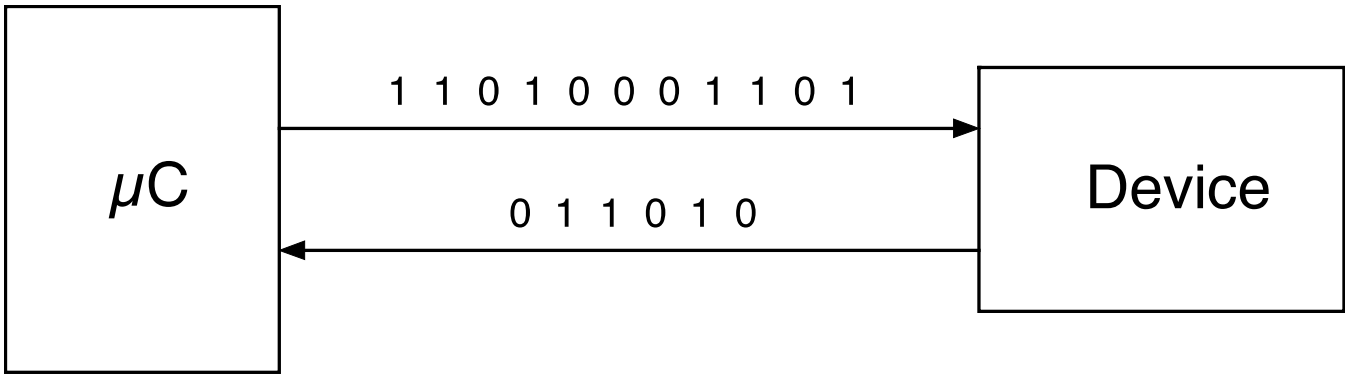
Unit 19 - Serial Communications

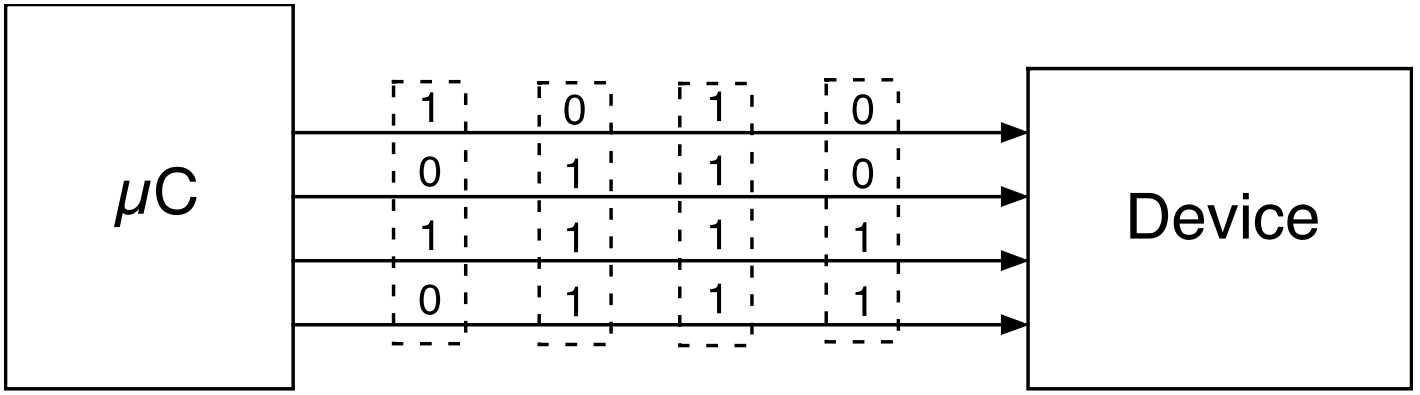
Serial Interfaces

- Embedded systems often use a serial interface to communicate with other devices.
- "Serial" implies that it sends or receives one bit at a time.



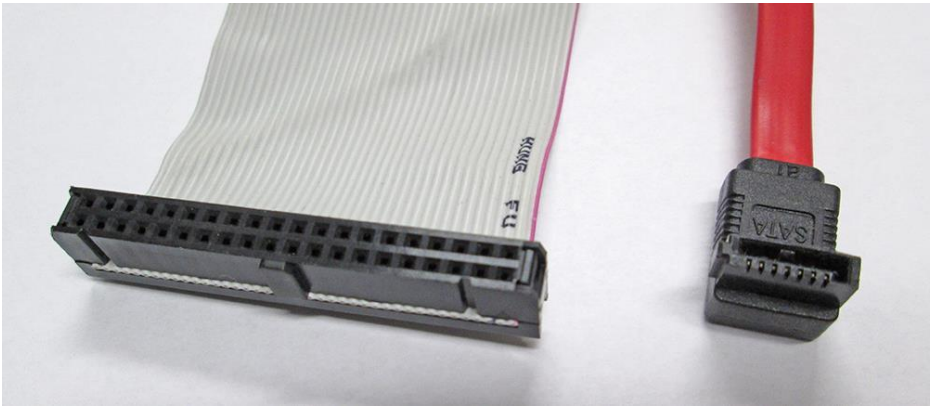
Serial Interfaces

- Different from a parallel interface that sends/receives multiple bits at a time.
- Example: The LCDs used it the labs used a 4-bit parallel interface to transfer commands and data.



Serial vs. Parallel

- Serial interfaces
  - Pros: less hardware ⇒ cheaper, good for consumer products
  - Cons: slower
- Parallel interfaces
  - Pros: faster
  - Cons: requires more wiring and larger connectors ⇒ more $$.
- Example: PATA vs. SATA disk interface
  - PATA (Parallel ATA) uses 40 conductors
  - SATA (Serial ATA) uses 7 conductor

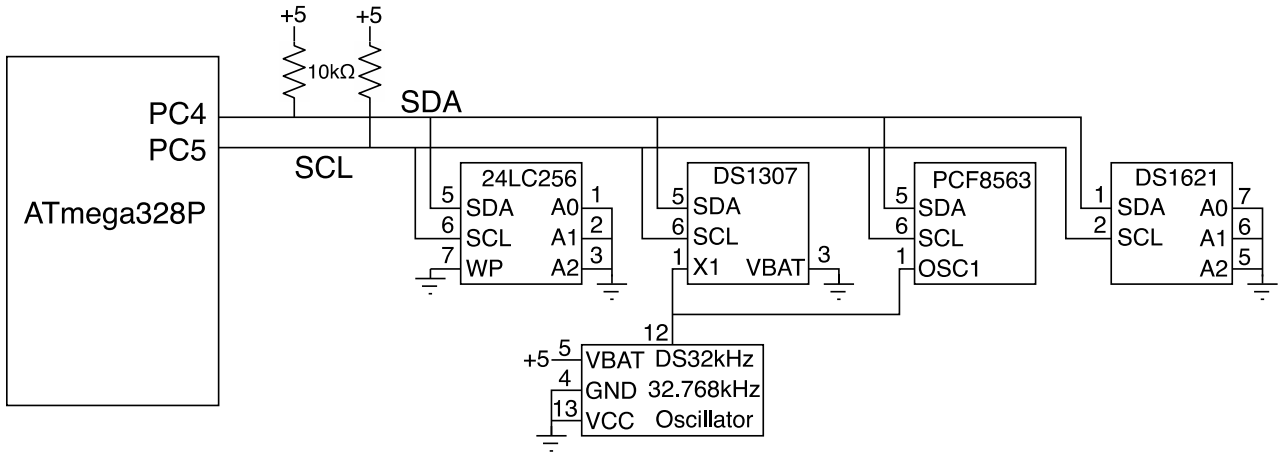


Pick Your Serial Interface

- Embedded systems can use a variety of serial interfaces.
  - Numerous manufacturers have developed interfaces.
  - Try to get customers to commit to using theirs in new designs
- Choosing which to use depends on several factors.
  - What interface is available on the device you need to talk to.
  - Speed
  - Distance between devices
  - Cost of wiring and connectors
  - Complexity of software
  - Reliability

I²C Interface

- I²C (Inter-Integrated Circuit) Interface
- Also known as the "Two Wire Interface" (TWI)
  - Clock generated by the master device
  - Data line is bidirectional
- Bus topology
  - One bus master can communicate with multiple slave devices over a single pair of wires.

