Introduction to Networks

Overview

In this lesson we will

- ✓ Begin with an overview of the hardware and software of basic networks.
- ✓ Introduce the network architecture and protocol stack.
- ✓ Identify the main components of a message structure.
- ✓ Introduce the basic reference models.
- ✓ Compare and contrast the OSI and TCP/IP models.
- ✓ Develop a simple local message exchange.
- ✓ Examine synchronous and asynchronous message exchange.
- ✓ Examine the I^2C bus.
- ✓ Learn some of the jargon and acronyms

Introduction

Networks - What are They

What is a network - a bit difficult to define
We use the word all the time and seem to understand its meaning
Can talk of networks of
People
Roads
Companies
Telephones
Here we're going to talk about networks of computers

In the early days of computing
Had one big computer
Painted blue
Kept in an air conditioned room and fed punch cards
All the data and information kept in one place

Cheaper computers
Allowed more computers
Moved them out of the air conditioned room onto the desktop
Stuff suddenly got distributed
People still needed to share the stuff

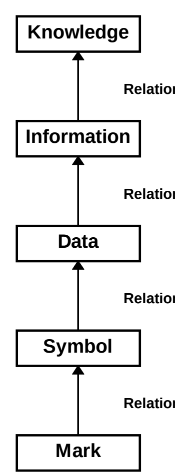
Sharing stuff
- At the start sharing stuff meant
  - Writing the stuff to be shared to a tape or big floppy disk
  - Walking over to the person who needed the stuff
  - Copying the files to the new computer
    - Synchronizing stuff became an immediate problem

- Major breakthrough
  - RS232
    - Someone got the bright idea
      - Let's connect the computers together using a couple of wires
    - Pins 2, 3, and 7 that's all you need - trust me
    - That was the start and the end
    - The rest is history

Stuff
- What is stuff
  - Can mean a lot of things
  - Let's start at the bottom
- We start with a collection of symbols or marks
  - These are our ***alphabet***
  - In isolation they have no meaning
- We next associate a meaning with the symbols
  - This gives us ***numbers letters*** or other such primitives
  - Still these have no real meaning

- We next begin grouping the symbols or associating a context
  - We now have ***data***
  - This is the first step towards something useful
  - The symbol 1 may mean a single buffalo or a biological need
  - The symbols leaf may indicate a tree appendage or a new set of clothes

- Applying additional groupings and context to data
  - Gives ***information***
    - Note the word *information* will be used throughout
      - Convey the general notion of
        - Stuff
        - Moving stuff
      - Different from meaning here
        - Intent should be evident from the context



  - Continuing gives ***knowledge***
  - This gives a hierarchy of the form
    - Each level provides
      - A richer expressive power
        - Greater flexibility

History - Early Networks

- Primitive with
  - Word of mouth
  - Drums
  - Smoke
  - Polished Glass
  - Messengers
  - Pony express
    - Passing message from person to person
    - Primitive routers
  - Alexander

Objective of Networks

- Move information from one place to another
- As amount and importance of information increased
  - Complexity and sophistication of the network
    - Increased correspondingly

Elements of a Network

- Piece of information - *Message*
  - May be encoded
  - Represented some way
- Means by which to send it - *Transport mechanism*
  - Hardware / software / whatever
- Place or places to send from and to

The Hardware

- Can offer a number of different views
  - Observe the views are not orthogonal
  - Can have combinations

✓ Types by Transmission Technologies

- Broadcast
- Multicast
- Point to Point

✓ Types by Information Flow

- Serial
  - Bit
  - Character
  - Word
- Parallel
  - Bit
  - Character
  - Word

- ✓ Types by Topology
    - Star
    - Ring
    - Tree
    - Full or complete connectivity
    - Ad hoc
    - Network can be
        - Interconnected collection of nodes and arc
        - Easily modeled using graph theoretic techniques
            - Simple
                - Each node in the graph
                    - Is a vertex
                - Each connection between two nodes
                    - Is an edge
        - Collection of sub-networks
            - Need interconnections between sub-networks
                - Such connections provide for inter network communication
                    - Referred to as ***inter-net*** connections
            - Most familiar is the Internet
                - Provides interconnection between networks all over the world
- ✓ Types by Distance
    - Local Area Networks - LAN
        - Small to medium geographical area
            - Automobile or aircraft have several LANs
            - Boundary scan test technology uses integrated on chip networks
            - A computer may internally connect constituents using a LAN
        - Single room with a couple of devices
        - Several buildings several hundred devices
        - Single city to several cities
    - Wide Area Networks - WAN
        - Medium to large geographical area
        - Single city to several cities
        - Single Country
        - Several Countries
- ✓ Types by Technology
    - Sneaker net
    - Wire
        - Simple as a couple of wires
        - Twisted pairs
        - Co-axial cable
    - Fiber Optics
    - Wireless
        - Radio waves of one form or another
            - May be direct digital form

Traditional modem through cellular telephone
Easy to install
Excellent in areas where wire infrastructure is not in place
Cheaper than wire based
Typically slower than wire type
More prone to errors
Environment
Interference

- ✓ Types by Mobility

Mobile computing
Typically wireless but not necessary
Stationary
Typically wired but not necessary

- ✓ Interconnecting Networks

The Software

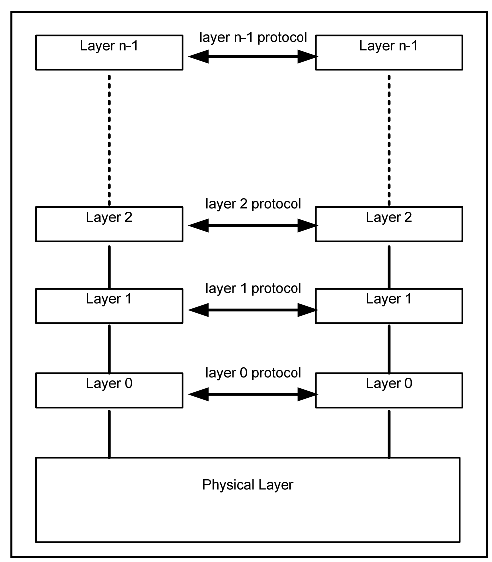
Hardware provides
The physical means by which data is moved from one place to another

Software includes
Data
Addressing and control mechanism

Virtual networks
Most contemporary communications networks can be viewed as
Hierarchy of virtual networks
Have talked about the hardware
This is the lowest layer
Often called the physical layer
Above the hardware
Varying number of software layers or levels

Between levels have relationship of
Service provider and a *service consumer*

At each level protocols may be implemented in
Either hardware or software
Generally
Lower levels are done in hardware
Upper levels are done in software



Terminology
    Entire collection
        Called *network architecture*
    Set of protocols used
        Called a *protocol stack*
    Information sent on each level
        Called a *message*

Possible that a message on higher level
    Composed of several lower level messages

