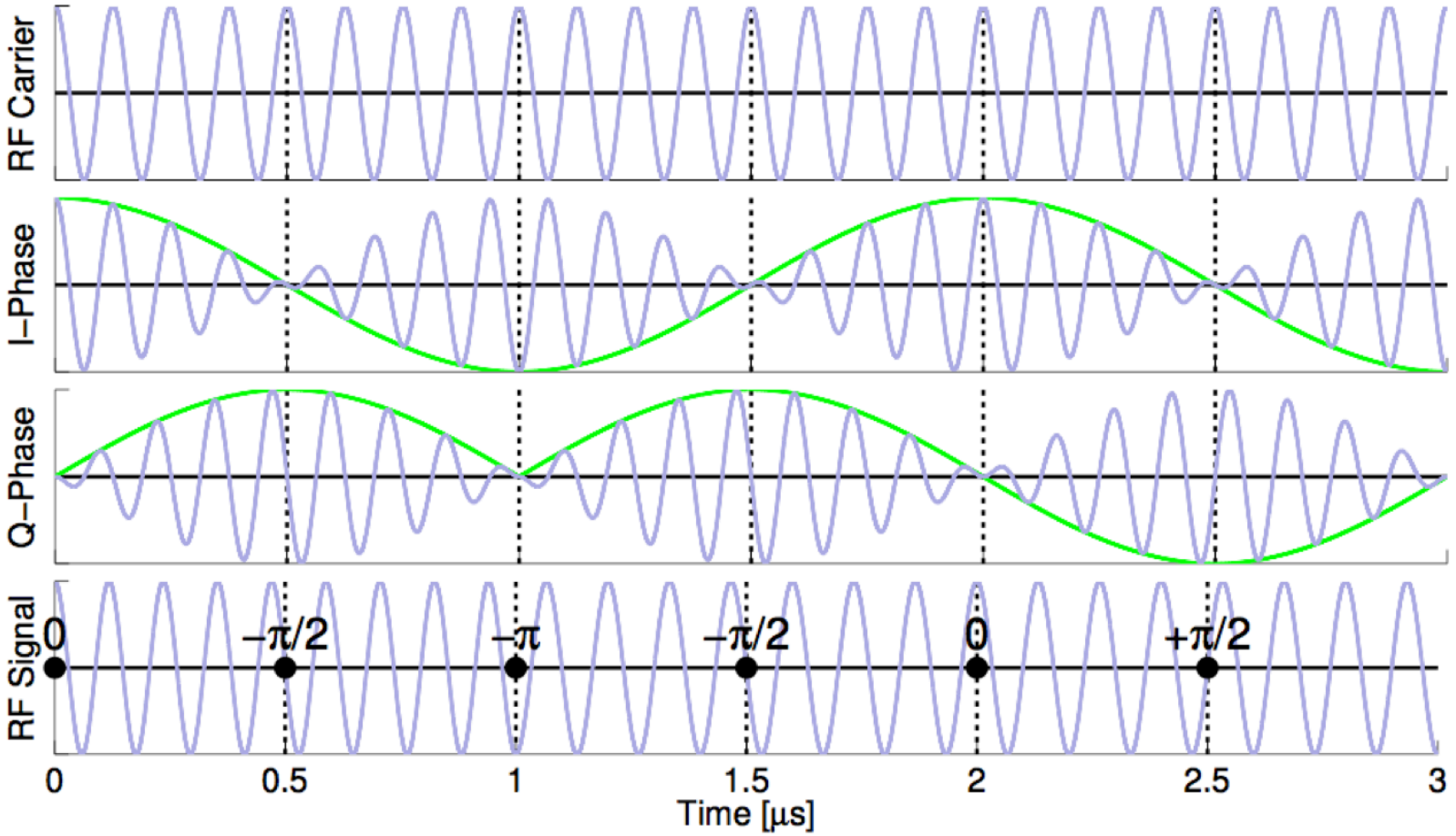


Even-indexed chips modulate *in-phase (I)*
Odd-indexed chips modulate *quadrature-phase (Q)*