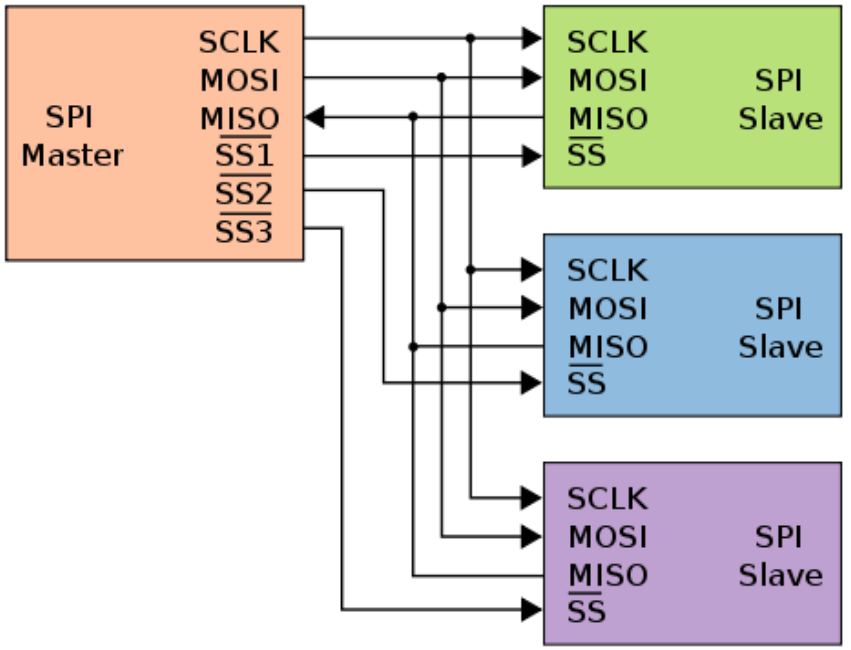


I^2C Interface

- Most commonly used on a single PC board to transfer data between two or more ICs.
- Data rates are relatively slow (usually < 100 kb/sec)
- Half duplex
  - Master ⇒ slave, or slave ⇒ master, but not at the same time
- Example: A non-volatile memory IC stores configuration data used when a system powers up.
  - Reducing the amount of wiring is more important than speed
- Software interface is relatively complex
  - Many µC's include I^2C hardware that simplify the task, a little.

SPI Interface

- Serial Peripheral Interface Bus
- Uses four wires (three in many cases)
- Full Duplex
  - Can transfer data in both directions at the same time
- Bus topology
  - One master can talk with multiple slave devices using three wires
  - Clock
  - MOSI (master out, slave in)
  - MISO (master in, slave out)
  - SS (slave select), one for each slave device



1-Wire Interface

- Uses a single wire to send data in both directions
- No clock. Timing based on length of the high and low states of the data line.
- Bus topology
  - Each 1-Wire device has a unique 64-bit ID number that the µC uses to identify who to talk with.
- Used for communicating to low speed devices (temperature IC's, iButtons, etc.)

RS-232 Interface

- A “legacy” serial interface developed in the 1960’s
- Still in wide-spread use due to it’s simplicity
- Questions to look up:
  - (search for keywords “RS-232” or ”RS232” or “UART”)
  - What is the topology of RS-232: bus, one-to-one, radial, etc.?
  - How far can an RS-232 interface communicate?
  - In the minimum case, how many wires are need?
  - What voltages does it use to signal ones and zeros?
  - Can it support full duplex signaling?
  - What are the two sizes of connectors commonly used (# of pins)?
  - What are start bits and stop bits?
  - What is a “parity bit”?
  - What is the “baud rate”
  - What is meant by “flow control”?

RS-232 Interface

- Until recently all PCs had “COM” ports that were RS-232 serial ports.
  - To add an RS-232 port to a newer system, use a USB to serial adapter.
- Uses a minimum of three wires
  - Transmit
  - Receive
  - Ground
  - Several handshake signals that are often not used.

RS-232 Interface

- One-to-one topology
- Full duplex (if both devices are capable of it)
- Longer distances
  - Specs say 50 feet, but can often be much longer (>1000 ft) with proper cables and data rates.
- Uses bipolar voltages to signal 1's and 0's

  –3 to –15 Volts = 1

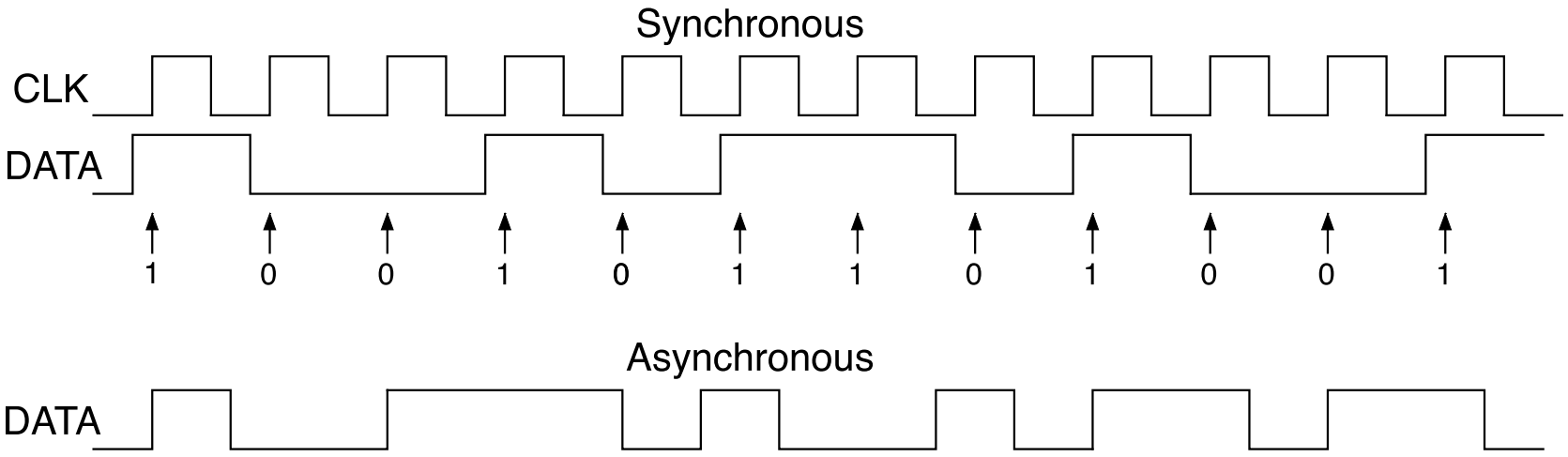
  +3 to +15 Volts = 0
- Very simple interface to implement in both hardware and software.

