➔NETWORK SECURITY

Network security is a term that describes the security tools, tactics and security policies designed to monitor, prevent and respond to unauthorized network intrusion, while also protecting digital assets, including network traffic. Network security includes hardware and software technologies (including resources such as savvy security analysts, hunters, and incident responders) and is designed to respond to the full range of potential threats targeting your network. Network security is the defence you use to protect yourself against ever- increasing cybercrime.

The Three Key Focuses of Network Security

There are three key focuses that should serve as a foundation of any network security strategy: protection, detection and response.

- **Protection** entails any tools or policies designed to prevent network security intrusion.
- ***Detection*** refers to the resources that allow you to analyze network traffic and quickly identify problems before they can do harm.
- **R*esponse*** is the ability to react to identified network security threats and resolve them as quickly as possible.

Benefits of Network Security

Network security tools and devices enable organizations to protect its sensitive information, overall performance, reputation and its continuity in business. Secure and reliable networks protect not just organizational interests and operations, but also any client or customer who exchanges information with the organization, in addition to the general public. The benefits of network security are:

- **Builds trust:** Security for large systems translates to security for everyone. Network security boosts client and consumer confidence, and it protects your business from the reputational and legal fallout of a security breach.
- **Mitigates risk:** The right network security solution will help your business stay compliant with business and government regulations, and it will minimize the business and financial impact of a breach if it does occur.
- **Protects proprietary information:** Your clients and customers rely on you to protect their sensitive information. Your business relies on that same protection, too. Network security ensures the protection of information and data shared across the network.
- **Enables a more modern workplace:** From allowing employees to work securely from any location using VPN to encouraging collaboration with secure network access, network security provides options to enable the future of work. Effective network security also provides many levels of security to scale with your growing business.

NETWORK SECURITY TOOLS AND TECHNIQUES

Enterprises' network encounters varying degrees of threats, and therefore should be prepared to defend, identify and respond to a full range of attacks. However, the reality is that the biggest danger to most companies is not fly- by-night threat actors, but the attackers that are well-funded and are targeting specific organizations for specific reasons. Hence, network security strategy needs to be able to address the various methods these actors might employ. Here are 14 different network security tools and techniques designed to help you do just that:

1. **Access control:** If threat actors cannot access your network, the amount of damage they will be able to do will be extremely limited. But in addition to preventing unauthorized access, be aware that even *authorized* users can be potential threats. Access control allows you to increase your network security by limiting user access and resources to only the parts of the network that directly apply to individual users' responsibilities.
2. **Anti-malware software:** Malware, in the form of viruses, Trojans, worms, keyloggers, spyware, etc. are designed to spread through computer systems and infect networks. Anti-malware tools are a kind of network security software designed to identify dangerous programs and prevent them from spreading. Anti-malware and antivirus software may also be able to help resolve malware infections, minimizing the damage to the network.
3. **Anomaly detection:** It can be difficult to identify anomalies in your network without a baseline understanding of how that network *should* be operating. Network Anomaly Detection Engines (ADE) allows you to analyze your network, so that when breaches occur, you will be alerted to them quickly enough to be able to respond.
4. **Application security:** For many attackers, applications are a defensive vulnerability that can be exploited. Application security helps establish security parameters for any applications that may be relevant to your network security.
5. **Data Loss Prevention (DLP):** Often, the weakest link in network security is the human element. DLP technologies and policies help protect staff and other users from misusing and possibly compromising sensitive data or allowing said data out of the network.
6. **Email security:** As with DLP, email security is focused on shoring up human-related security weaknesses. Via phishing strategies (which are often very complex and convincing), attackers persuade email recipients to share sensitive information via desktop or mobile device, or inadvertently download malware into the targeted network. Email security helps identify dangerous emails and can also be used to block attacks and prevent the sharing of vital data.
7. **Endpoint security:** The business world is becoming increasingly, "*bring your own device"* (BYOD), to the point where the distinction between personal and business computer devices is almost non-existent. Unfortunately, sometimes the personal devices become targets when users rely on them to access business networks. Endpoint security adds a layer of defence between remote devices and business networks.
8. **Firewalls:** Firewalls function much like gates that can be used to secure the borders between your network and the internet. Firewalls are used to manage network traffic, allowing authorized traffic through while blocking access to non-authorized traffic.
9. **Intrusion prevention systems:** Intrusion prevention systems (also called intrusion detection) constantly scan and analyze network traffic/packets, so that different types of attacks can be identified and responded to quickly. These systems often keep a database of known attack methods, so as to be able to recognize threats immediately.
10. **Network segmentation:** There are many kinds of network traffic, each associated with different security risks. Network segmentation allows you to grant the right access to the right traffic, while restricting traffic from suspicious sources.
11. **Security information and event management (SIEM):** Sometimes simply pulling together the right information from so many different tools and resources can be prohibitively difficult particularly when time is an issue. SIEM tools and software give responders the data they need to act quickly.

