Cyber Systems Overview

Introduction to Cyber Systems (Danmarks Tekniske Universitet)

CYBER SYSTEMS OVERVIEW

EMBEDDED SYSTEMS OVERVIEW

Embedded systems are specialized computer systems built into various devices to perform specific functions. Unlike general-purpose computers, they are designed for handling particular tasks within the devices they control. These systems are crucial for ensuring that devices operate efficiently and reliably. You can find embedded systems in a wide range of applications, from household appliances like microwaves and washing machines (we can find around 50 per household) to complex systems in cars and medical devices.

Embedded systems are really important for modern technology, providing specialized computing power where it's most needed, these systems are also optimized for specific tasks, ensuring efficient operation with minimal power usage, embedded systems use targeted components to reduce costs, making technology accessible to more people.

CHARACTERISTICS

Embedded systems are specialized computing units within larger devices, characterized by several distinct attributes:

1. **Single-Functioned**: Each system performs a specific function and is designed to execute one particular task effectively within its host device.
2. **Tightly-Constrained**: These systems often operate under strict constraints, including limited processing power, memory, and energy resources, requiring highly efficient programming and hardware design.
3. **Reactive**: Embedded systems continuously respond to changes in their environment or to user inputs, making them reactive to external events.
4. **Real-Time**: Many embedded systems operate in real-time, meaning they must complete their tasks within strictly defined time constraints. This is crucial in applications where delay could cause failure or hazards, such as in automotive braking systems or pacemakers.

OPTIMIZING DESIGN METRICS

- **Unit Cost**: The monetary cost of producing each copy of the system, excluding the NRE cost.
- **NRE Cost**: The one-time monetary cost of designing the system.
- **Size**: The physical space required by the system.
- **Performance**: The execution time of the system.
- **Power**: The amount of power consumed by the system.
- **Flexibility**: The ability to change the functionality of the system without incurring heavy NRE costs.
- **Time-to-Prototype**: The time needed to build a working version of the system.
- **Time-to-Market**: The time required to develop a system to the point that it can be released and sold to customers.
- **Maintainability**: The ability to modify the system after its initial release.

BINARY CODE

UNSIGNED INTEGERS: $0 <-> 2^n - 1$

SIGN MAGNITUDE: $-2^{n-1} <-> 2^{n-1}$

2's COMPLEMENT: $-2^{n-1} <-> 2^{n-1}-1$

MODELS AND LANGUAGES

- The **computational model** describes the system's behavior (state machine model)
- **Languages** capture models (C, C++, R)

COMMON COMPUTATION MODELS

Embedded systems utilize various programming models to address different functional and operational requirements. Here are some of the most common models:

1. **Sequential Program Model**: This model operates through a series of statements executed in a predefined order. It includes specific rules for composing and executing statements, ideal for tasks that require a straightforward, step-by-step process.
2. **Communicating Process Model**: This model involves multiple sequential programs running concurrently, allowing systems to handle multiple tasks simultaneously. It's particularly useful in environments where processes must interact or communicate effectively.
3. **State Machine Model**: Designed for control-dominated systems, this model monitors control inputs and sets control outputs accordingly. It's used in scenarios where system behavior can be distinctly defined by states and transitions, such as in automated ticketing systems or robotic controls.
4. **Dataflow Model**: Suitable for data-dominated systems, the Dataflow model processes and transforms input data streams into output streams. This model is effective in handling large volumes of data, such as in signal processing or real-time analytics systems.
5. **Object-Oriented Model**: This model breaks complex software into simpler, well-defined pieces or objects, which can interact but are self-contained. It's particularly useful in large, complex systems where modularity, reuse, and encapsulation are priorities, such as in software development for consumer electronics.

Each model provides unique advantages, making them suitable for specific types of embedded systems applications, enhancing efficiency, and reducing development complexity.

FINITE-STATE MACHINE (FSM)

An FSM is a computational model used to design and analyze systems that have a limited number of specific states. It operates by transitioning from one state to another in response to external inputs. An FSM is defined by: A finite set of states, a set of inputs that trigger state transitions, and a set of outputs. For Mealy machines, outputs depend on the state and input; for Moore machines, outputs depend only on the state.