RS-232 Interface

- Despite its age, RS-232 is still heavily used
  - Industrial devices
  - Data logging devices
  - “Headless” servers, for use during installation
  - Anything that needs a simple interface, often for configuration

RS-232 Interface

- An “asynchronous” interface
  - I^2C and SPI are synchronous interfaces since there is clock signal
  - RS-232 only sends data, no clock signal accompanying the data
  - In order to correctly receive the data, the receiver must derive clocking information by examining the data

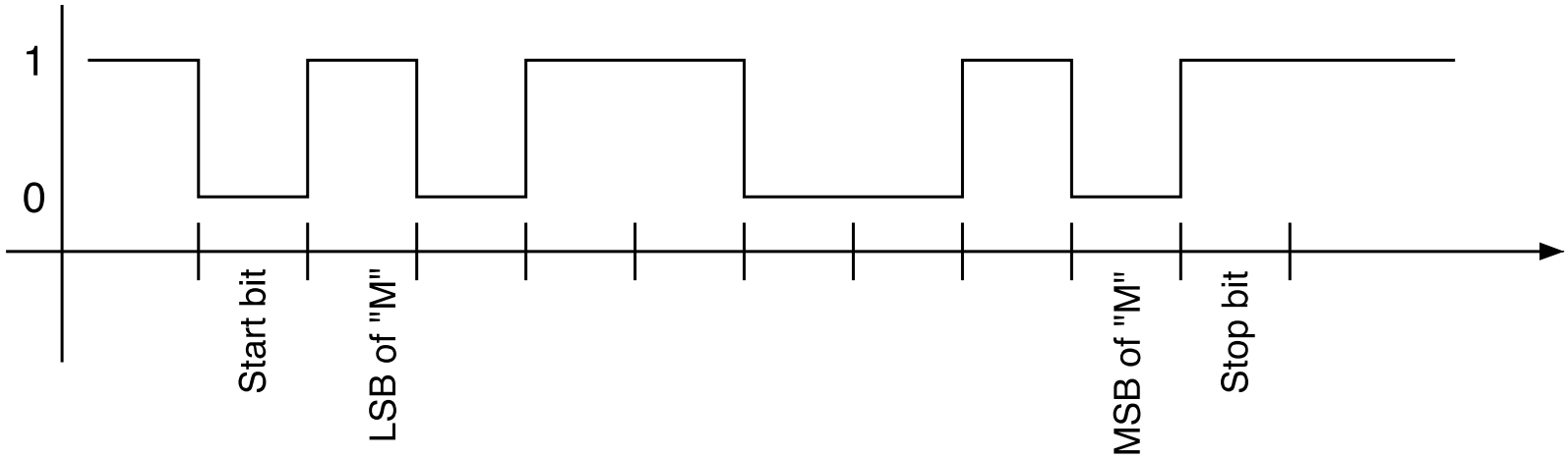


RS-232 Interface

- To correctly receive the data, the transmitter and receiver have to agree on how the data will be sent
- Must agree on data rate
  - Data rates given in bits/second or “baud rate”
  - Use any rate, as long as TX and RX devices agree on the rate
  - In most cases, standard rates are used:
    - 300, 2400, 9600, 28800, 57600, 115200, etc.
  - Many devices will specify that they can only communicate at one rate
- Must agree on the format of the data
  - How many data bits sent for each character?
  - Which comes first, the MSB or the LSB?
  - What other bits are sent along with the data?

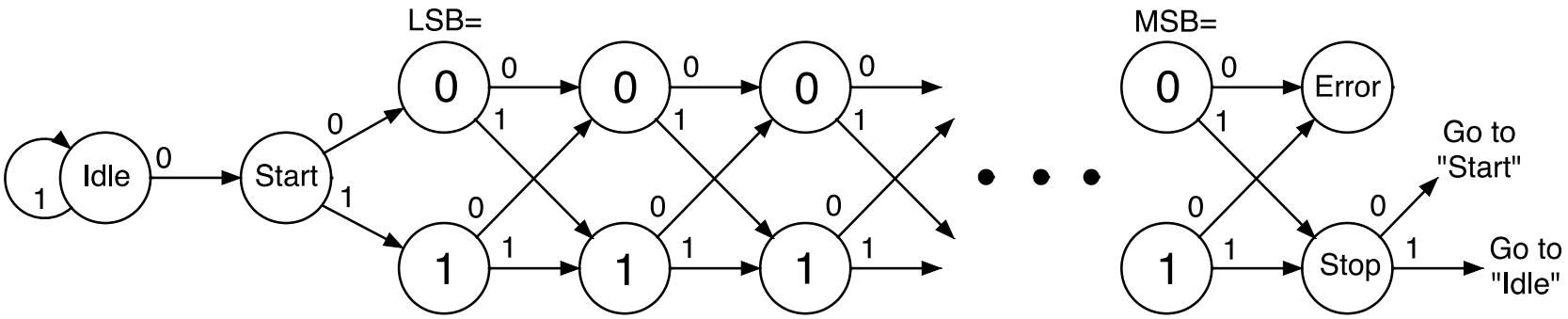
RS-232 Interface

- To send a byte, the transmitter sends…
  - Start bit (a zero)
  - Data bits, LSB first, MSB last
  - Parity bits (optional)
  - Stop bits (a one, 1 or 2 of them)
- Example: to send an “M”
  - ASCII code = 0x4D = 01001101



RS-232 Interface

- To receive a byte, the receiver uses a state machine.
- Based on the incoming bits, the receiver makes transitions between states until all the data has arrived, or an error has been detected.

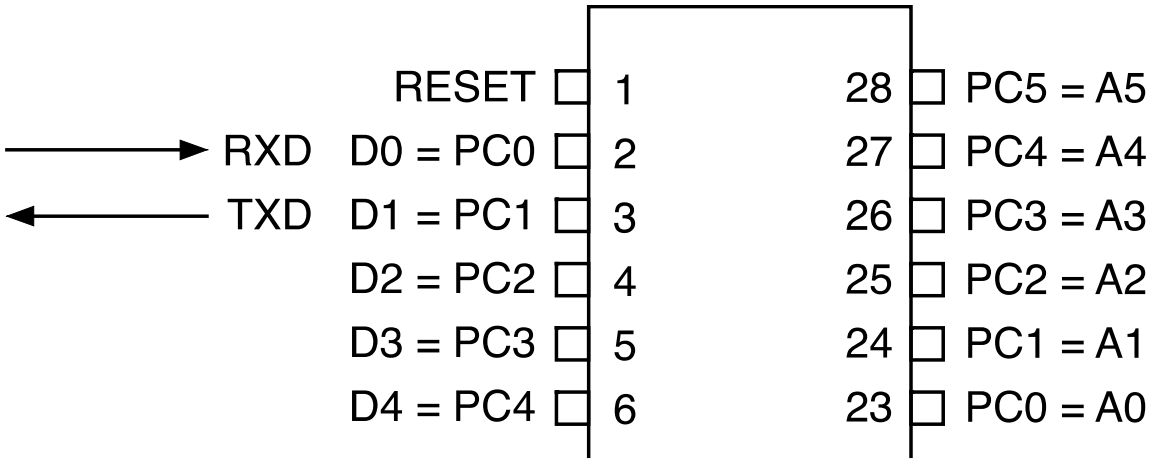


RS-232 Interface

- Parity bit – sent after the MSB to help detect errors
- Even parity
  - Transmitter adds a 0 or 1 so the number of ones sent is even
  - Receiver checks that an even number of ones was received
- Odd parity
  - Transmitter adds a 0 or 1 so the number of ones sent is odd
  - Receiver checks that an odd number of ones was received
- Transmitter and receiver better agree: odd or even
- If parity at received end is incorrect, a flag is set