12. **Virtual private network (VPN):** VPN tools are used to authenticate communication between secure networks and an endpoint device. Remote-access VPNs generally use IPsec or Secure Sockets Layer (SSL) for authentication, creating an encrypted line to block other parties from eavesdropping.
13. **Web security:** Including tools, hardware, policies and more, web security is a blanket term to describe the network security measures businesses take to ensure safe web use when connected to an internal network. This helps prevent web-based threats from using browsers as access points to get into the network.
14. **Wireless security:** Generally speaking, wireless networks are less secure than traditional networks. Thus, strict wireless security measures are necessary to ensure that threat actors are not gaining access.

➔ CLIENT-SERVER COMPUTING

There are two configurations of networks: Client-Server and Peer-to-Peer networks. In client server, the client requests resources while the server serves same. In Peer-to-peer configuration, each node is free to communicate with others or not. The nodes under this configuration are not over-seen by any node or the other, they relate in a workgroup.

Client Server Computing

In client-server computing, the client requests a resource and the server provides that resource. A server may serve multiple clients at the same time while a client is in contact with only one server. Both the client and server usually communicate via a computer network, as pictured in figure 1.4.1, but sometimes they may reside in the same system.

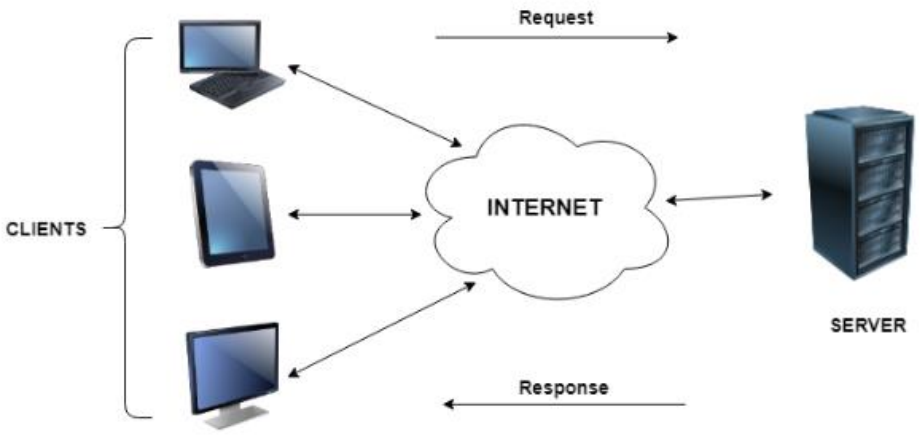


Figure: Client-Server Computing

Characteristics of Client Server Computing

The salient points for client server computing are as follows:

- The client server computing works with a system of request and response. The client sends a request to the server and the server responds with the desired information.
- The client and server should follow a common communication protocol so they can easily interact with each other. All the communication protocols are available at the application layer.

- A server can only accommodate a limited number of client requests at a time. So it uses a system based to priority to respond to the requests.
- Denial of Service (DoS) attacks hinders servers' ability to respond to authentic client requests by inundating it with false requests.

An example of a client server computing system is a web server. It returns the web pages to the clients that requested them.