Will discuss in more detail shortly

Today, a wide variety of protocol standards are available
    Are times when a proprietary network and protocol must be used
        Given a choice one should opt for one of the standards

    General objective of the standards
        Facilitate message exchange in a specific application context
            Small computer networks (EIA-232 or USB)
            Simple local area networks (Firewire, Bluetooth, I^2C, SPI)
            WIFI or other types of wireless technology
            Automotive networks (CAN bus)
                Manufacturing environments (CAMAC)

        Virtual network hierarchy should be familiar concept
            Consider a computer
                Lowest level is the physical machine
                Above the hardware
                    Machine Code
                    Assembler
                    Higher level languages C C++ Basic
                At each level we have a virtual machine
                    Each machine has its own language
                    Each has its own set of instructions

Bits and Bytes

At the lowest level
    Communication is simply a collection of 1's and 0's
As we move to higher levels
    We place (different) meaningful interpretations on
        Information on the level below

Protocols

A protocol is simply an interpretation placed on
    Bits and bytes to be transmitted or received
As noted earlier we're transferring ***messages***

Message

Is comprised of a collection of bits

Some are interpreted as

Data

Some are interpreted as

Control information

Often called a ***header***

Data

This is the actual stuff that is to be sent

Moving the data is the goal of the communication

Control

Control information is the overhead

Getting the data from one place to another

Two kinds of control

Header information

Scheme for executing the data transfer

Header Information

Potential header elements might be

Address or identifier information

Provides routing and destination information

Identifies the senders and receivers of the message

Indication of the message

Start and End

Size

Message type or structure

Padding or fill bits

To ensure proper size

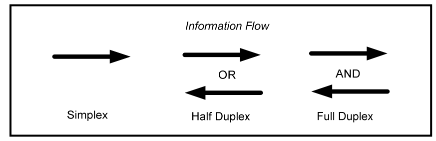
Separator

Error information

Detection only

Detection and correction

Transfer scheme



Considerations

Simplex transport

Information being sent in one direction only

Sender to receiver or receiver to sender

Half duplex transport

Information permitted to flow in both directions

Only one direction at a time

Full duplex transport

Exchange supports flow in both directions

Possibly simultaneously

- Number of channels
- Speed or Timing
  - Individual bits
  - Flow control
- Message ordering
  - Sub messages are not always sent in sequential order
- Handshaking
  - Use or not use
  - Level
    - Complete message
    - Sub messages

Connection and Connectionless

Two kinds of services

Connection Oriented

Connectionless

Connection Oriented

Transaction protocol

Establish the connection

Transact the business

Terminate or release the connection

Messages

Enter one end

Extracted from the other end

Ordering preserved

Connectionless

Each (sub)message carries full address information

If a message is composed of several sub messages

The sub-messages may not be in the order sent

Messages sent as

Datagram service

Message sent - receiver does not acknowledge

Acknowledged datagram service

Message sent - receiver acknowledges

Request-reply service

Sender transmits a datagram containing a request

Receiver returns a datagram containing the answer

Frequently used in the client-server model

Services and Service Primitives

Services

Collection of primitive actions available to the user

Request some action to be performed

Report on the action taken

These provide the public interface to the service

Similar to

Function members in C++

Methods in Smalltalk

Types

Services may be

Confirmed

Transaction protocol

request - for an action

indication - the action occurs

response - receiver responds to the action

confirm - the receiver confirms the action

Unconfirmed

Transaction protocol

request - for an action

indication - the action occurs

Service Primitive

Implementation of a particular action

Network Models

Let’s examine several different network models in current use
Each is implemented as a collection of layers as discussed
We generally have two kinds of systems

Open

Designed to communicate with other systems and vice versa

Closed

Communication is limited to the confines of the system

While unique to their particular context
Most of these models trace back to two major protocol schemes or stacks
OSI and *TCP/IP*

The OSI and TCP/IP Models and Protocol Stacks

OSI *Open Systems Interconnection model*
Proposed and developed by International Standards Organization – ISO
OSI protocol specifies a 7-layer virtual machine

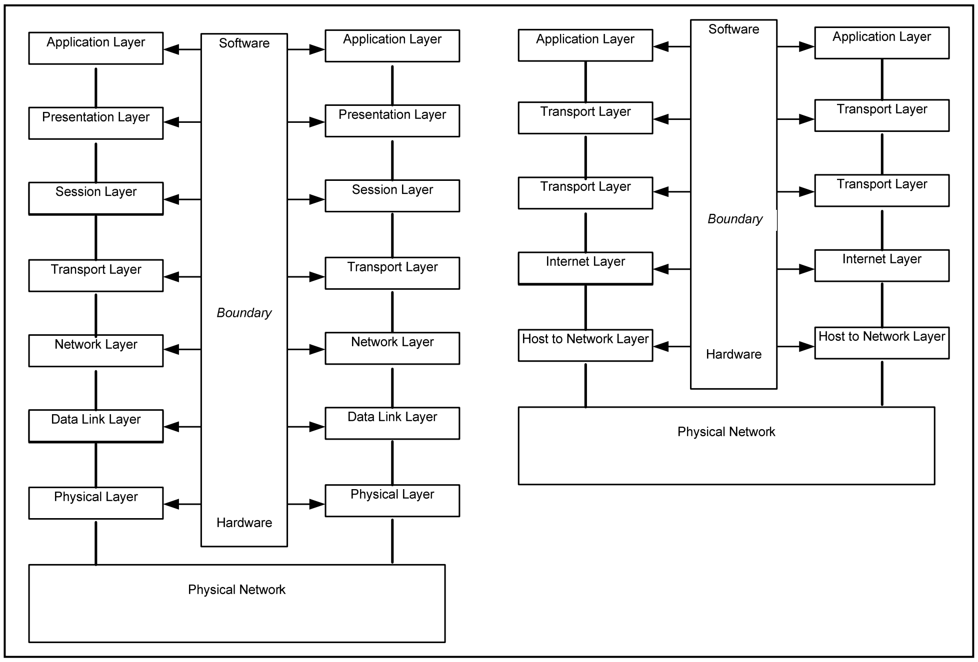
TCP /IP *Transmission Control Protocol / Internet Protocol*
Comprises 5-layer virtual machine

Physical and data link layers of OSI
Combined into host to network layer in TCP/IP

OSI	**TCP / IP**
Physical	Host to Network
Data Link	
Network	Internet
Transport	Transport
Session	Not Present
Presentation	Not Present
Application	Application

Following diagrams present and compare
Hierarchical architecture and layers
For the OSI and the TCP/IP models

At the network layer and below
Models are hardware based
Above that level
Models expressed in software



The Physical Network

Similar in both OSI and TCI/IP models
    Can be implemented as
        Copper, fiber, or air connection

Provides physical means to move collections of *bits* (1's and 0's)
    Over a communications channel
No meaning or structure to the collections