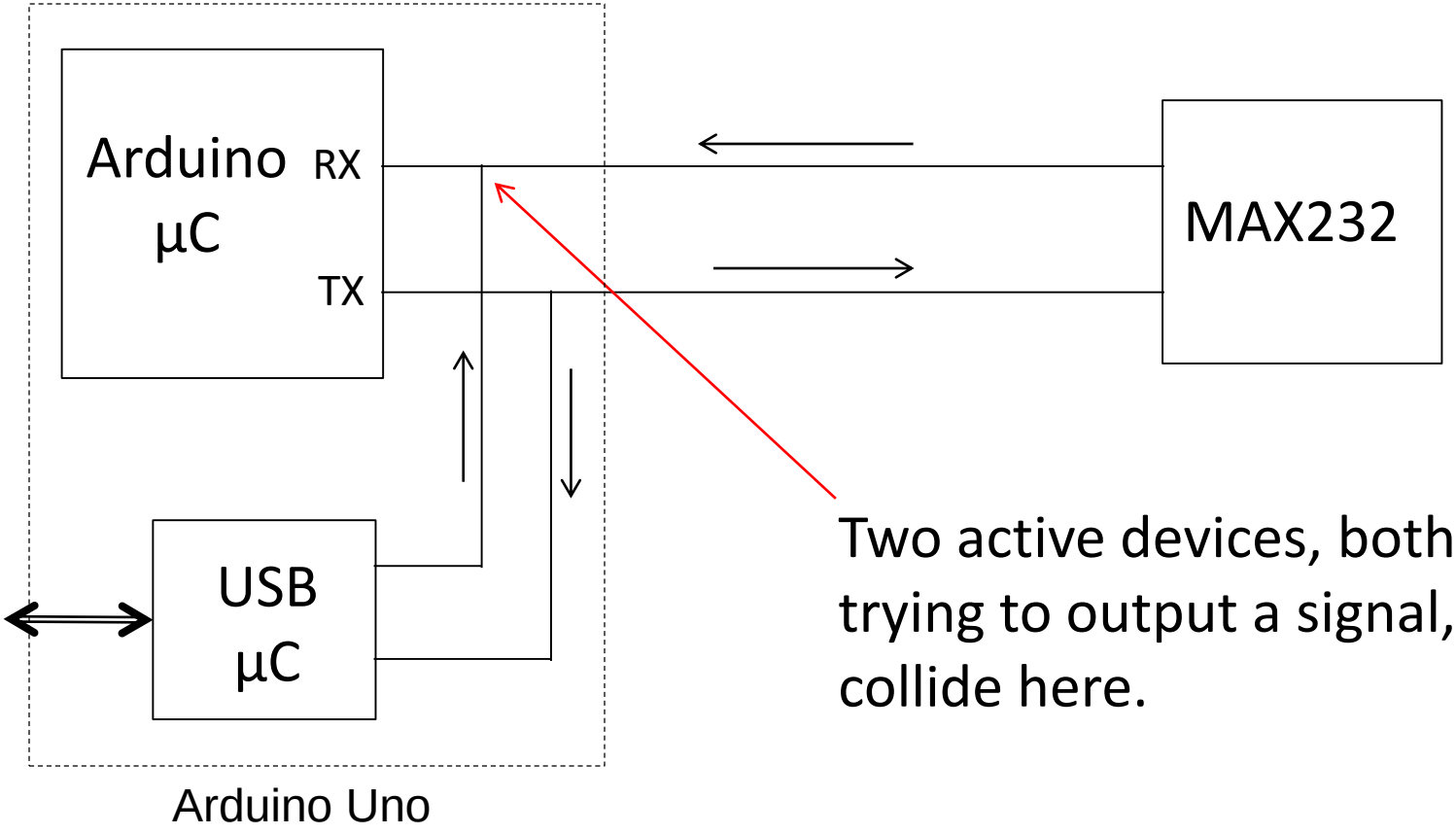
AVR USART0 Module

- Supports both asynchronous and synchronous modes
- Data lengths of 5, 6, 7, 8 or 9 bits, plus parity
- Interrupt generation on both transmit and receive
- Uses same pins as PORTD, bit 0 and 1
- If TX or RX enabled, can't use that pin for I/O