Differences between Client-Server and Peer-to-Peer Computing

The major differences between client-server computing and peer-to-peer computing are as follows:

- In client server computing, a server is a central node that services many client nodes. On the other hand, in a peer-to-peer system, the nodes collectively use their resources and communicate with each other.
- In client server computing, the server is the one that communicates with the other nodes. In peer-to-peer computing, all the nodes are equal and share data with each other directly.
- Client-Server computing is believed to be a sub-category of the peer-to-peer computing.

Advantages of Client- Server Computing

- All the required data is concentrated in a single place i.e. the server. So it is easy to protect the data and provide authorisation and authentication.
- The server need not be located physically close to the clients yet, the data can be accessed efficiently.
- It is easy to replace, upgrade or relocate the nodes in the client-server model because all the nodes are independent and request data only from the server.
- All the nodes i.e. clients and server may not be built on similar platforms yet, they can easily facilitate the transfer of data.

Disadvantages of Client Server Computing

- If all the clients simultaneously request data from the server, it may get overloaded. This may lead to congestion in the network
- If the server fails for any reason, then none of the requests of the clients can be fulfilled. This leads to failure of the client-server network.
- The cost of setting and maintaining a client-server model are quite high.

➔PARALLEL SYSTEMS

Parallel computing is a type of computation in which many calculations or processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time. There are several different forms of parallel computing: bitlevel, instruction-level, data, and task parallelism. Parallelism has long been employed in high-performance computing, but has gained broader interest due to the physical constraints preventing frequency scaling. As power consumption (and consequently heat generation) by computers has become a concern in recent years, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multicore processors.

Parallel Processing Systems are designed to speed up the execution of programs by dividing the program into multiple fragments and processing these fragments simultaneously. Such systems are

multiprocessor systems also known as tightly coupled systems. Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster or a combination of both. Parallel computing is an evolution of serial computing where the jobs are broken into discrete parts that can be executed concurrently.

Each part is further broken down to a series of instructions. Instructions from each part execute simultaneously on different CPUs. Parallel systems are more difficult to program than computers with a single processor because the architecture of parallel computers varies accordingly and the processes of multiple CPUs must be coordinated and synchronized. Several models for connecting processors and memory modules exist, and each topology requires a different programming model. The three models that are most commonly used in building parallel computers include synchronous processors each with its own memory, asynchronous processors each with its own memory and asynchronous processors with a common, shared memory.

Flynn's Classification of Parallel Systems

Flynn has classified the computer systems based on parallelism in the instructions and in the data streams. These are:

1. **Single Instruction, Single Data stream (SISD):** An SISD computing system is a uniprocessor machine capable of executing a single instruction, which operates on a single data stream (see Figure 1 below). In SISD, machine instructions are processed sequentially; hence computers adopting this model are popularly called *sequential computers*. Most conventional computers are built using the SISD model. All the instructions and data to be processed have to be stored in primary memory. The speed of the processing element in the SISD model is limited by the rate at which the computer can transfer information internally. Dominant representative SISD systems are IBM PC, Macintosh, and workstations.

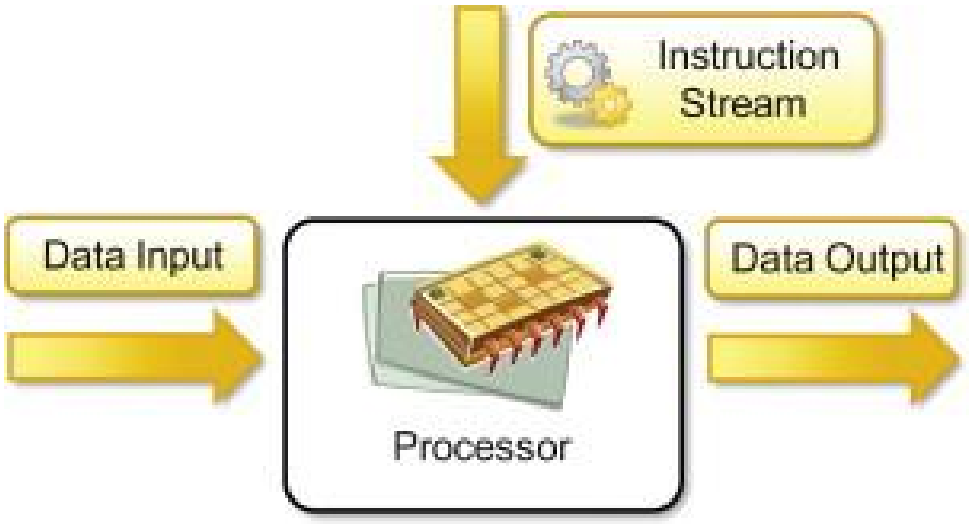


Figure1: Single Instruction, Single Data stream (SISD)