OSI – Physical Layer

At this level concern is for
- ✓ Mechanical and electrical interfaces
- ✓ Integrity of bits
- ✓ Physical characteristics of the bits
    - Logical 1 or 0
    - Such characteristics include
        - Number of volts
        - Width (in time) of each bit
            - Determines bit transfer rate through the channel
- ✓ Transfer scheme
    - Uni-directional
    - Bi-directional
- ✓ Connectors and cabling

Control issues address
    How a connection is established and released

OSI – Data Link Layer

Moves collections of bits aggregated as *frames*
- Sender
  - Breaks the data stream into *data frame*
- Receiver
  - Acknowledges reception via an *acknowledgment frame*

Data Link Layer must
- ✓ Create and recognize frame boundaries
  - Accomplished by surrounding a frame with delimiters
    - Header and trailer
    - Delimiters must be distinguishable from the data frame
- ✓ Handle frame re-transmission in the case of corruption
  - Must accommodate duplicate re-transmission
    - If acknowledge frame is lost
- ✓ Handle duplex transmission and acknowledgment
- ✓ Handle broadcast traffic including shared channel access

TCP/IP – Host to Network

No significant requirements are specified at this level
Host system must simply be able to
- Connect to the network
- Transmit or receive IP (Internet protocol) packets

OSI – Network Layer

The OSI *network layer* corresponds to the TCP/IP *Internet* layer.

Manages the routing of a transmission from the source to the destination
Routing alternatives
- Fixed path via static tables
  - Predetermined and integrated into the network
- Determined at the time of the connection
- Dynamically modified throughout the session
  - May be done to relieve congestion
  - Accommodate network failure
  - Find 'shortest' path

Must accommodate
- Different characteristics between or among networks
  - Addressing
  - Message size
  - Protocols
- Manage accounting for network use

At the network layer and below
    Activities are directed toward
        Managing the network
        Physical movement of data
    Bits are collected into manageable packets
Above the network layer
    Collection of virtual machines
        These have the task of managing the session
    The top-level message is broken down into manageable packets
The interface occurs at the *transport* layer
    Below is the subnet
    Above is the session

TCP/IP – Internet Layer

As noted
    Key layer
Defines the official
    Packet format
    Protocol
        IP - Internet Protocol
Objective
    Get message comprised of packets
        From point A to point B
    No requirement placed on
        Packet ordering during transmission
        Route a packet may take
            All packets may not take the same route

OSI – Transport Layer

Among the tasks of the *transport* layer
    Accept data from the *session* layer
        Immediately above
    Subdivide into packets that are compatible with the *network* layer
        Immediately below
Desire transactions to be implemented such that
    Hardware appears invisible to the higher layers
Must accommodate
    Speed demands
        High speed
            Subdivide problem into multiple parallel transmissions
        Low speed
            Implement as a single path
    Price demands
        Maximally use the transport mechanism is the transaction cost is high

Determine the type of service
- Point to point in order sent
- Transport of individual message components
- Broadcast to multiple destinations

Control the flow of information
- Prevent over / under run

TCP/IP – Transport Layer

Equivalent to the OSI transport layer
- Similar responsibilities
  - Interface with the network

Two communication protocols are defined for the layer
- TCP
- UDP

<u>TCP</u>

- TCP - Transmission Control Protocol
- Very reliable
- Connection oriented protocol
- Ensures data stream
  - Originating on one machine
  - Delivered to any other machine on the internet
- Process
  - Disassembles the byte stream into packets
  - Passes them to the internet level
- Handles flow control

<u>UDP</u>

- UDP - User Datagram Protocol
- Unreliable
  - In the sense that messages are not inherently acknowledged
- Connectionless protocol
- Alternative to TCP
  - Designed for hosts who want to implement their own
    - Packet sequencing
    - Flow control

- Finds application in
  - Request - response type applications
    - Client server
  - Trade speed for accuracy

OSI – Session Layer

- Permits users on separate or different machines to communicate
- Like transport layer - supports movement of data between machines
- Offers richer set of features and capabilities
  - Analogous to moving to a higher-level language
    - Can do a job in assembler
    - Easier to do in language like C

- Potential services
  - Note
    - Moving from *must dos* to *offers to do*
  - Manage dialog control
    - For single direction transmission
      - Track turns to send
  - Manage tokens in token passing protocols
    - Example
      - IBM token ring
    - Like a relay race
      - Cannot use the network until you have been given permission
      - Permission is in form of a token passed around
  - Synchronize transactions
    - Reassemble message if necessary
    - If complete transfer cannot be completed in single session
      - Major error
      - Line drop
      - Machine crash
    - Rather than retransmit complete message
      - Resume from point of last correct reception

OSI – Presentation Layer

- Moving up another layer to richer set of tools
- Goal of presentation layer is to offer generic set of solutions to common problems
- Potential services
  - Map information - types, structures, encoding etc.
    - From source computer representation
    - To network representation
    - From network representation to destination representation

  - Similar to p-code generated from some compilers
    - At the p-code level can be moved between machines
    - At the machine code level cannot be moved

OSI – Application Layer

- This level also deals with incompatibilities between
  - Systems at opposite ends of the network
    - Hardware
      - Some accommodation at this level
      - Usually done at a lower level
    - Software
      - Primary focus
- Potential incompatibilities
  - File systems
  - Terminal types
  - Mail systems
  - Remote procedure execution

TCP/IP – Application Layer

- Supports / Contains all the high level protocols
- Three basic ones in support of the original intent of the development
  - Virtual Terminals - TELNET
    - Recall the discussions of OSI applications layer
  - File Transfer - FTP
    - File transfer protocol
  - Electronic Mail - SMTP
    - Simple mail transfer protocol
- Later additions
  - Mapping of host name to network address - DNS
    - Domain Name Service
  - Moving news articles - NNTP
    - Network news transfer protocol
  - Interface to the WWW - HTTP
    - Hypertext transfer protocol

Data Communication Services

- We've looked at
  - Hardware
  - Software
  - Several reference models
- Recall *network service*
  - Set of primitives or operations
    - A layer offers to its users
    - Represents a set of tools to get a job done

Let's briefly examine several communication services

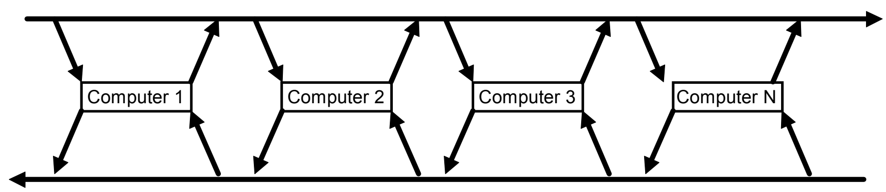
Introducing the acronyms - Gotta do that