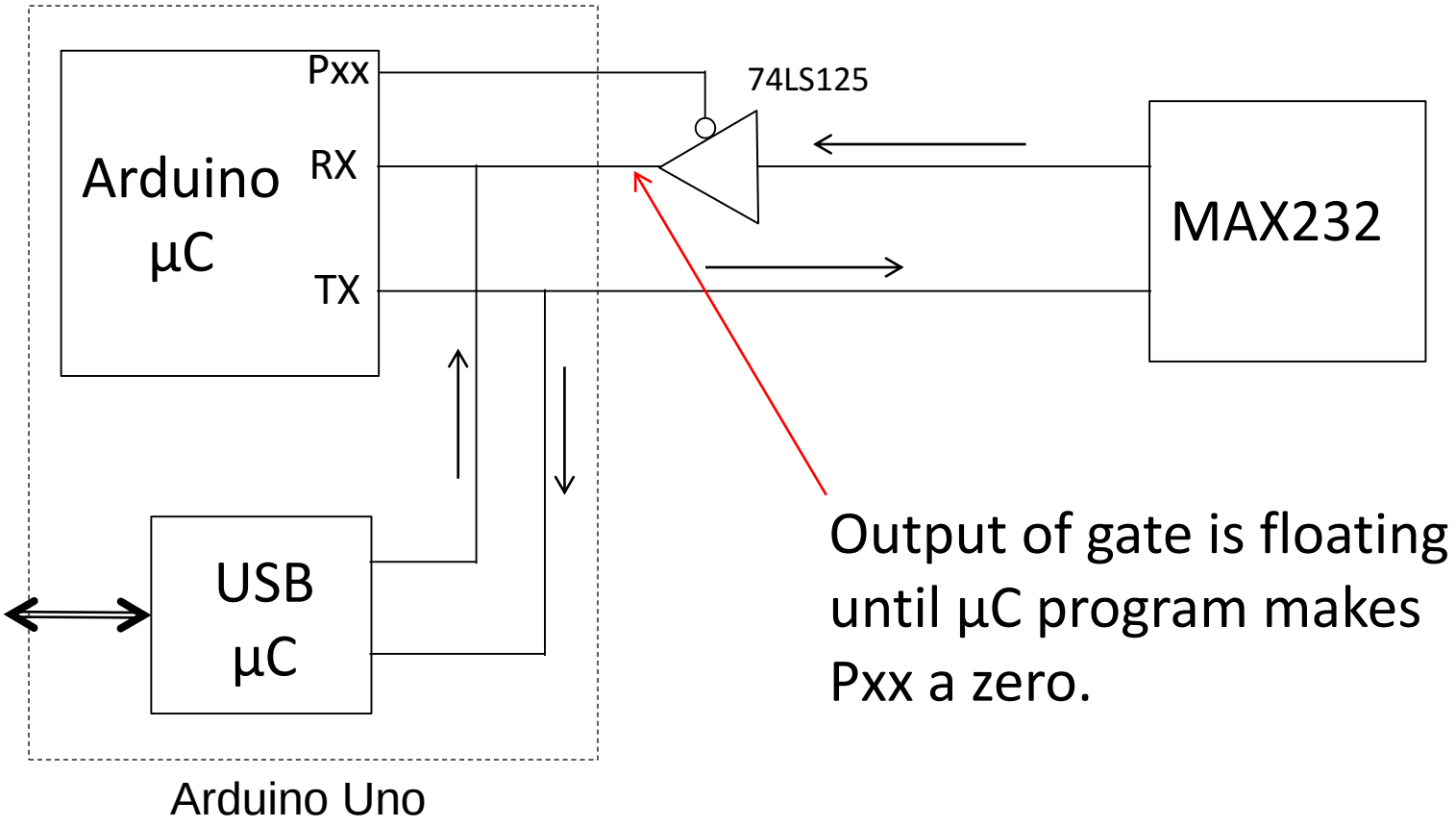
Tri-State Gates

Problem: How can you use the serial I/O lines of the Arduino, which are also used for programming it?



Tri-State Gates

Solution: Use a Tri-State gate to isolate the MAX232 received data from the µC until programming is over.



AVR USART0 Module

- Bad News: lots of registers and bits

Control and Status Register A (UCSR0A)	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Control and Status Register B (UCSR0B)	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	USCZ02	RXB80	TRXB80
Control and Status Register C (UCSR0C)	UMSEL01	UMSEL02	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0
Data Registers (UDR0)	UDR0[7:0]							
Baud Rate Register (UBRR0H & UBRR0L)					UBRR0[11:8]			
	UBRR0[7:0]							

AVR USART0 Module

- Good News: Can ignore most bits or leave as zero

Control and Status Register A (UCSR0A)	RXC0	TXC0	UDRE0					
Control and Status Register B (UCSR0B)				RXEN0	TXEN0			
Control and Status Register C (UCSR0C)	0	0	0	0	0	UCSZ01	UCSZ00	
Data Registers (UDR0)	UDR0[7:0]							
Baud Rate Register (UBRR0H & UBRR0L)					UBRR0[11:8]			
	UBRR0[7:0]							

- UDR0 – received and transmitted data register
  - Actually two registers at the same address
  - Write to it ⇒ stores data to be transmitted
  - Read from it ⇒ gets data that has been received

RX and TX by polling

- First step, find the value to go in UBRR0 for the desired baud rate.

$$\mathrm{UBRR} = \frac{\mathrm{f_{osc}}}{16 \times BAUD} - 1$$

- Use compiler directives to calculate the value

```
#define FOSC 16000000            // Clock frequency
#define BAUD 9600                // Baud rate used
#define MYUBRR (FOSC/16/BAUD-1) // Value for UBRR0
```

- Store it in the UBRR0 register

```
UBRR0 = MYUBRR;                  // Set baud rate
```

RX and TX by polling

- Second steps
  - Enable the receiver and/or transmitter
  - Set the values in UCSR0C for the desired communications settings
  - Most of the bits in UCSR0C can be left as zeros

```
UCSR0B |= (1 << TXEN0 | 1 << RXEN0);  // Enable RX and TX
UCSR0C = (3 << UCSZ00);               // Async., no parity,
                                      // 1 stop bit, 8 data bits
```

- The receiver and transmitter are now ready to go and waiting for data.

RX and TX by polling

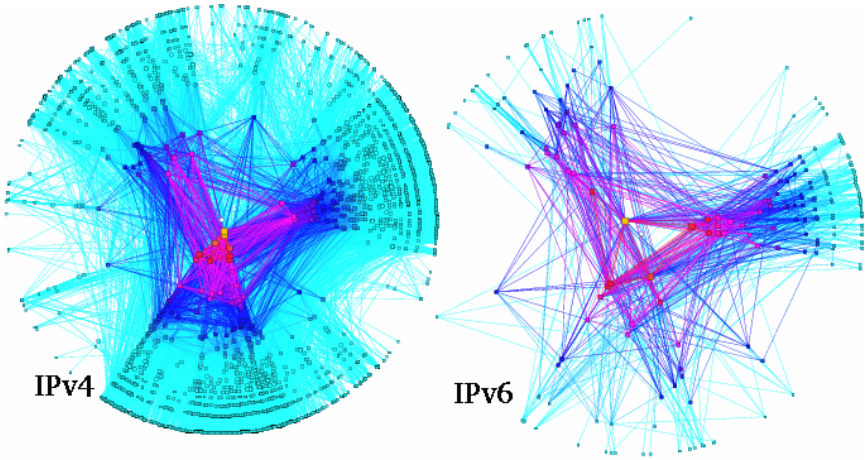
- Routines for RX and TX
  - Receiver: checks RXC0 bit to find out when new data has come in.
  - Transmitter: checks UDRE0 bit to find out when transmitter is empty.

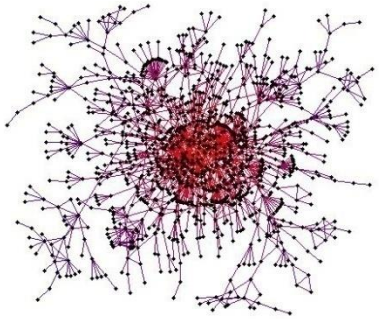
```
char rx_char()
{
    // Wait for receive complete flag to go high
    while ( !(UCSR0A & (1 << RXC0)) ) {}
    return UDR0;
}

void tx_char(char ch)
{
    // Wait for transmitter data register empty
    while ((UCSR0A & (1<<UDRE0)) == 0) {}
    UDR0 = ch;
}
```

NETWORKS OVERVIEW

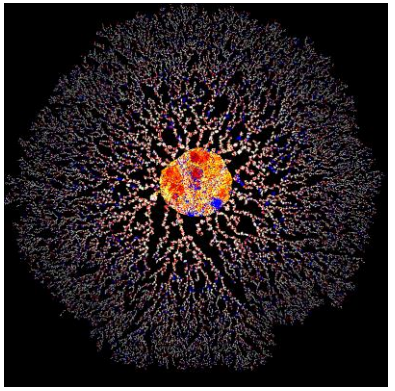
What is a Network?