2. **Single Instruction, Multiple Data stream (SIMD):** SIMD represents single-instruction multiple-data streams. The SIMD model of parallel computing includes two parts such as a front-end computer of the usual von Neumann style, and a processor array as displayed in figure 2.

The processor array is a collection of identical synchronized processing elements adequate for simultaneously implementing the same operation on various data. Each processor in the array has a small amount of local memory where the distributed data resides while it is being processed in parallel. The processor array is linked to the memory bus of the front end so that the front end can randomly create the local processor memories as if it were another memory.

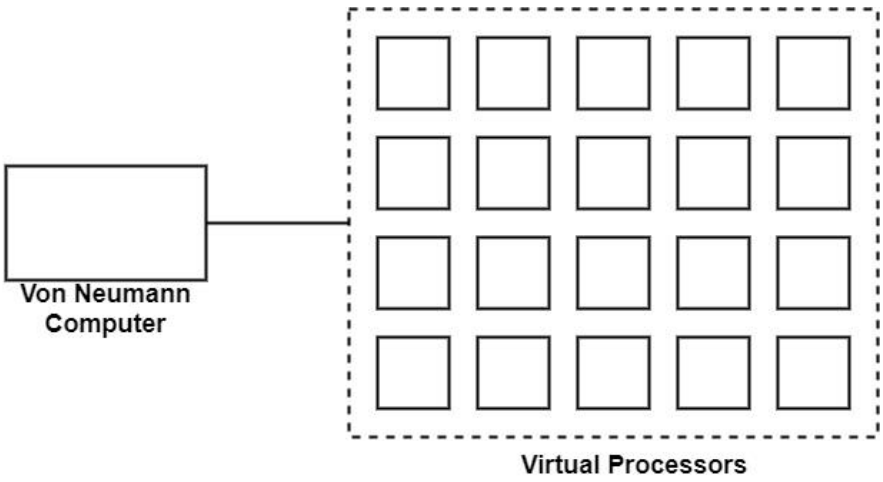


Figure 2: Single instruction stream, multiple data stream (SIMD)

A program can be developed and performed on the front end using a traditional serial programming language. The application program is performed by the front end in the usual serial method, but problem command to the processor array to carry out SIMD operations in parallel. The similarity between serial and data-parallel programming is one of the valid points of data parallelism. Synchronization is created irrelevant by the lock-step synchronization of the processors. Processors either do nothing or similar operations at the same time.

In SIMD architecture, parallelism is exploited by using simultaneous operations across huge sets of data. This paradigm is most beneficial for solving issues that have several data that require to be upgraded on a wholesale basis. It is dynamically powerful in many regular scientific calculations.

Two main configurations have been applied in SIMD machines. In the first scheme, each processor has its local memory. Processors can interact with each other through the interconnection network. If the interconnection network does not support a direct connection between given groups of processors, then this group can exchange information via an intermediate processor.

In the second SIMD scheme, processors and memory modules communicate with each other via the interconnection network. Two processors can send information between each other via intermediate memory module(s) or possibly via intermediate processor(s). The BSP (Burroughs' Scientific Processor) used the second SIMD scheme.