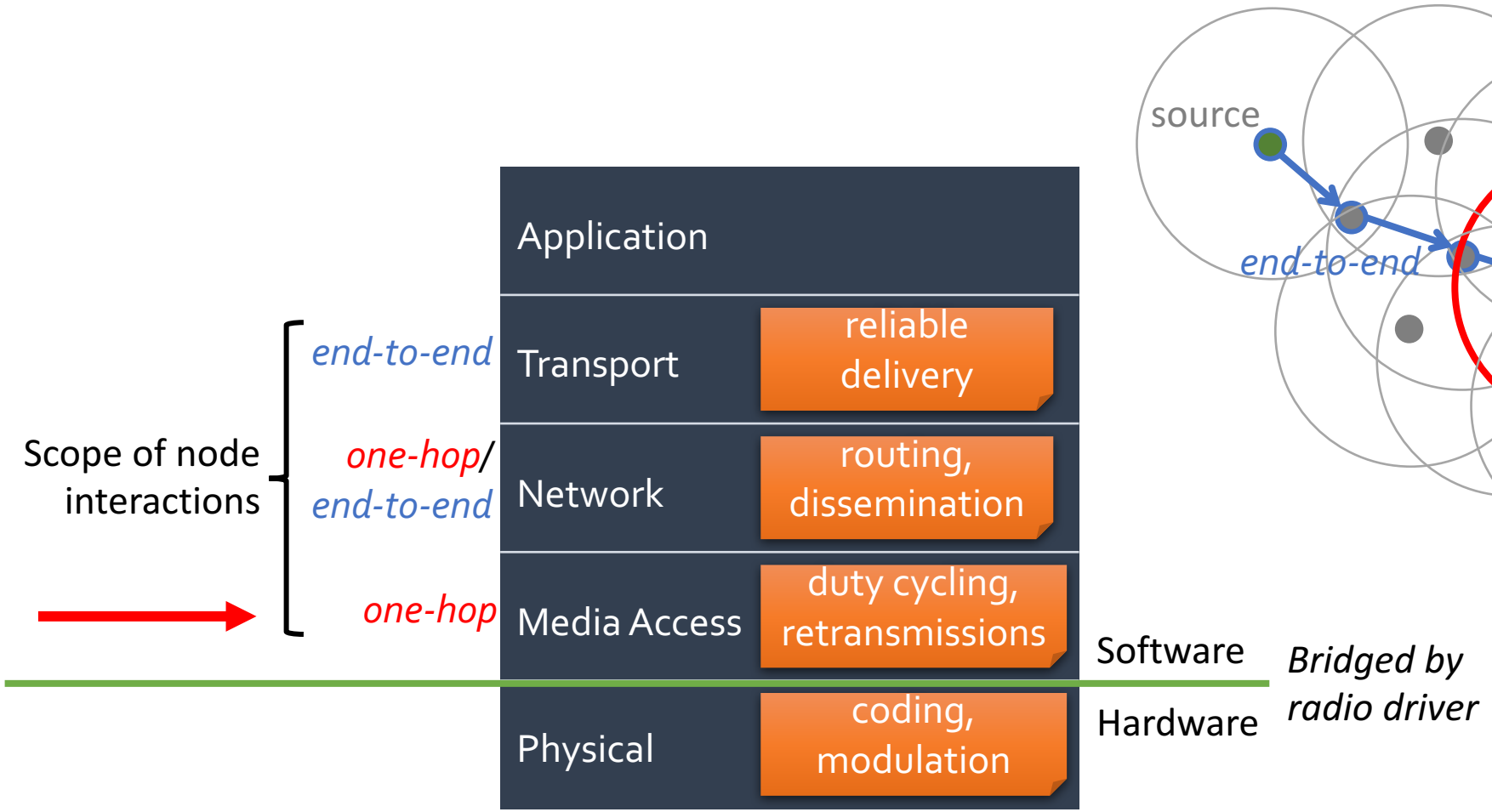
Each chip has a half-sine pulse shape

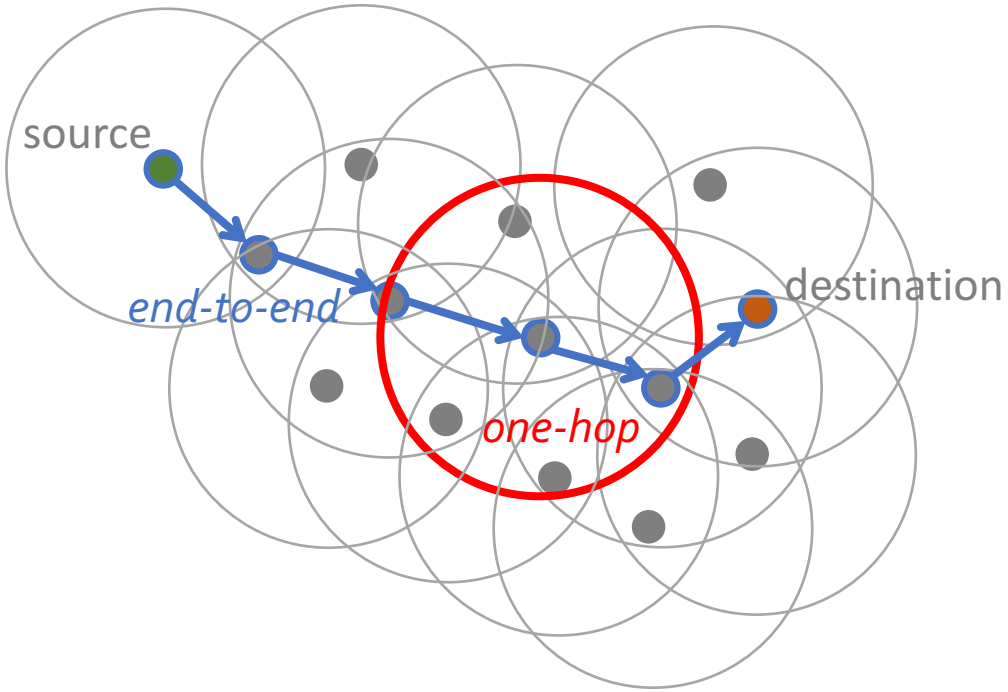
IEEE 802.15.4 Physical Layer (3)