DQDB - Distributed Queue Dual Bus
SMDS - Switched Multimegabit Data Service
X.25 - X.25
Frame Relay - Frame Relay
B-ISDN - Broadband Integrated Services Digital Network
ATM - Asynchronous Transfer Mode

DQDB - Distributed Queue Dual Bus

Mentioned earlier

Computers are interconnected via two unidirectional buses



Message traffic to the

Right proceeds on the upper bus

Left proceeds on the lower bus

SMDS - Switched Multimegabit Data Service

Designed by Bellcore as an internet service

Interconnect multiple local area networks - LANs

Broadband network - High Speed Data Service

Broadband as many interpretations

Telephony

Anything wider than 4KHz

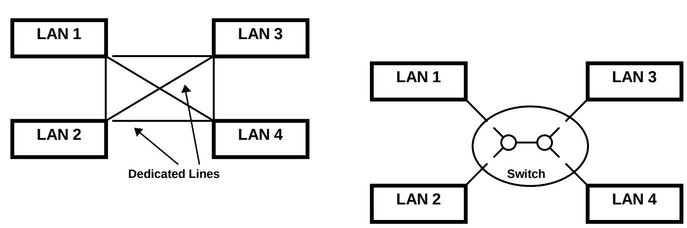
Generally assumed to support higher frequency transmission

Up to 10's to 100's of megahertz

SMDS - 45 Mbps

Switched Service

Lines not dedicated but switched as needed



- For above system
  - To support full interconnectivity among all LANs
    - Require 6 dedicated lines
  - Unless high utilization
    - The cost of dedicated system does not make sense
  - Alternative
    - Connection on demand
    - When needed - make connection
      - LAN via dedicated line to switch
      - Above with SMDS requires 4 lines
        - Eliminate 2 lines
    - Such a system is designed to handle bursts of data

Connectionless service with *packet* transmission
- Packet order not guaranteed
- Packet format
  - Up to 10K bytes
    - 16 address bytes
      - 8 byte destination address
      - 8 byte source address
      - Format
        - 4 bit code
        - 15 decimal digit telephone number
          - Country code
          - City / area code
          - Local number
          - Each digit is encoded in a 4 bit field
    - 9188 data bytes
      - Maximum
      - No restrictions on the content of the data
        - Bytes may contain any arbitrary protocol

Dynamic Message Length
- Each access point contains a counter
  - Continuously incremented at constant rate
- When a packet arrives
  - Length in bytes compared to counter value
    - If less
      - Transmit
      - Subtract length from the count
    - Else
      - Discard

Example

Let counter increment at 10 μs rate
Every second will accumulate 100,000 counts

Average
Gives an average throughput of 100,000 bytes/sec

Burst
Assume a short message - 100 to 1000 bytes
Can assure full speed (45 mbs) if transmit every 1ms to 10ms

X.25

International standard developed in Europe early 70s by CCITT
Comite Consultatif International Telegraphique et Telephonique

Designed to provide an interface between
Public packet switched networks
Users

Digital transmission rather than analog
Most telecommunication signaling is analog based
Has slowed acceptance

Connection oriented service with *packet* transmission
Packet order guaranteed
Transmission commences after establishment of a (virtual) circuit
Packets up to 128 bytes

Circuit Connection

Switched virtual
Source - Make a request to communicate
Connection established
Communication proceeds
Link relinquished

Permanent virtual
Connection established in advance
Not relinquished when transaction complete
Not a physical nor dedicated connection
Designed for burst data transmission

Frame Relay

- Low end service
  - Provides a means to identify
    - Start and End of a frame
  - Error detection
    - Discard on error
    - Error recover is user’s responsibility
  - No flow control

  - Packet order guaranteed
  - Reception not acknowledged

  - <u>*Permanent virtual*</u>
    - Connection established in advance
    - Usually between 2 points
    - Can have a one to many configuration
      - Each virtual circuit identified by a 10 bit number
        - Included in each transmitted frame
  - Packets up to 1600 bytes

ATM - Asynchronous Transfer Mode

- As the name suggests
  - ATM is the opposite of STM
    - Synchronous Transfer Mode
- Error Management
  - No packet acknowledgment
  - No retransmission on error
  - Can be implemented on higher layer

- STM
  - Most commonly used for reliable long distance
    - Voice and data transmission
  - Works as follows
    - Bandwidth of the link divided into *time slots* or *buckets*
      - *Time Division Multiplexing* - TDM
    - Buckets are collected together in a group like a linked list
    - Thus

T_0	T_1	T_2	…	T_{n-1}

  - USB uses a similar scheme
    - On the channel
      - The group is repeated every T sec
      - There can be up to *m* different groups

- Such a configuration gives a total of n*m time slots
  - When a connection is established
    - It is assigned one of the time slots
    - For the duration of the transmission
      - Data for that connection placed in that time slot
      - Time slot occupied even if no data transmitted
  - Maximum number of connection therefore
    - n*m
  - Values of T, m, and n
    - Set up by international standards committees

- The telecommunications companies are typically using
  - Fiber optic cross country and cross oceanic links
    - With Gigabit/sec speeds
  - Would like to carry or need to carry
    - ✓ Real time traffic
      - Voice and hi-resolution video
        - Which can tolerate some loss but not delay
    - ✓ Non real time traffic
      - Computer data and file transfer
        - Which may tolerate some delay but not loss

    - Problem
      - Peak bandwidth requirements
        - May be quite high
          - As in high-resolution full motion video
      - The duration for which the data
        - May be quite small
      - That is
        - The data comes in bursts
        - Must be transmitted at the peak rate of the burst
        - Average arrival time between bursts
          - May be quite large and randomly distributed.

      - For bursty connections
        - Waste of bandwidth
          - To reserve them a bucket at their peak bandwidth rate
        - When on the average
          - Only 1 in 10 buckets may actually carry the data.

      - STM becomes inefficient with increasing
        - Peak bandwidth of the link,
        - Peak transfer rate of the traffic
        - Overall burstiness of the traffic
          - Expressed as a ratio of peak/average

ATM

- Instead of always identifying a connection by bucket number
  - Carry the connection identifier along with data - think header
  - Keep the size of the bucket small
    - If any bucket got dropped
      - Not too much data lost,
      - Some cases could easily be recovered
- Similar idea to packet switching

- Fixed packet size
  - Arose out of motivation
    - To sustain the same voice quality as in STM networks
    - But in the presence of some lost packets

- Two end points are associated with each other via
  - Virtual Circuit Identifier - VCI label
    - Instead of by a time-slot or bucket number
  - VCI is carried in the header portion of a packet
    - The packet is carried in the same type of bucket as STM

- The terms fast packet, cell, and bucket
  - Used interchangeably refer to the same thing.