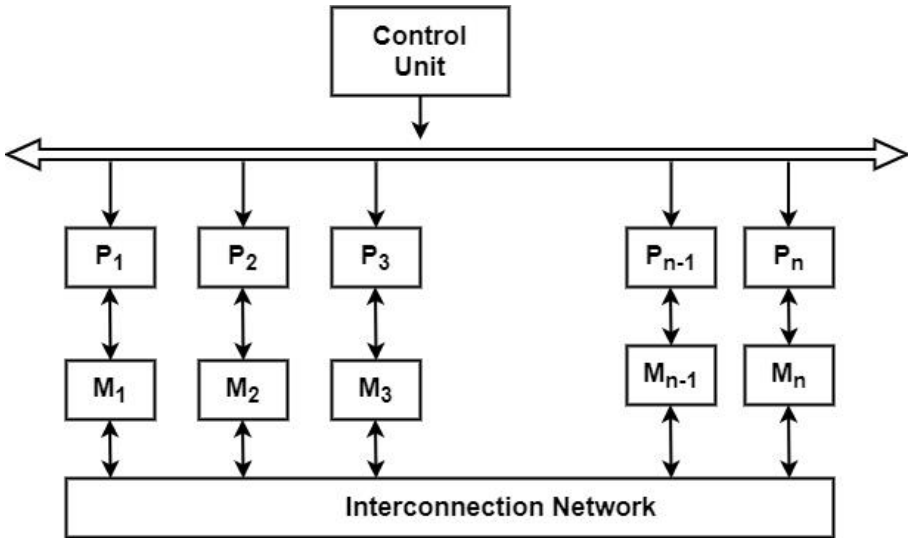


Figure 2a: Single instruction stream, multiple data stream (SIMD) Scheme-1

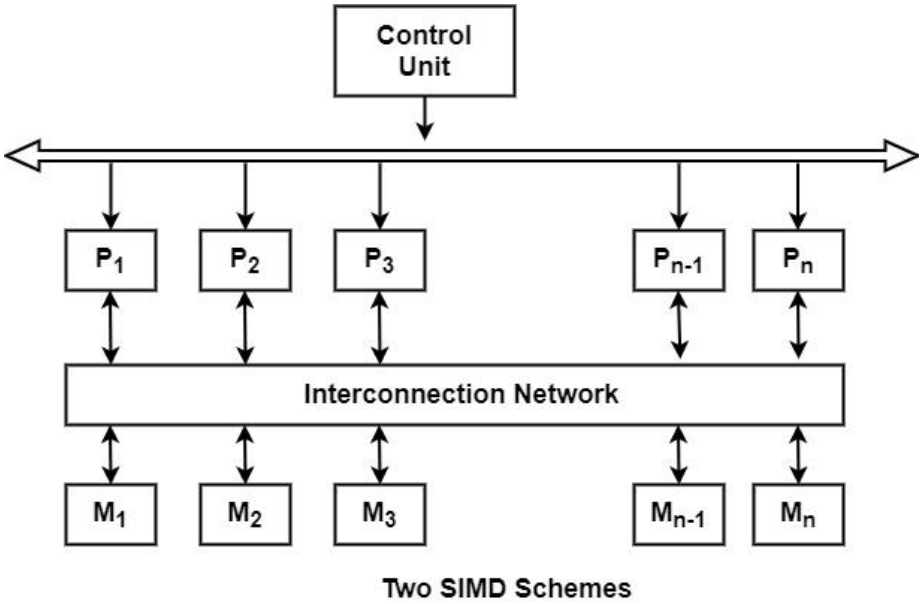


Figure 2b: Single Instruction, Multiple Data stream (SIMD) Scheme-2

3. **Multiple Instruction Single Data stream (MISD):** In this association, multiple processing elements are structured under the control of multiple control units. Each control unit is handling single instruction stream and processed through its corresponding processing element. But single processing element is processing only a one data stream at a time. Hence, for handling single data stream and multiple instruction streams, multiple processing elements and multiple control units are organised in this classification. All processing elements are relate with the common shared memory for the organisation of one data stream as given in Figure 3. The only identified instance of a computer capable of MISD operation is the C.mmp built by Carnegie-Mellon University.
   This type of computer organisation is denoted as:
   $I_s > 1$
   $D_s = 1$