- An FSM is a 6-tuple **<S, I, O, F, H, s_0>**
  - **S** is a set of all states $\{s_0, s_1, \ldots, s_l\}$
  - **I** is a set of inputs $\{i_0, i_1, \ldots, i_m\}$
  - **O** is a set of outputs $\{o_0, o_1, \ldots, o_n\}$
  - **F** is a next-state function ($S \times I \rightarrow S$)
  - **H** is an output function ($S \rightarrow O$ *or* $S \times I \rightarrow O$)
  - **s_0** is an initial state
- Moore-type
  - Associates outputs with states (as given above, **H** maps **S** → **O**)
- Mealy-type
  - Associates outputs with transitions (**H** maps **S x I** → **O**)
- FSMs use only Boolean data types and operations, no variables

FINITE-STATE MACHINES WITH DATAPATH (FSMDS)

An FSMD extends the concept of an FSM by integrating a datapath that includes data storage elements like registers and accumulators, and operational components like arithmetic logic units (ALUs). Unlike a basic FSM, an FSMD can perform arithmetic and logical operations and store temporary data. This integration allows FSMDs to handle more complex control and data manipulations, making them suitable for applications that require both control flow and data processing.

- FSMD: 7-tuple **<S, I, O, V, F, H, s_0>**
  - **S** is a set of states $\{s_0, s_1, \ldots, s_l\}$
  - **I** is a set of inputs $\{i_0, i_1, \ldots, i_m\}$
  - **O** is a set of outputs $\{o_0, o_1, \ldots, o_n\}$
  - **V is a set of variables $\{v_0, v_1, \ldots, v_n\}$**
  - **F** is a next-state function (**S x I x V** → **S**)
  - **H** is an action function (**S** → **O + V** or **S x I** → **O + V**)
  - **s_0** is an initial state

Moore *Mealy*

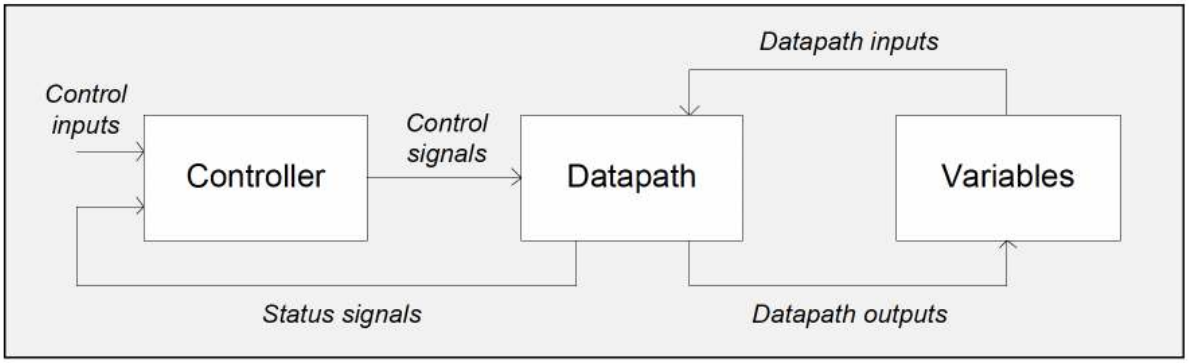
FSMD extends FSM with complex data types and variables for storing data

FSMD BLOCK DIAGRAM

The FSMD computational model is a mathematical abstraction that describes a computing system's behavior and functionality as:

–**CONTROLLER:** controls the program flow

–**DATAPATH**: performs data processing operations



STATE MACHINE VS. SEQUENTIAL PROGRAM MODEL

State machine vs. sequential program model

- Different thought process used with each model
- State machine:
  - Encourages designer to think of all possible states and transitions among states based on all possible input conditions
- Sequential program model:
  - Designed to transform data through series of instructions that may be iterated and conditionally executed