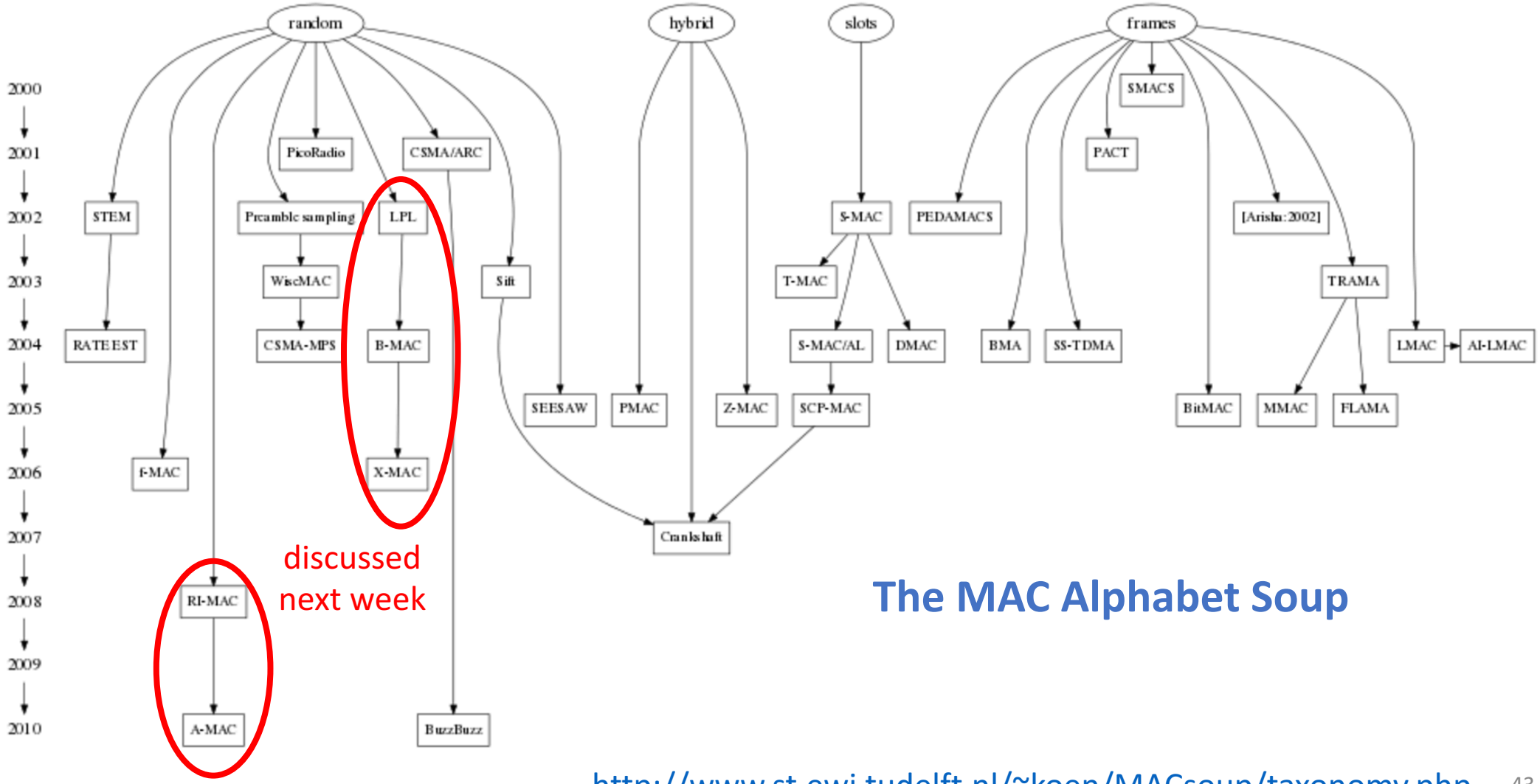
- Each chips generates a 90 degrees phase change every $T_c = 0.5\mu s$
- Data rate: $1/T_c$ chip/s = 2 Mchip/s = 62.5 ksymbol/s = 250 kbps

Low-power Wireless Communication Stack





Low-power Wireless Media Access



The MAC Alphabet Soup

http://www.st.ewi.tudelft.nl/~koen/MACsoup/taxonomy.php

Summary of Today's Lecture

- Communication with peripherals: serial buses
  - SPI de-facto standard
- Communication between microcontrollers: interconnects
  - Type of interconnect affects predictability, energy efficiency, software complexity
- Communication between devices: low-power wireless
  - Many powerful applications, but also significant challenges to realizing them
  - IEEE 802.15.4 physical layer popular for short-range, low-rate low-power wireless