- Fast packet switching attempts to solve
  - Unused bucket problem of STM

- Technique
  - Statistically multiplex several connections on the same link
    - Based on their traffic characteristics

  - If a large number of connections are very bursty
    - Assign all to the same link
    - Theory
      - Statistically they will not all burst at the same time
      - If some of them do burst simultaneously
        - There will sufficient elasticity that the burst
          - Can be buffered up
          - Put in subsequently available free buckets
  - Called *statistical multiplexing*

Connection oriented service with *packet* transmission
- Packet order guaranteed
- Reception not acknowledged
- Packet format
  - Up to 53 bytes
    - 5 header bytes
      - 3 byte VCI label
      - 1 Control field
      - 1 byte Header checksum
      - 4 byte Adaptation layer
        - Optional
    - 44 or 48 data bytes - based upon adaptation layer bytes
      - Maximum
      - No restrictions on the content of the data
        - Bytes may contain any arbitrary protocol

Designed with the goal of replacing most of the existing
- Analog networks
- Provide a means to transmit higher demand signals
  - High resolution / full motion video
  - Computer data and large files

Key technology is the ATM transmission
The ISDN reference model is different from those examined thus far

ISDN Model
- Three layers - three dimensions
- Broken down very well in an object centered way
- Tasks and responsibilities are collected into related
  - Layers
  - Planes

- Layers
  - <u>*Physical Layer*</u>
    - Physical medium including voltages and timing
    - Places no restrictions on the actual hardware
      - Copper OK
      - Aimed towards fiber implementation

ATM Layer

- Concerned with
  - Packets and packet transport
  - Management of virtual connections
  - Flow and congestion control

ATM Application Layer

- Accommodates data groupings larger than the specified packet size
- Source
  - Breaks data into appropriately sized packets
- Destination
  - Reassemble back into original

Physical and Application layers

- Divided into two pieces
  - Interface to higher layer
  - Collection of routines to do the work of the layer

The tasks of managing the data transport and connection process

- Separated into to groups that sit above or to the side
  - The reference hierarchy
- Denoted the
  - User plane and the control planes

User Plane

- Data transport
- Flow control
- Error management

Control Plane

- Manage the connection / disconnection processes

Implementing a Local Message Exchange

Accompanying diagram adds basic network capability

- To support simple message based exchanges

When messages are exchanged

- Within local system
- Between local system and peripheral devices
- Receiving device must be able to
  - Accept incoming stream of information
  - Detect and identify
    - Start and end of a bit
    - Start and end of a character
    - Start and end of a message block or frame

These are known as

- *Bit*
- *Character*
- *Frame synchronization*

Different transmission modes
Give rise to two general categories of message exchange
Asynchronous transmission
Receiver resynchronizes at start of each bit
Synchronous transmission
Receiver resynchronizes
Continuously based upon encoded clock edge transitions
Start of each block or frame

Asynchronous Transmission
Characterized by irregular intervals between transmitted data groups
Inter character spacing may vary widely
May be bursts of activity
Followed by long periods of inactivity

Synchronous Transmission
When blocks of regularly spaced data
Transferred over a serial line
Transmitter and receiver can be synchronized to common clock
Permitting character transfer at much higher rate

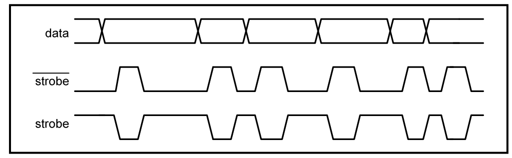
Generally synchronous transfer requires less overhead
Therefore is more efficient than asynchronous design

Asynchronous Exchange

Because no inherent clock associated with asynchronous exchange
Coordination and synchronization accomplished
Using protocol that permits (re)synchronization of data
To receiving system's internal clock
Both the protocols and amount of data exchanged can vary tremendously

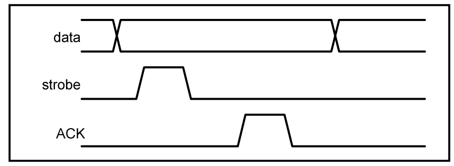
Because no clock
Exchanges co-ordinated using some form of handshaking protocol
Such protocols can be simple or complex
Typical examples given in following diagrams

Basic Strobe



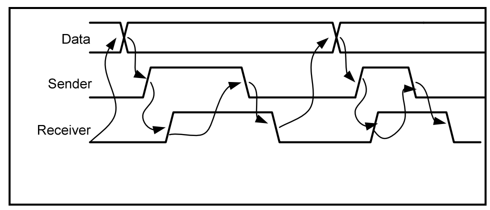
- Strobe associated with each data word
  - No acknowledgment of acceptance
- Observe
  - Strobe can be of either polarity

Strobe with Acknowledge



- Strobe associated with each data word
  - Each data word acknowledged
    - With return strobe

Full Handshake

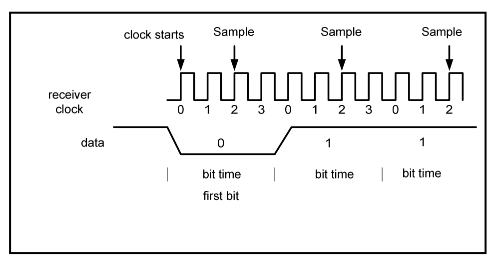


- Confirms all phases of the exchange
- Ready for data
  - Here’s data
- I’ve got it
  - OK
- Exchange looks like this

Resynchronization

- Another form of synchronization protocol
  - Entails generating sampling signal on receiver side based upon
    - Knowing the transfer rate
    - When the first data bit has arrived
- Such an approach is known as *bit timing*

- Approach works as follows
  - Transmitter and receiver timed by independent clocks
  - To capture incoming signal reliably - one must know
    - Length or duration of a bit
      - When transmission starts
  - In the ideal situation
    - Signal is sampled in center
      - Gives maximum tolerance for errors
        - On either side of signal



  - Scheme illustrated in accompanying diagram,

  - Initial agreements between sender and receiver
    - Bit time and message length
    - Idle state of data line will be a logical 1
    - Transition from logical 1 to logical 0
      - Will signify the start of a transmission

In current design

We select receiver clock with a period one fourth of bit time

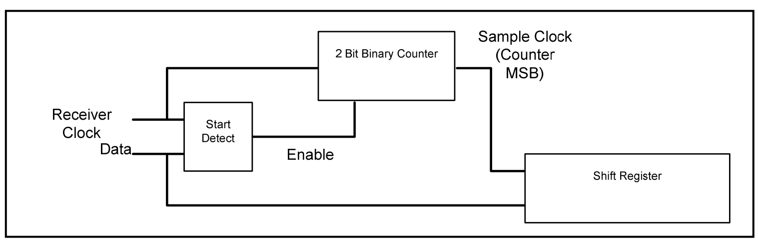
There are four receiver clocks during each bit time