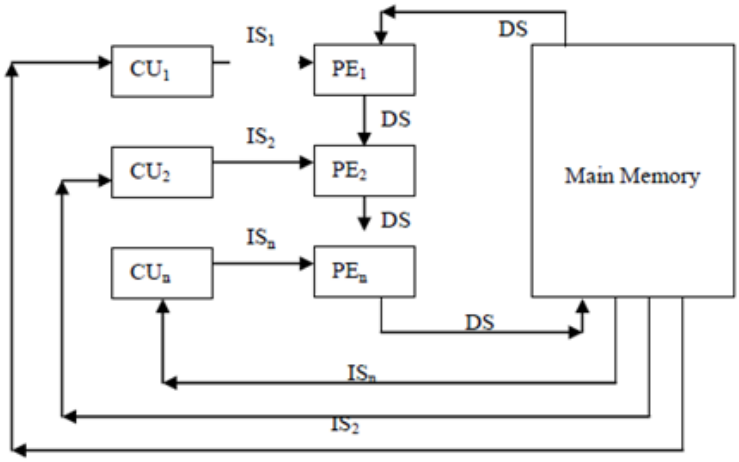
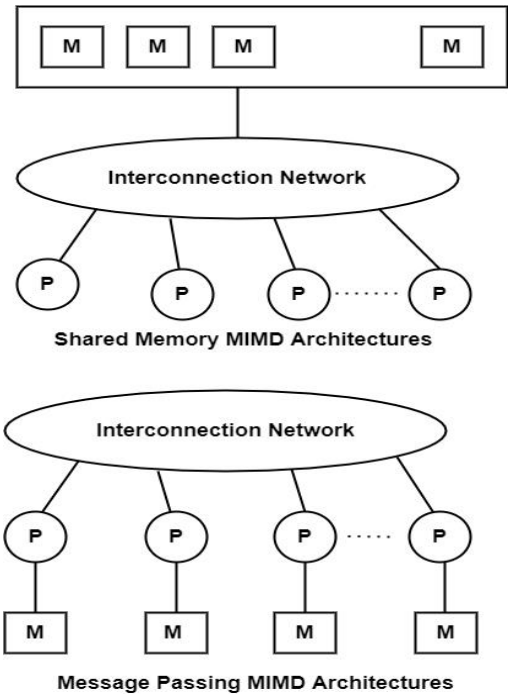


Figure 3: Multiple-Instruction Single-Data streams (MISD)

This classification is not popular in commercial machines as the thought of single data streams implementing on multiple processors is rarely functional. But for the particular applications, MISD organisation can be very useful. For example, Real time computers need to be fault tolerant where several processors implement the same data for producing the redundant data. This is also called as N-version programming. All these redundant data are measured to as results which should be similar; otherwise faulty unit is returned. Thus MISD machines can be useful to fault tolerant real time computers.

4. **Multiple Instruction, Multiple Data stream (MIMD):** MIMD stands for Multiple-instruction multiple-data streams. It includes parallel architectures are made of multiple processors and multiple memory modules linked via some interconnection network. They fall into two broad types including shared memory or message passing. A shared memory system generally accomplishes interprocessor coordination through a global memory shared by all processors. These are frequently server systems that communicate through a bus and cache memory controller. The bus/ cache architecture alleviates the need for expensive multi-ported memories and interface circuitry as well as the need to adopt a message- passing paradigm when developing application software. Because access to shared memory is balanced, these systems are also called SMP (symmetric multiprocessor) systems. Each processor has an equal opportunity to read/write to memory, including equal access speed.

Figure 4: Multiple-Instruction Multiple-Data streams (MIMD)



The above classification of parallel computing system is focused in terms of two independent factors: the number of data streams that can be simultaneously processed, and the number of instruction streams that can be simultaneously processed. Here, by 'instruction stream' we mean an algorithm that instructs the computer what to do whereas 'data stream' (i.e. input to an algorithm) means the data that are being operated upon.

Relevance of Flynn's Classification to Parallel Systems

Even though Flynn has classified the computer 'systems into four types based on parallelism but only two of them are relevant to parallel computers. These are SIMD and MIMD computers.