COMPILATION PIPELINE

Source code (human readable)

```
Inputs: int floor; bit b1..bN; up1..upN-1;
        dn2..dnN;
Outputs: bit up, down, open;
Global variables: int req;

void UnitControl()
{
   up = down = 0; open = 1;
   while (1) {
      while (req == floor);
      open = 0;
      if (req > floor) {up =
1;}
      else {down = 1;}
      while (req != floor);
      up = down = 0;
      open = 1;
      delay(10);
   }
}
```

Machine code(not human readable)

```
0x32c4 :      addil 0,dp
0x32c8 :      ldw 0x22c(sr0,r1),r26
0x32cc :      ldil 0x3000,r31
0x32d0 :      ble 0x3f8(sr4,r31)
0x32d4 :      ldo 0(r31),rp
0x32d8 :      addil -0x800,dp
0x32dc :      ldo 0x588(r1),r26
0x32e0 :      ldil 0x3000,r31
0x32c4 :      addil 0,dp
0x32c8 :      ldw 0x22c(sr0,r1),r26
0x32cc :      ldil 0x3000,r31
0x32d0 :      ble 0x3f8(sr4,r31)
0x32d4 :      ldo 0(r31),rp
0x32d8 :      addil -0x800,dp
0x32dc :      ldo 0x588(r1),r26
0x32e0 :      ldil 0x3000,r31
```

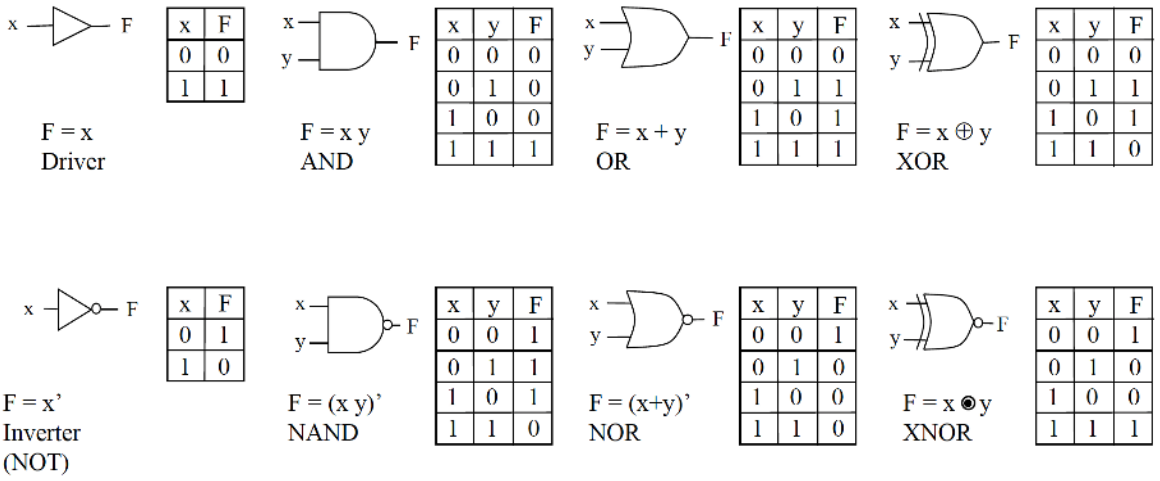
Solution:
Source code is human friendly, machine code is machine friendly and not (less) human readable.
The compilation pipeline transforms source code into machine code.

PRE-PROCESS: The pre-processing step occurs before the actual compilation. The preprocessor handles directives, which are instructions that begin with `#` in the source code. These directives include: `#include` (for including other source files or libraries), `#define` (for creating macros) and `#if`, `#else`, `#endif` (for conditional compilation).

COMPILE: Transformation of computer code written in one programming language (source language) into another computer language (target language).

LINK: The linking stage takes one or more object files generated by the compiler and combines them into a single executable program.

BASIC LOGIC GATES



TYPES OF LOGIC

Combinational logic: Produce outputs solely based on the current inputs, with no memory of past inputs. These circuits immediately respond to changes in inputs, making them ideal for simple decision-making processes.

Applications:

- Basic operations like arithmetic calculations and data routing.
- Found in devices such as calculators and simple data selectors.

Sequential logic: Produce outputs based on current inputs and their historical input sequence, thus incorporating memory elements. They require a mechanism to remember previous states, influencing their outputs.