Exchange proceeds as follows

- ✓ Start of a character signaled by transition of data line
  - From logical 1 to logical 0
- ✓ Receiver clock started
- ✓ Number of receiver clocks is counted
  - Based upon the agreed upon timing
    - Known that the falling edge of the second clock
      - Will be in center of data bit
  - At this point incoming data bit can be stored
- ➢ Falling edge of the sixth clock pulse
  - two clocks + four clocks

  Will occur in the center of the second data bit
  - Which can now be stored
- ➢ Process repeated until all data received

Implementing hardware given as



Analysis of Asynchronous Exchange

Potential problems

- Difficult to test
- Clock noise more difficult to filter out
- Potentially complex protocol
  - To identify start / end of transmission

Potential advantages

- Devices may run at different / differing speed
- No clock skew on long busses

Synchronous Exchange

There are several drawbacks of the asynchronous transmission schemes

- Extra overhead of control bits
- Bit clock synchronization scheme less reliable at higher data rates

Problems can be mitigated with synchronous transmission
Still must achieve
Bit, character, frame synchronization
Frame synchronization usually derived from the former
Generally includes clock in control lines

Exchanges between sender and receiver
Synchronized to the clock

- Directly
- Signals derived from the clock
  Manchester phase encoding

- Serial Exchange
  Clock either separate or encoded in data
- Parallel Exchange
  Clock one of control lines

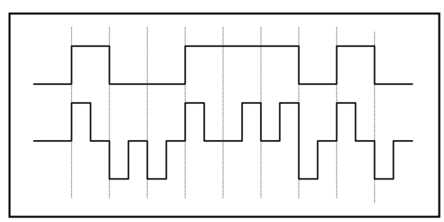
Bit Synchronization

To achieve bit synchronization
Two step typically used

- Encode the clock in the data
- Re-derive the clock from the data

Encoded Clock

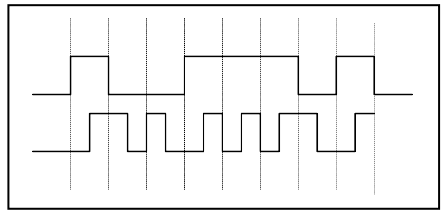
Three different methods generally used



Bipolar Encoding

Binary 0's and 1's are represented by
Different polarity signals
Each bit cell contains clocking information
Observe
Signal returns to zero level after each encoded bit
Referred to as *return-to-zero RZ* signal
Scheme requires 3 distinct signal levels

Manchester Phase Encoding



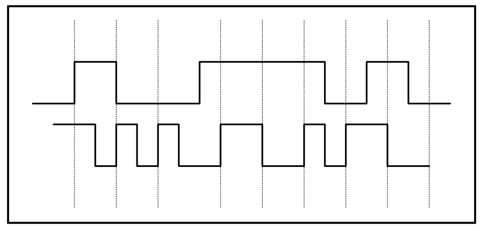
Binary 0 - high to low signal transition
Binary 1 - low to high signal transition

- Transition in the center of each bit cell
  - 0 → 1 // indicates a 1
  - 1 → 0 // indicates a 0

  Provides the clock information

Observe
- Signal does not return to zero level after each encoded bit
- Referred to as *non-return-to-zero NRZ* signal

Manchester Differential Encoding



- Transition in the *center* of each bit cell
  - 0 → 1
  - 1 → 0

  Provides the clock information

- Transition
  At the *start* of each bit cell
  - Only if next bit to be encoded is a binary 0

  Otherwise in *center*

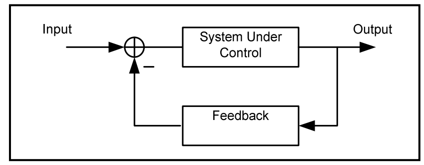
Re-derive the Clock

To re-derive clock from data
- Transmission begins with a preamble
  - Including a synchronization sequence
- A *phase locked loop - PLL*
  - Based upon a very stable receiver clock
  - Used to keep sample clock locked to incoming signal transitions
  - Data must be encoded
    - To ensure a sufficient number of signal transitions
      - To retain synchronization
      - At each transition
  - Sample timing adjusted to ensure sampling in center of bit

Design will tolerate intervals without transitions
- Provided there is a stable fundamental clock

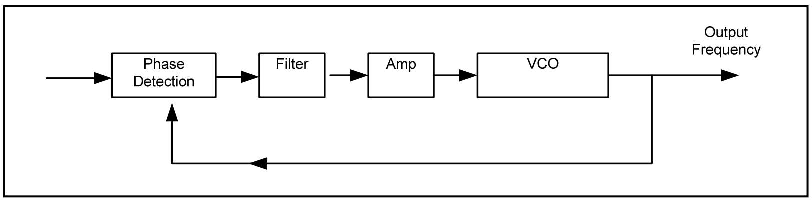
PLL is a conventional closed loop control system
- With some additions and modifications

Basic structure for a closed loop system given accompanying diagram



For traditional analog closed loop control system
- Input is algebraically added to a signal that is
  - Modified and fed back from system output
- Result is an error signal
  - That serves as an input to the system under control
- Objective is to drive the error signal to zero

For phase locked loop
- Objective is to control a frequency
- Basic block diagram appears as



- Output of the PLL generated
  - Using a voltage controlled oscillator (VCO)
- Output of VCO is fed back to a phase detector
  - That compares
    - Input signal characteristics with
    - Those of signal being fed back
  - When the frequency and phase difference
    - Between input signal and VCO output is zero
      - System has *locked* onto input frequency
  - Difference in frequency and phase appears as an error voltage
    - ✓ Error signal is filtered by the low pass filter
    - ✓ Amplified
    - ✓ Provides an input voltage to the VCO
  - Output of VCO can now serve as clock to the system
  - Input signal to the PLL is
    - Any of the encoded data streams

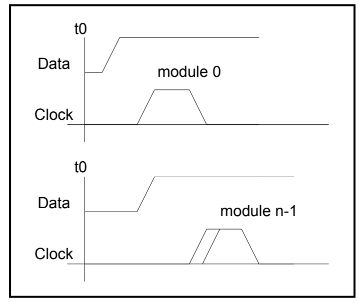
Use a preamble including sync sequence
Done using *phase lock loop- PLL*
Based upon very stable receiver clock

PLL used to keep sample clock locked
- Signal transitions of incoming signal

Data encoded to ensure a sufficient number of signal transitions
At each transition sample timing adjusted to ensure
- Sampling in center of bit

Scheme will tolerate intervals without transitions
Based upon stability of the clock

Bit / Character / Frame Synchronization
Once individual bits are identified
Character and frame synchronization rather straight forward



Analysis of Synchronous Exchange
Potential problems

- All devices must run at same speed
- Can have different propagation delays along a bus

Because of (potentially) different loading on clock vs. data lines

- Clock skew on long busses

Because of different loading
Have propagation delay along bus
Possibility that clock will arrive at the various destinations
At different times with respect to the data
As illustrated
One can easily get clock skew (with respect to the data)
On long busses
Particularly at higher clocking frequencies