- Interconnection between objects and their communication patterns
  - Computer and Data Networks
  - Biological Networks
  - Social Networks
  - Economic Networks





www10.org/program/society/yawyl/mit.jpg



Protein Homology

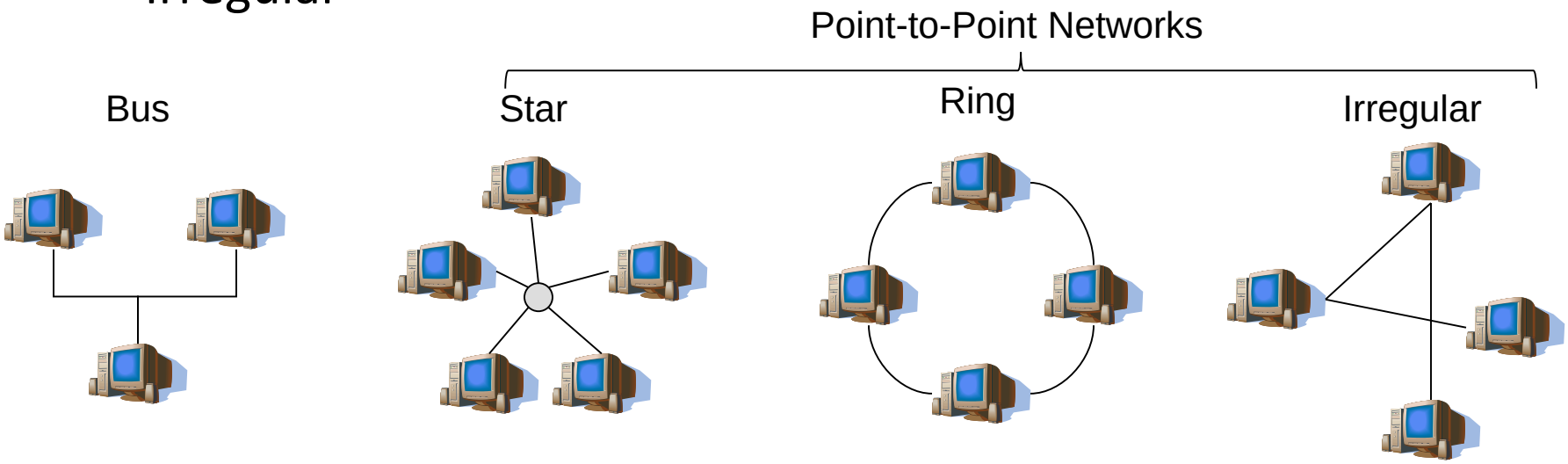
Current Issues and Areas of Research

- Routing in mobile and ad-hoc networks
  - How do we keep track of you if you are moving around
- Congestion and control of large, distributed networks (e.g. the Internet)
  - Many congestion hot spots can cause lost or late data
- Network security
  - Identifying viruses, worms, spam and stopping their propagation
  - Encryption and privacy
- Delivery of Media
  - Compression, QoS (Quality of Service)
- Sensor networks
  - Efficiently Aggregating massive amounts of sensor data to solve specific problems

TERMS AND CONCEPTS

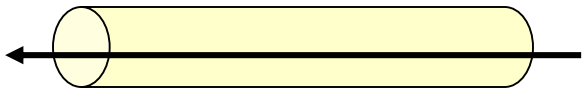
Network Topologies (Shapes)

- Topologies refer to the organization of connection paths between computers (nodes)
  - Bus (Shared connection between computers)
  - Star (Central hub of connection)
  - Ring
  - Irregular

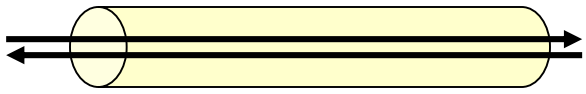


Communications Links

- Physical communication links come in two varieties
  - Half Duplex = Can either send OR receive at one time (not both at the same time)
  - Full Duplex = Can both send and receive at the same time



Half-Duplex



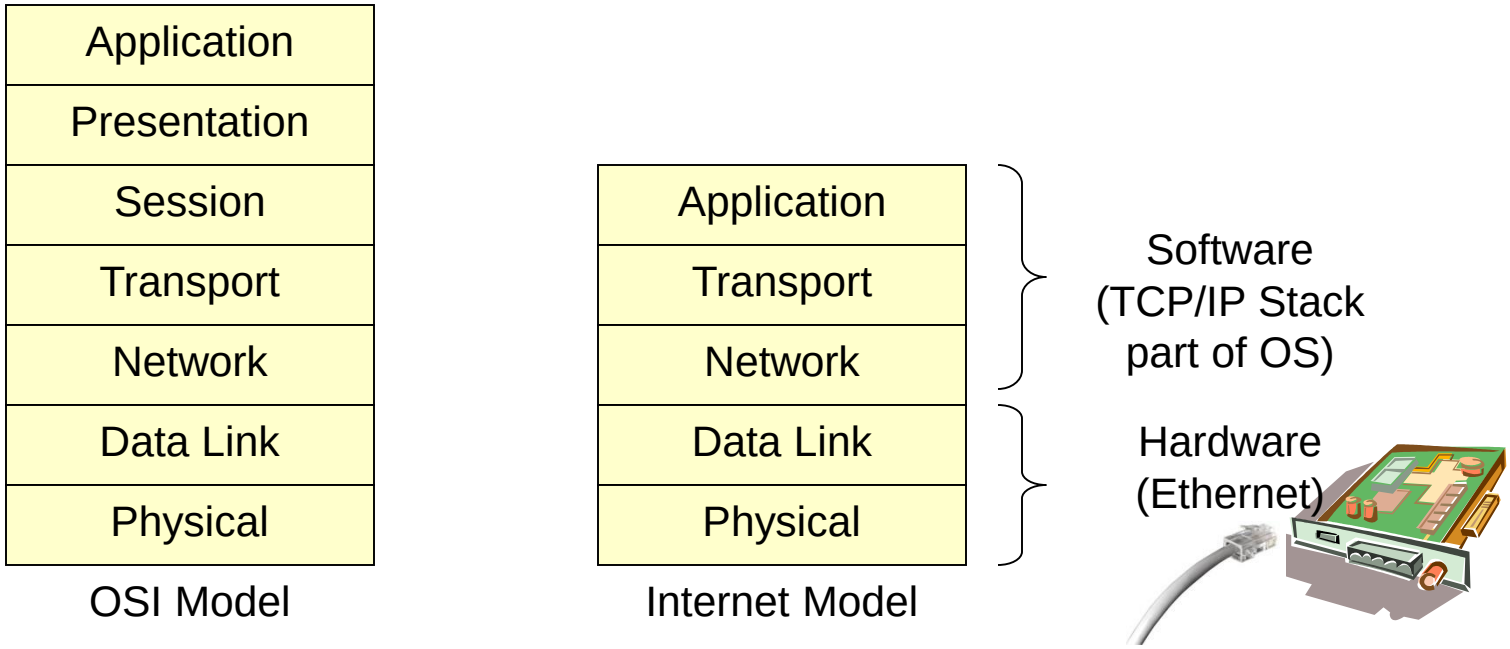
Full-Duplex

Network Goals and Issues

- Reliable, timely delivery of data from a source to destination
  - Throughput (Bandwidth): total bits per second that can be transferred
  - Latency: time it takes to deliver a message
- Issues
  - Reliability (Error Checking)
  - Timeliness (Latency/Throughput/Quality of Service)
  - Delivery (Routing)
  - Interoperability (Connecting different networks and services)