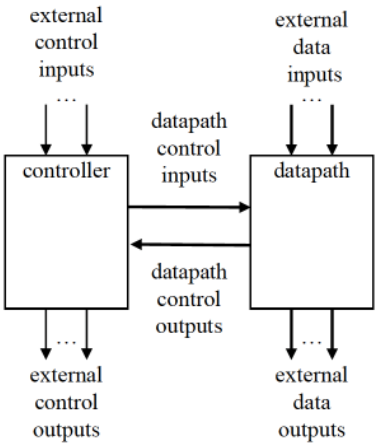
Characteristics: Often clock-driven, updating outputs at defined intervals.

Applications:

- Essential for tasks requiring memory of past events.
- Used in digital watches, computers, and various types of counters.

SINGLE-PURPOSE PROCESSOR

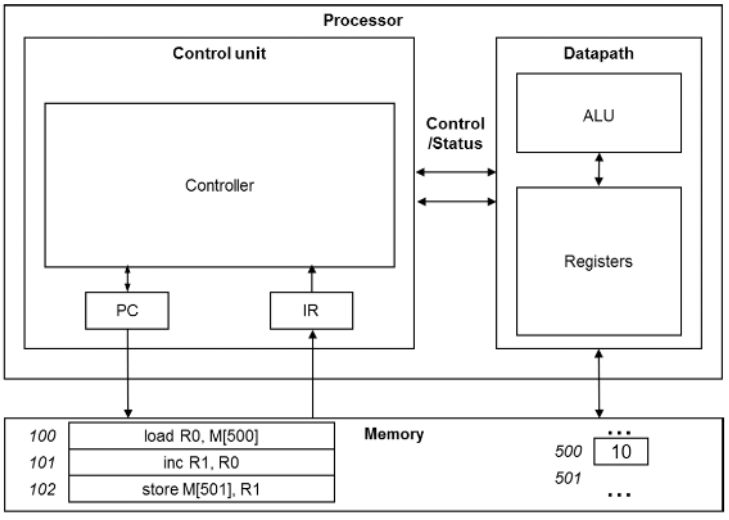
Single-purpose processors are tailored for specific applications, providing efficient, cost-effective solutions for tasks that do not require the computational flexibility of general-purpose processors. Their design allows them to perform designated functions more efficiently, with lower power consumption and at a reduced cost, making them ideal for many electronic products and industrial applications.



A single-purpose processor is a type of microprocessor that is designed to execute only one specific task or function. Unlike general-purpose processors that can run a wide variety of programs, single-purpose processors are optimized for a particular application, offering advantages in terms of speed, power efficiency, and cost for their intended task. EXAMPLE: MICROWAVE

GENERAL-PURPOSE PROCESSOR

A general-purpose processor (GPP) is a microprocessor designed to perform a wide array of tasks across different applications. Unlike specialized processors, general-purpose processors can handle diverse operations, making them suitable for devices like computers, smartphones, and servers. EXAMPLE: TRAFIC LIGHT



FUNDAMENTAL COMPONENTS OF A (CPU)

Control Unit: The control unit is responsible for directing the operations of the processor. It interprets the instructions collected from the computer's memory and generates signals to control how data moves within the CPU and what operations the datapath performs. (DIRECTS OPERATIONS WITHIN THE CPU).

Datapath: The datapath is a network of processor components that perform arithmetic operations, hold transient data and shift data between locations within the CPU. It typically includes components like the ALU, which performs mathematical and logical operations, and various registers, which store data temporarily during instruction execution. (PERFORMS THE OPERATIONS AND MOVES DATA)

Memory: This is where the computer stores data and program instructions that the CPU needs quick access to while operating. Memory is crucial for providing the necessary data and instructions to both the control unit and the datapath. (STORES THE DATA AND INSTRUCTIONS NEEDED FOR THIS PROCESSES)

STORING A PROGRAM

–Express program as a sequence of coded instructions

–Memory holds both data and instructions

–CPU gets, interprets, and executes successive instructions of the program

INSTRUCTION CYCLE (OF THE CONTROL UNIT):

–**Fetch**: Get the next instruction into IR

–**Decode**: Determine what the instruction means

–**Fetch operands**: Move data from memory to the datapath register

–**Execute**: Move data through the ALU

–**Store results**: Write data from the register to memory

INSTRUCTION SET ARCHITECTURE

The Instruction Set Architecture (ISA) is a crucial part of computer architecture that defines the specific code and associated syntax that a processor understands and can execute. The ISA serves as the interface between software and hardware, specifying the processor's capabilities and how it interacts with memory and other system components.