Potential advantages

- Easier to test
- Generally protocol is simpler that asynchronous approach
- Easier to stay in sync with data

I^2C – A Local Area Network

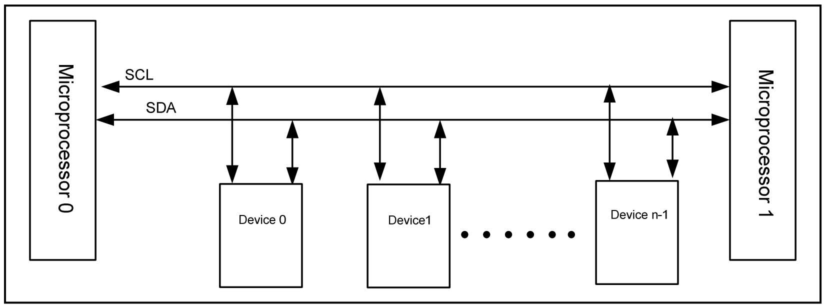
The I^2C bus – Inter Integrated Circuit Bus
Developed in the 1980's by Philips Semiconductor
Means of supporting communication amongst a set of chips
Internal to a specific system
At its initial introduction
Bus was intended for small, lower speed systems

Initial bit rate of 100 K bits/sec has increased to 400 K bits/sec today

In addition to its low cost and ease of implementation
I^2C bus offers several interesting features

The Architecture

The I^{2}C bus utilizes a simple serial two wire *multi-master-slave* architecture
As illustrated in the high level block diagram



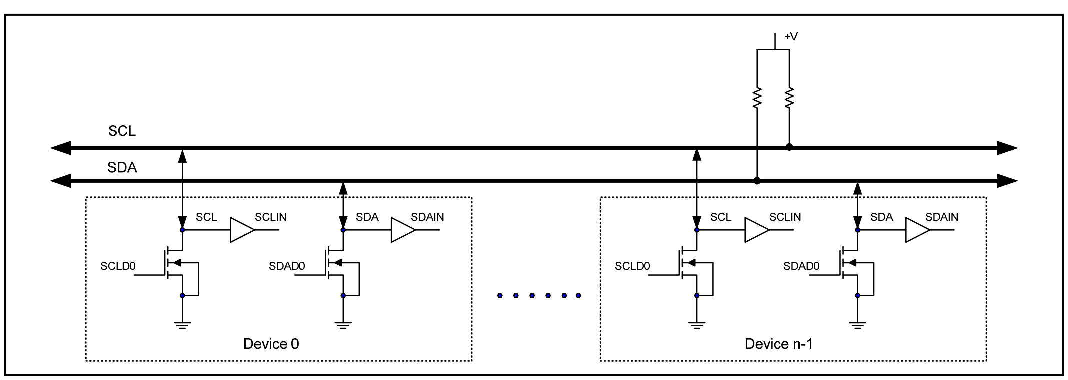
Typically devices with bus master capability are microprocessors
But, they need not be
Independent of bus master capabilities
Any device on the bus has the potential to be
Sender or receiver – source or destination – of a transaction

The bidirectional I^{2}C bus implements *wired AND* signaling
Only special interface circuitry required to connect a device onto bus
Two open drain or open collector devices
That enable the device to pull either line to ground

Each device on the bus has a unique address
Independent of physical device type
Yet the device type is embedded into the address.

In addition to ground bus comprises two signal lines as illustrated
serial clock – SCL
serial data – SCD

The drawing reflects the details of the bus and its connection to several devices

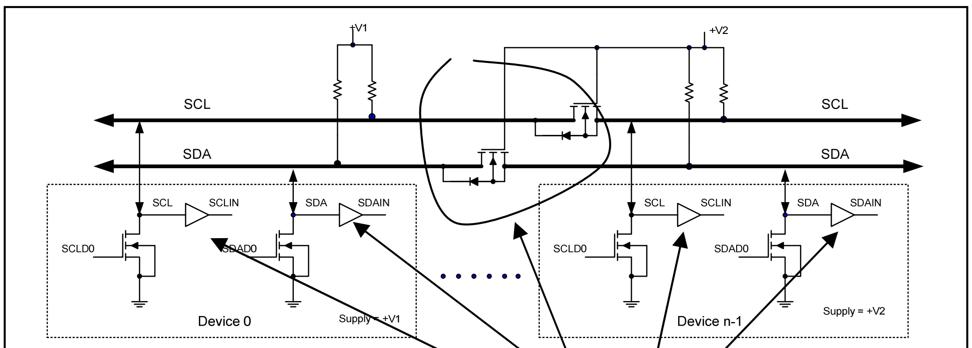


Electrical Considerations

The design of the I^2C bus
    Places no restrictions on the type of devices that may be connected to the bus
        Similar to IP portion of TCP/IP

Can configure a system with a mixture of various TTL and CMOS families
    Each with its own different supply voltage
The different voltage levels found in logic devices
    Operating at either 5.0 V or 3.3 V

To accommodate bidirectional communication with such devices
    Operating on less than 5.0 VDC



Recommended that a simple buffering circuit be incorporated
    Into the SCL and SDA signal paths as shown
The two subsystems operating at different voltage levels
    Now separated from one another

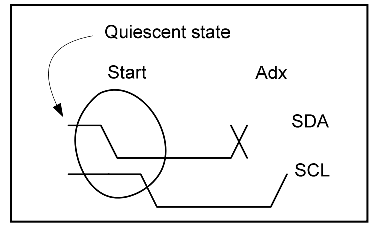
To support standard (100 K bits/sec) and fast mode (400 K bits/sec) transfer rates
    Total bus capacitance must be less than 400pf
        Similar to USB spec
    Signal rise and fall times at 1000ns and 300 ns
These are measured at the 30% and 70% points on the signal
    Rather than the traditional 10% and 90% points

Such constraints place a limit on the number of devices
    That can be interconnected on the bus

Basic Operation

Let's now take a look at the basic bus operation in a system comprising
Single master and several slaves
Look at both a master read and a master write operation

The quiescent condition for both the SCL and SDA lines is the high state



- A bus cycle
  - ✓ Begins with a *Start* condition
  - ✓ Ends with a *Stop*

  These are always generated by the master

A *Start* is signaled
When the master causes a HIGH to Low transition on the SDA line
While holding the SCL line in the HIGH state
A *Stop* is signaled by a LOW to HIGH transition on the SDA line
While holding the SCL line in the HIGH state

An I^2C *address* comprises 7 bits
- The four most significant bits (A7-A3)
  Identify the category of the device being addressed
- The three least significant (A2-A0)
  Identify a programmable hardware address assigned to the device

Thus, up to eight instances of the same type of device
Can be included in the system

For example if a system included eight serial EEPROMS
Each would have
The base address 1010
Concatenated with one of the addresses 000..111