SIMD computers are consisting of 'n' processing units receiving a single stream of instruction from a central control unit and each processing unit operates on a different piece of data. Most SIMD computers operate synchronously using a single global clock. The block diagram of SIMD computer is shown below:

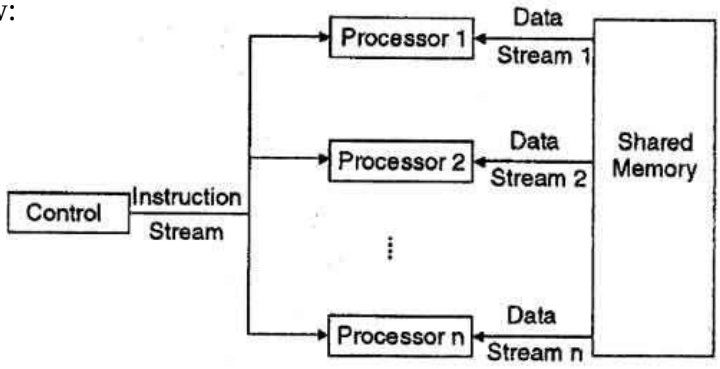


Figure 5: Single Instruction, Multiple Data stream (SIMD) Block Diagram

MIMD computers are consisting of 'n' processing units; each with its own stream of instruction and each processing unit operate on unit operates on a different piece of data. MIMD is the most powerful computer system that covers the range of multiprocessor systems. The block diagram of MIMD computer is shown.

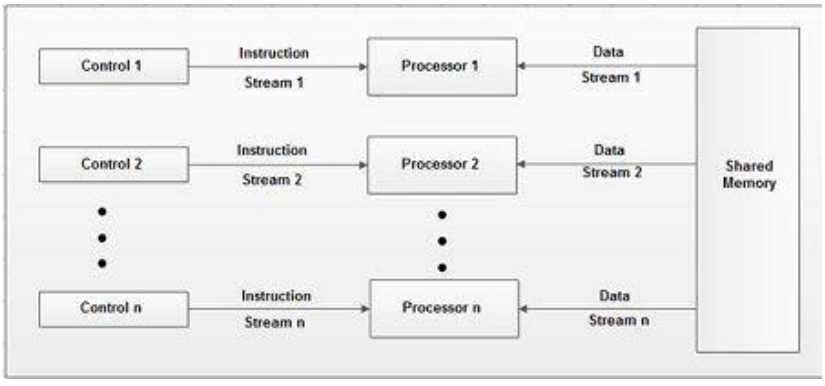


Figure 6: Multiple Instruction, Multiple Data stream (MIMD) Block Diagram

The SIMD systems are easier to program because it deals with single thread of execution. On the hand, the MIMD machines are more efficient because you can utilize the full machine power.

Parallel Computers and Applications

Parallel operating systems are primarily concerned with managing the resources of parallel machines. A parallel computer is a set of processors that are able to work cooperatively to solve a computational problem. So, a parallel computer may be a supercomputer with hundreds or thousands of processors or

may be a network of workstations.

A few years ago, parallel computers could be found only in research laboratories and they were used mainly for computation intensive applications like numerical simulations of complex systems. Today, there are a lot of parallel computers available in the market; used to execute both data intensive applications in commerce and computation intensive applications in science and engineering.

Today, new applications arise and demand faster computers. Commercial applications are the most used on parallel computers. A computer that runs such an application should be able to process large amount of data in sophisticated ways. These applications include graphics, virtual reality, and decision support, parallel databases, medicine diagnosis and so on. We can say with no doubt that commercial applications will define future parallel computers architecture but scientific applications will remain important users of parallel computing technology.

Concurrency becomes a fundamental requirement for algorithms and programs. A program has to be able to use a variable number of processors and also has to be able to run on multiple processors computer architecture. According to Tanenbaum, a distributed system is a set of independent computers that appear to the user like a single one. So, the computers have to be independent and the software has to hide individual computers to the users. MIMD computers and workstations connected through LAN and WAN are examples of distributed systems. The main difference between parallel systems and distributed systems is the way in which these systems are used. A parallel system uses a set of processing units to solve a single problem A distributed system is used by many users together.

➔PARALLEL PROGRAMMING MODELS

In computing, a **parallel programming model** is an abstraction of parallel computer architecture, with which it is convenient to express algorithms and their composition in programs. The value of a programming model can be judged on its generality: how well a range of different problems can be expressed for a variety of different architectures, and its performance: how efficiently the compiled programs can execute. The implementation of a parallel programming model can take the form of a library invoked from a sequential language, as an extension to an existing language, or as an entirely new language. Consensus around a particular programming model is important because it leads to different parallel computers being built with support for the model, thereby facilitating portability of software. In this sense, programming models are referred to as bridging between hardware and software.