Instruction	Syntax (example)	Meaning (example)
Arithmetic instructions		
Addition	`ADD R1, R2, R3;`	`R1 = R2 + R3`
Subtraction	`SUB R1, R2, R3;`	`R1 = R2 - R3`
Bitwise OR	`OR R1, R2, R3;`	`R1 = R2 or R3`
Bitwise AND	`AND R1, R2, R3;`	`R1 = R2 and R3`
Bitwise NOT	`NOT R1, R2;`	`R1 = not(R2)`
Data transfer instructions		
Load immediate	`LI R1, 6;`	`R1 = 6`
Load data	`LD R1, R2;`	`R1 = memory(R2)`
Store data	`SD R1, R2;`	`memory(R2) = R1`
Control and flow instructions		
Jump	`JR R1;`	`go to R1`
Jump if equal	`JEQ R1, R2, R3;`	`if(R2==R3) go to R1`
Jump if less than	`JLT R1, R2, R3;`	`if(R2<R3) go to R1`
No operation	`NOP;`	`do nothing`
End execution	`END;`	`terminates execution`

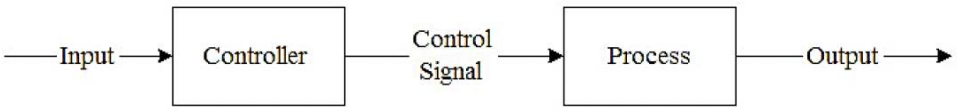
- The instruction-set architecture is basically machine language, it defines the set of elementary commands that a processor is able to perform
- It defines everything a machine language programmer needs to know in order to program a computer

CONTROL SYSTEMS

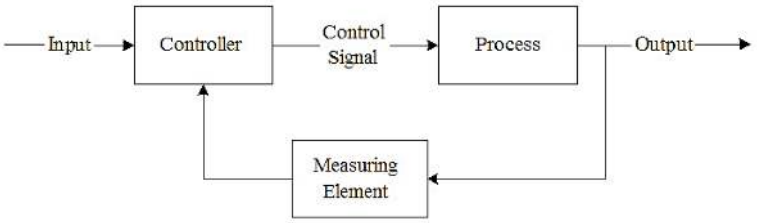
Control systems are used to manage, command, direct, or regulate the behavior of other devices or systems. They operate by continuously adjusting the performance of the target system to meet desired specifications or behaviors. To do this they set the system input to control the physical system output.

TYPES OF CONTROL SYSTEMS

1. **Open-Loop Control Systems**: These systems operate without feedback. They perform a specific function and do not adjust based on the output. An example is a microwave oven that cooks for a set time regardless of the food's temperature at the end.



2. **Closed-Loop Control Systems** (Feedback Control Systems): These systems use feedback to adjust the control actions based on the output. For instance, a thermostat maintains room temperature by measuring the actual temperature and adjusting the heating or cooling output as needed.



UNIVERSAL PID CONTROLLER

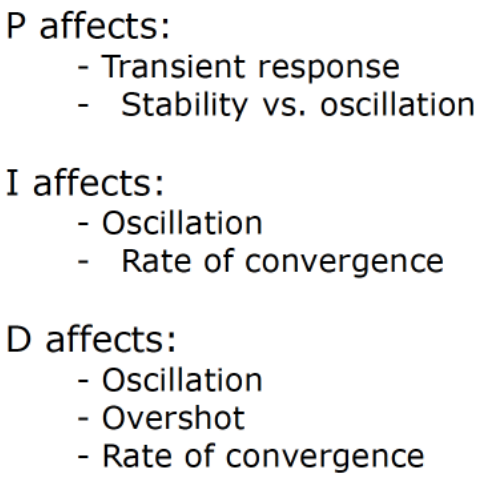
The Universal PID (Proportional-Integral-Derivative) controller is a widely used control mechanism that helps maintain the desired output level of various systems, ranging from industrial equipment to consumer electronics. It adjusts the system's output by calculating and combining three primary elements: proportional, integral, and derivative, which relate to the error between a desired setpoint and the actual output.