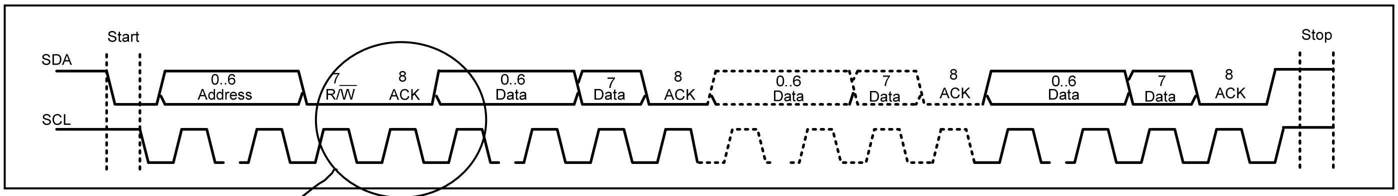
Data is sent most significant bit first
- ✓ Each bit is accompanied by a clock signal
- ✓ Can only change when the SCL line is in the LOW state
- ✓ Is transferred as an unrestricted number of bytes
  - Each byte must be acknowledged by the receiver

A transaction may be
A *write* operation – master to slave
A *read* operation – slave to master
A combination in which there is a change in transmission direction
During a transaction

The contents of a complete data transfer cycle given as

Start
Slave Address
Read / Write
Acknowledge
Data – Acknowledge (the pair is repeated as necessary)
Stop

The message format is given as



For a *Read* operation
The *Read/Write* bit will be a HIGH
For a *Write*
The *Read/Write* bit will be a LOW

Following the *ACK*
Which is generated by the slave and appears in bit 8
If the ensuing operation is to be a *read*
The master changes roles from transmitter to receiver
The slave from receiver to transmitter
Despite the reversed roles
Master still generates the *Stop* and manages the end of the transaction

Flow of Control

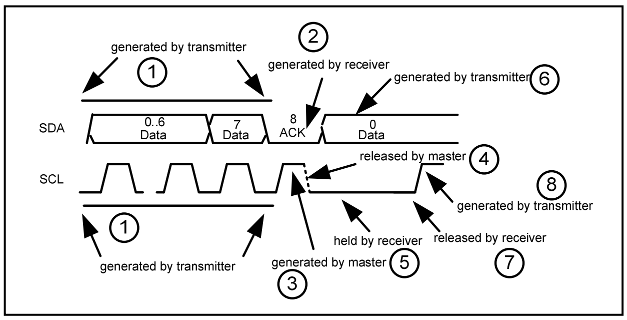
✓ The transaction protocol requires each data byte be acknowledged
✓ The clock pulse associated with the acknowledge (ACK*)*
Generated by the master
✓ During that clock time
- The transmitter – which may not be the master
Slave → Master, Read Operation
Must release the SDA line allowing it to float
- The receiver must pull the SDA line LOW
For the duration that the clock pulse on the SCL line is in the HIGH state
This is an ACK

Under normal circumstances following the ACK bit time
The master will release the SCL line
So that transmission may continue with the next byte

If receiver is temporarily unable to proceed
It will hold the SCL line LOW thereby extending the ACK interval

When able to proceed again
The receiver will release the SCL line and transmission continues

The timing diagram illustrates the ACK interval extension



Following the end of a communication session with one slave
The master typically will issue the *Stop* directive to end the session
If it wishes to establish a connection with different slave
Rather than issue the *Stop*
The master will issue another *Start*
Using the address of the new device

Multiple Masters

Several problems can occur when a system has multiple masters
The first arises if two or more masters try to talk at the same time
Resolved by *arbitration*
The second results from having multiple clocks in the system
Resolved by *synchronization*.

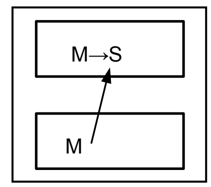
Arbitration

A (note a not the) master can initiate transfer cycle only if the bus not in use
As noted such a cycle begins
When the master places the SDA line in the LOW state … the start
While keeping the SCL line in the HIGH state
At this point if multiple masters simultaneously issue a Start
There is no way to distinguish amongst them
Thus arbitration can only begin
At the most significant bit of the address…which follows

Each device attempting to communicate on the bus
Drives the SDA line
The wired AND aspect of the bus inherently gives priority to
Device driving the SDA line to the LOW state

Each master
Is monitoring the SDA line
Knows the state of the bit it has put onto the line

If a master sees the SDA line in the LOW state
Knowing that it has set a HIGH
It will disable its data drive capability and back-off



The arbitration process can continue bit by bit
For a number of SCL clock cycles until resolved

If the loosing master(s) also incorporates a slave function
Possibility exists that the winning master is trying to address that slave
The loosing master(s) must immediately switch into the receiver mode

Synchronization

Each master typically
Has its own internal clock
Is responsible for generating the SCL clock for the bus
Since each of the master clocks is asynchronous with respect to the others
For the arbitration scheme to work properly
Clocks must be synchronized

The operation also takes advantage of the wired AND connection scheme
A HIGH to LOW transition on SCL line from any of the masters
Directs each other master to place its SCL driver in the LOW state
Begins timing the LOW duration for its SCL clock
When that interval expires on one of the masters
That device places its SCL driver in the HIGH state
However the SCL line may not follow
If some other master(s) is (are) still timing its LOW interval

As each master completes timing the LOW SCL interval
It places its SCL driver in the HIGH state
Enters a wait state until the last device
Releases the SCL line and it enters the HIGH state

At that time each master
Begins timing the HIGH duration for its SCL clock
In similar manner the last device to complete timing its HIGH duration
Will change the state of the SCL line to LOW

Based upon such a scheme, its evident that for each master
The LOW intervals will be the same as will the HIGH intervals

The LOW intervals will be set by the device with longest interval
The HIGH intervals will be set by the device with the shortest interval

Using the I^2C Bus

Today there is a great variety of integrated circuit devices including
Different microprocessors and microcontrollers that support the I^2C bus
Such devices range
From
Displays and serial EEPROMS
To
Analog to digital converters and video acquisition systems

The bus can provide a highly effective lower speed network
For locally distributed embedded applications
The extent of the topographic distribution is subject to
The 400 pf capacitive loading specification
Such a constraint suggests
20-30 devices and a maximum signal path of approximately 10 meters

Summary

Have looked at a general overview of networks

- ✓ Began with an overview of the hardware and software of basic networks.
- ✓ Introduced the network architecture and protocol stack.
- ✓ Identified the main components of a message structure.
- ✓ Introduced the basic reference models.
- ✓ Compared and contrasted the OSI and TCP/IP models.
- ✓ Developed a simple local message exchange.
- ✓ Examined synchronous and asynchronous message exchange.
- ✓ Examined the I^2C bus.
- ✓ Learned some of the jargon and acronyms