Layered Approach

- To deal with these issues networks use a layered approach combining hardware and software
- Two layered models
  - OSI model – General template; not all layers needed in practice
  - Internet model – Current model of the Internet (includes TCP/IP)



Internet Model

- Description of Layers
  - Application
    - Software application that uses the network capabilities
    - Example: Browser, IM, E-mail
  - Transport
    - Provides end-to-end reliability (error-correction) and coordinates multiple applications accessing the network through the use of ports
    - Example: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol)
  - Network
    - Provides network-to-network routing capabilities
    - Example: IP (Internet Protocol)
  - Data Link
    - Hardware for transmitting a message one hop (just to the next node on the overall route)
    - Example: Ethernet, PPPoE
  - Physical
    - Actual wire or wireless link/carrier
    - CAT 5 Wire, Radio-Frequency Wireless transmission

Protocols

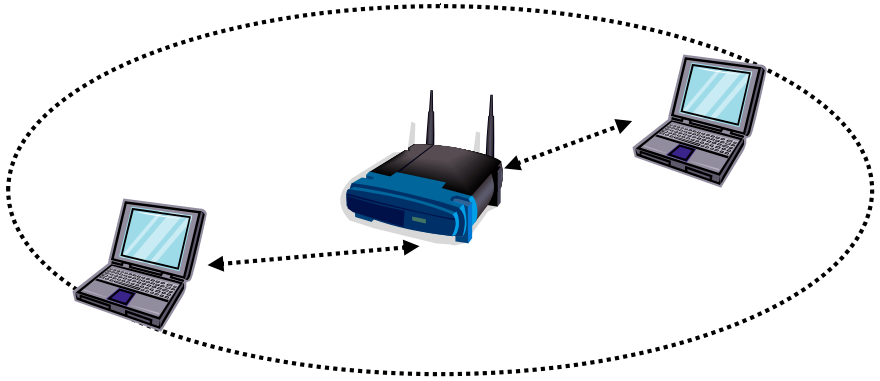
- Rules by which systems interact
- Includes…
  - Rules to access a shared communication link
  - Rules by which systems agree to interact
  - Rules for the format of data so that receivers know how to interpret what they get

Issues and Implementation

WIRELESS NETWORKS

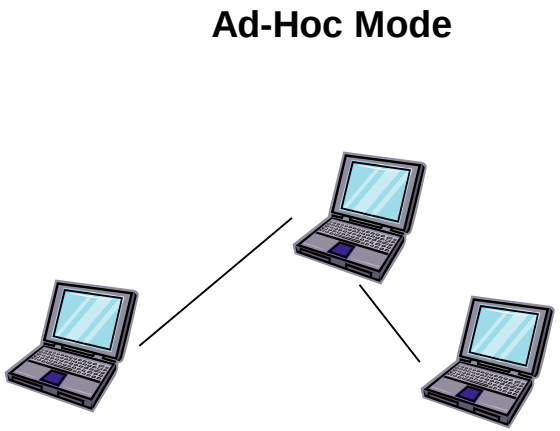
Wireless Networks

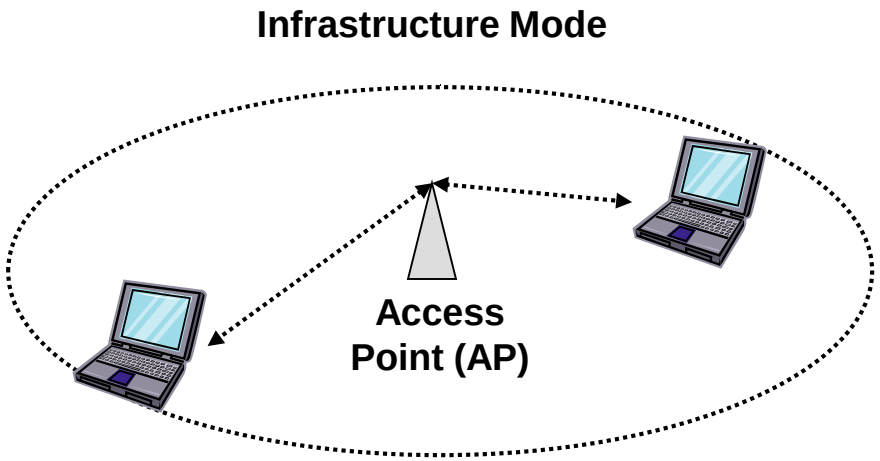
- Communication via radio-frequency transmission
- IEEE 802.11b,g,n,ac standard



Wireless Network Organization

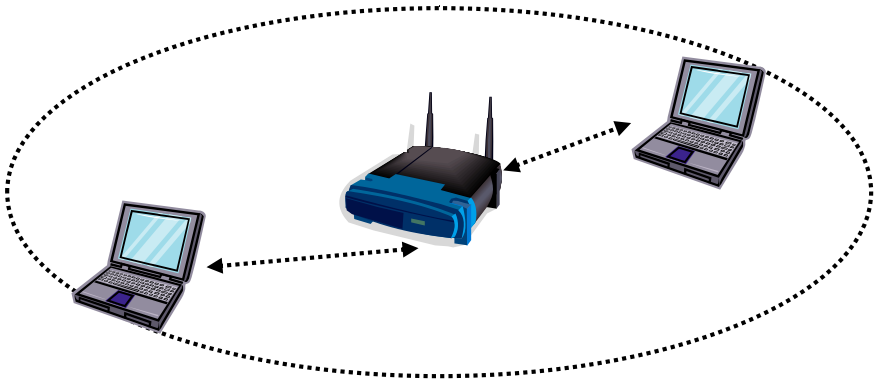
- Infrastructure Mode: All communication through a common access point (AP)
- Ad-Hoc Mode: Computer-to-Computer communication to those within range





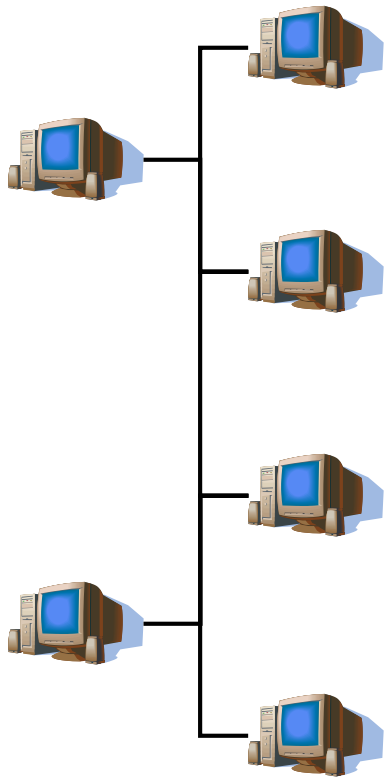
Wireless Issues

- Security
  - Anyone can “hear” the transmissions
- Access Control
  - When is it safe/unsafe to transmit?
    - When 2 nodes transmit at same time (called a “collision”), information is garbled