PID CONTROLLER COMPONENTS:

Proportional (P): Adjusts the output proportionally to the error. The larger the error, the larger the proportional response. This helps the system respond quickly to changes. Effect: Quickly reduces the error but can overshoot the target or leave a small persistent error.

Integral (I): Adds up past errors over time, helping to correct any ongoing, consistent deviation from the desired setpoint. Effect: Eliminates steady-state error (the remaining error after the system has settled) by adjusting the control output based on the cumulative sum of past errors.

Derivative (D): Reacts to the rate at which the error changes, effectively predicting future error based on its current rate. Effect: Dampens the response to reduce overshoot and improves system stability by smoothing out potential fluctuations.



UART

UART is a type of communication protocol used to allow two devices to talk to each other over a serial connection. It’s called "asynchronous" because it doesn’t use a clock signal to synchronize the transmission of data between devices.

Key Points:

1. UART sends data one bit at a time, sequentially, over a single wire.
2. Devices using UART must agree on the speed of communication and data format beforehand since there's no clock signal to keep them in sync.
3. The data sent includes a start bit, the actual data bits, and stop bits to mark the beginning and end of a message.

How It Works:

- A device sends out a data packet that starts with a start bit, followed by the actual data, and ends with stop bits.
- The receiving device looks for the start bit, reads the data bits at the agreed speed, and stops at the stop bits.

Usage:

UART is commonly used for simple, direct communication between devices like microcontrollers, computers, and other electronic devices that need to exchange information directly. It’s a basic but effective way to achieve device-to-device communication without needing complex wiring or synchronization mechanisms.

I2C

I2C is a communication protocol that allows multiple master and slave devices to communicate with each other using only two wires.

Key Features:

-Uses two lines, SDA and SCL
-Allows multiple master devices and multiple slave devices on the same bus.
-Each device connected to the bus has a unique address used for communication.

Usage: Often used for connecting low-speed peripherals like sensors or other components to microcontrollers in embedded systems.

SPI

SPI is a communication protocol known for its speed and efficiency in transferring data between a master device and one or more slave devices.

Key features:

-Typically uses four lines, MOSI, MISO, SCK and SS.
-Generally has one master and multiple slaves, with individual lines to select each slave.
-Allows simultaneous data transmission and reception.

Usage: Used in applications where devices need to communicate quickly and efficiently, such as in SD card interfaces, liquid crystal displays, and complex peripherals.

Comparison:

- **Wiring Complexity**: SPI typically requires more wires than I2C.
- **Speed**: SPI is generally faster than I2C.
- **Ease of Use**: I2C uses less wiring and can handle multiple masters, but requires unique addresses for each device; SPI is simpler in protocol structure but requires more hardware lines for multiple slaves.

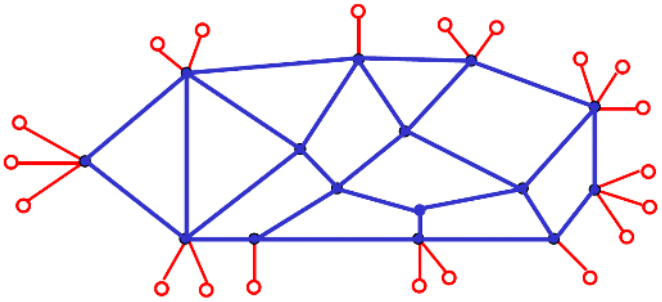
Both I2C and SPI are widely used in electronic systems, each fitting different needs based on speed, complexity, and resource availability.

BASIC NETWORK CONCEPTS

A computer network is a set of nodes connected by communication links.
Nodes may be:

End systems, on which applications can run (COMPUTER)

Communication nodes, which just pass data (ROUTER)



TYPES OF COMPUTER NETWORK

Computer networks are often classified into:

Local Area Networks (LAN): Size up to a few kilometres, typically covering a building, company or institution.

Wide Area Networks (WAN): Large geographical coverage, perhaps worldwide.

Metropolitan Area Networks (MAN): Covering a town or other relatively large area.

INTERNET LAYERED ARCHITECTURE

A simplified model, with OSI Upper Layers reduced to a single layer:

Application	Direct support to application processes
Transport	End-to-end transfer of data
Network	Transfer of data between arbitrary systems
Data Link	Transfer of data between directly connected systems
Physical	Signalling on physical medium

SERVICES

Service describes what facilities are offered by a layer viewed as a "**black box**", for example: Sequence preservation

Service does not tell us how these features are achieved.

PROTOCOLS

Specify rules for how to provide the desired service:

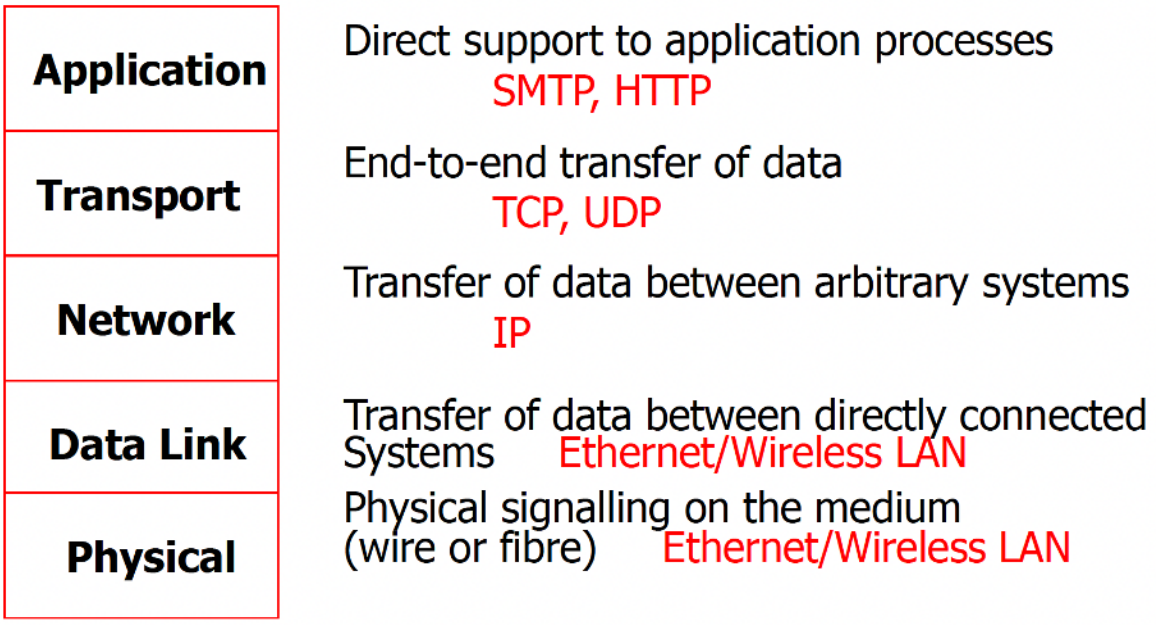
Rules of procedure: These are predefined methods in a network protocol that determine the sequence of messages to be exchanged in response to different events or conditions. This ensures reliable, predictable, and efficient communication between devices within a network.

Message formats: Format and encoding of messages to be transferred between the parties involved.

OSI notation:

Service Data Unit (SDU): A message supplied by a user of a service.

Protocol Data Unit (PDU): A message exchanged between two or more parties as part of a protocol (SDU + Protocol Control Information).



INTERNET PROTOCOL (IP)

IP is a fundamental network protocol used for transmitting packets between devices across different networks. It operates without requiring a continuous connection, making it efficient for sending data over the internet, it also supports simultaneous two-way data transmission. It supports both point-to-point and multicast data transmission, enabling communication between specific pairs of devices or groups of devices.