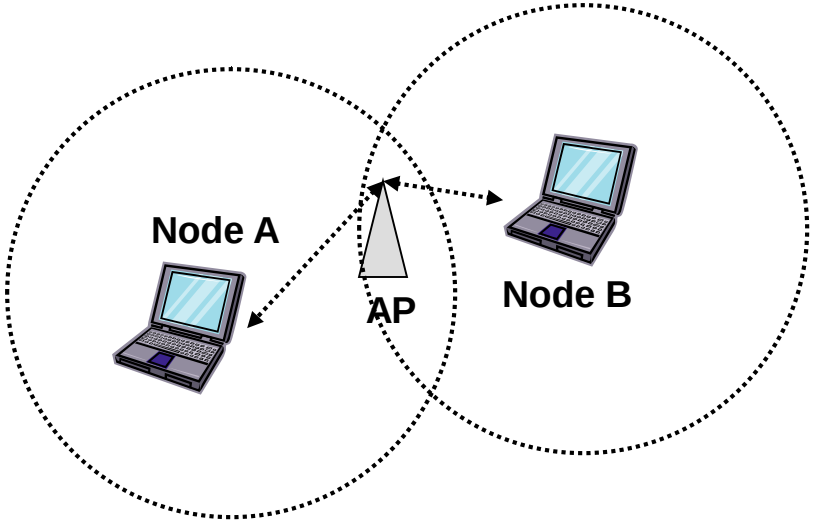
Handling Collisions on Wired Networks

- On a shared link, only one computer can transmit at once
- Collisions occur if two transmissions start at the same time
- Algorithm - CSMA/CD (Carrier-Sense, Multiple Access / Collision Detection)
  - Wait for data to transmit
  - if (someone is already transmitting)
    - Wait until transmission is complete
  - Begin your transmission and while transmitting, check for collision (i.e. listen while you speak/transmit)
  - if (collision)
    - wait a “random” amount of time and start again (go back to the first if statement)
  - Done



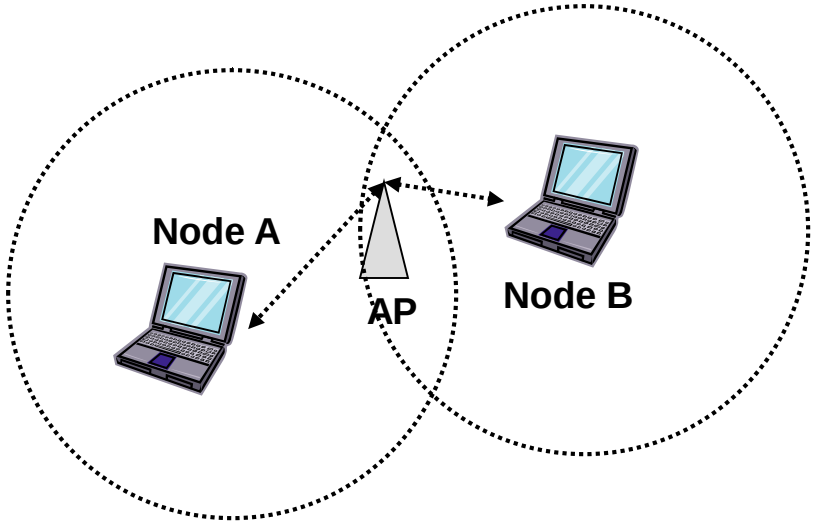
Problems Detecting Wireless Collisions

- Radio devices can usually only transmit or receive (listen) at once (half-duplex)
- Two transmitters may be in range of common destination but not of each other ("Hidden terminal" problem)
- Solution
  - Collision "Avoidance"



Hidden Terminal Problem

- The carrier frequency may be free at the sender but not the receiver
- Two nodes are in range of the AP but not of each other

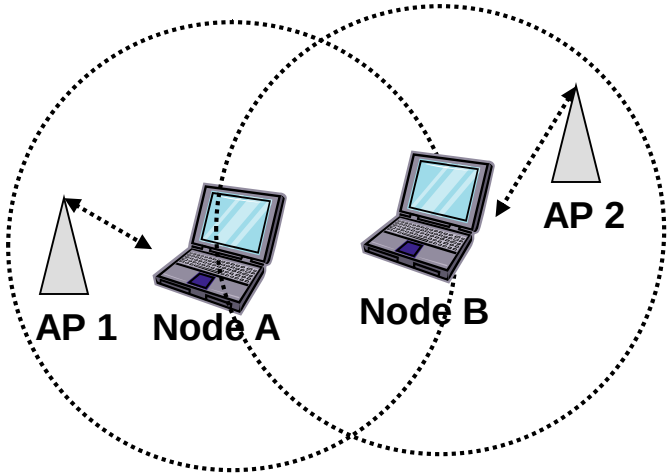


Node A is in range of AP but out of range of B. Thus, even if it listens before transmitting, it can't detect that B is already transmitting and thus a collision will occur at the AP.

Node B is "hidden" to A

Exposed Terminal Problem

- The carrier may be free at the receiver but not the sender
- Two node are in range of each other but sending to different destinations. A collision would not occur.

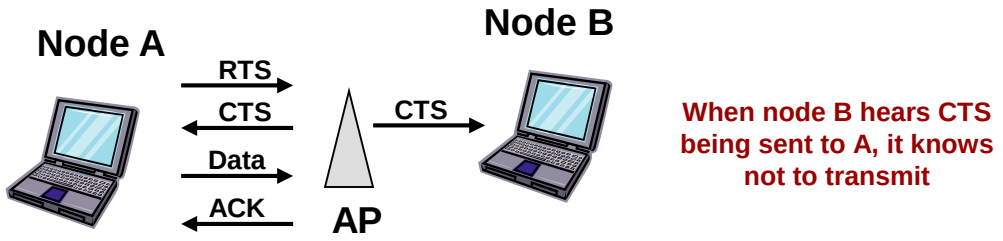


Node A is in range of AP1 and node B. Thus, if node B is transmitting, A will think it cannot. However, it could safely transmit to AP1 w/o a collision.

Node B is an "exposed" terminal

Virtual Carrier Sensing

- Solves hidden terminal problem
- Ask permission from AP before initiating a transfer
  1. Send short RTS (Request-to-Send)
  2. AP replies with a CTS (Clear-to-Send)
  3. The sender can proceed hopefully w/o a collision
  4. AP acknowledges after "good" frame received w/o error or collision
- Key:
  - RTS heard by all nodes within range of Sender
  - CTS heard by all nodes in range of receiver and indicates that they someone else is going to transmit
  - Those in range of either Sender or Receiver know not to transmit



Virtual Carrier Sensing

- What problems may arise?
  - Two nodes send RTS at same time (If no CTS within certain time period, backoff and try again later)
  - A node moves into range after CTS is sent out
- Still need acknowledgement in case of collision

CSMA/CA

- Carrier-Sense, Multiple Access / Collision Avoidance
  - Algorithm
    - Wait for data to transmit
    - If carrier frequency is idle, begin transmission
    - If it is busy, select a random back-off time and wait until carrier frequency *has been idle for that amount of time* to try again
  - Acknowledgement frames are used to indicate successful receipt of a wireless transmission.