Versions:

- **IPv4**: Uses 32-bit addresses, allowing for about 4.3 billion unique addresses (e.g., 130.225.76.44).
- **IPv6**: Introduced to expand the available address space with 128-bit addresses, accommodating a vast increase in devices online (e.g., ff:aec1:0:0:0:ffff:fffe:1).

CLIENT/SERVER SYSTEMS

This is a design paradigm where some computers, called servers, provide services to other computers, called clients. Clients make requests to servers which then process these requests and return the responses. This model underpins most internet applications, providing a straightforward way to structure and manage network communications. This model allows efficient, scalable, and manageable networked interactions, making it ideal for everything from small local networks to the global internet.

TRANSMISSION CONTROL PROTOCOL(TCP)

TCP is connection-oriented, meaning it requires a connection to be established between the communicating devices before data can be sent. It ensures reliable data transmission by using acknowledgements, sequence numbers, and retransmissions if necessary.

Benefits: Provides error checking and guarantees that data will be delivered in the order it was sent. Ideal for applications where accuracy is more critical than speed.

Usage: Used for applications like web browsing, email, and file transfers where reliability is crucial

USER DATAGRAM PROTOCOL (UDP)

UDP is connectionless, which means data is sent without establishing a prior connection, making it faster but less reliable than TCP. It does not guarantee message order or retransmit lost packets.

Benefits: Because it has lower overhead, it is faster and more efficient for applications that can tolerate some loss of data.

Usage: Commonly used for live broadcasts, video streaming, and online games where speed is more important than perfect delivery.

In summary, TCP is used when reliability is key, while UDP is preferred for faster communication where occasional data loss is acceptable.

SMTP (SIMPLE MAIL TRANSFER PROTOCOL)

SMTP is primarily used for sending emails between servers and across networks. It handles the transfer of email messages from a sender's email server to the recipient's email server.

Operation: When you send an email, SMTP is responsible for connecting to the recipient's mail server and transferring the email to it. Once the email reaches the recipient's mail server, other protocols are used to retrieve the email.

Characteristics: SMTP only facilitates the sending of emails and does not retrieve them. It ensures that the message is correctly formatted and transmitted to the receiving mail server.

HTTP (HYPERTEXT TRANSFER PROTOCOL)

HTTP is used to load web pages using hyperlinks. It is the protocol used by the World Wide Web that allows web browsers and servers to communicate.

Operation: HTTP functions as a request-response protocol in a client-server computing model. A web browser (client) sends an HTTP request to the server; then the server returns the response to the client. The response contains status information about the request and may also contain the requested content such as HTML files, images, and other data.

Characteristics: HTTP is stateless, meaning that each request from a client to a server is treated as new, with no memory of previous interactions. It can be used to transmit virtually any kind of data.

SMTP is specifically designed for email transmission, ensuring that emails are sent to the correct server. In contrast, HTTP is more versatile, used primarily for retrieving HTML pages and associated content which makes up web pages. Both play crucial roles in daily internet communications and browsing experiences.

ETHERNET

Ethernet is a technology for connecting devices in a wired network. It uses cables to link devices such as computers, routers, and switches.

Ethernet transmits data over cables using electrical signals. Each device on an Ethernet network connects to a central hub, switch, or router via an Ethernet cable, which facilitates communication between the devices.

Key Features:

Known for their reliability and speed, Ethernet connections are typically faster and more secure than wireless connections because they are less susceptible to interference.

WIRELESS LAN

Wireless LAN is a type of local area network that uses high-frequency radio waves rather than wires to communicate between devices.

WLAN allows devices to connect to a network using Wi-Fi. A wireless router serves as the central communication hub, broadcasting and receiving signals to and from connected devices within its range.

Key Features: The main advantage of WLAN is the convenience of mobility; devices can connect to the network without physical cables, providing flexibility and ease of use. However, WLANs can be more susceptible to interference from other radio signals and generally offer lower speeds than wired connections.

In summary, Ethernet provides a stable, fast, and secure connection for networked devices via cables, making it ideal for environments where high performance is critical. In contrast, WLAN offers flexibility and mobility, ideal for environments where users need to move around, such as in homes or businesses. Both are essential for creating efficient and effective local area networks.