A parallel programming model is a set of program abstractions for fitting parallel activities from the application to the underlying parallel hardware. It spans over different layers: applications, programming languages, compilers, libraries, network communication, and I/O systems. Two widely known parallel programming models are:

a. *shared memory* and
b. *message passing*

There are also different:

c. Combinations of both.

a. **In the shared-memory programming model**, tasks share a common address space, which they read and write in an asynchronous manner. The communication between tasks is implicit. If more than one task accesses the same variable, the semaphores or locks can be used for

synchronization. By keeping data local to the processor and making private copies, expensive memory accesses are avoided, but some mechanism of coherence maintenance is needed when multiple processors share the same data with the possibility of writing.

b. **In the message-passing programming model,** tasks have private memories, and they communicate explicitly via message exchange. To exchange a message, each sends operation needs to have a corresponding receive operation. Tasks are not constrained to exist on the same physical machine.
c. **A suitable combination of two previous models** is sometimes appropriate. Processors can directly access memory on another processor. This is achieved via message passing, but what the programmer actually sees is shared-memory model.

Mainstream parallel programming environments are based on augmenting traditional sequential programming languages with low-level parallel constructs (library function calls and/or compiler directives).

The Programming Models

1. Message Passing Interface (MPI): The MPI is a library of routines with the bindings in Fortran, C, and C++ and it is an example of an explicitly parallel API that implements the message-passing model via library function calls. The set of processes with separate address spaces coordinate the computation by explicitly sending and receiving messages. Each process has a separate address space, its own program counter, and its own call stack. However, high- level constructs such as synchronization, communication, and mapping data to processes are left to a programmer to implement. MPI supports point-to-point communication between any two processes. It also enables the collective communication operations where a group of processes perform global/collective operations, such as gather, scatter, reduce, and scan. In an heterogeneous environment, in order to optimize the performance, an MPI implementation may map processes to processors in a particular way. Similarly, an MPI implementation may optimize the way processes communicate during a global operation. For example, in case of MPI_Reduce, the communicating nodes do not have to form a tree structure, if an alternative structure brings better performance for the underlying parallel machine.
2. OPenMP (Open Multi-Processing): On the other side, OpenMP is an example of mainly implicit parallel API intended for shared-memory multiprocessors. It exploits parallelism through compiler directives and the library function calls. Unlike MPI, where all threads are spawned at the beginning of the execution and are active until the program terminates, in OpenMP, a single master thread starts execution, and additional threads are active only during the execution of a parallel region. To reduce the overheads, these threads are spawned when the program enters a parallel region for the first time, and they are blocked while the program is executing a nonparallel region.
3. MapReduce**:** One of the most widely used parallel programming models today is MapReduce. MapReduce is easy both to learn and use, and is especially useful in analyzing large datasets. While it is not suitable for several classes of scientific computing operations that are better served by message-passing interface or OpenMP, such as numerical linear algebra or finite element and finite difference computations, MapReduce's utility in workflows frequently called “big data” has made it a mainstay in high performance computing.

MapReduce programming model and the Hadoop open-source framework supports it.

4. ***OpenCL*** (Open Computing Language): OpenCL has some advantages over other parallel programming models. First of all, it is the only one of the "open" standards for which there, actually, are implementations by all major vendors—unlike for OpenMP or OpenACC. The level of vendor support, however, is a different story. OpenCL is a library that can be used with any C/C++ compiler, which makes it independent of additional tools. The kernels are written separately in a C-like language and compiled at runtime for the present hardware. The kernel compiler comes with the OpenCL Implementation provided by the hardware vendor. A kernel written in OpenCL will run everywhere, including conventional CPUs, Intel Xeon Phi coprocessors, GPGPUs, some FPGAs, and even mobile devices. OpenCL programs are divided into host and kernel code. Only the latter is executed on the compute device. In the host program, kernels and memory movements are queued into command queues associated with a device. The kernel language provides features like vector types and additional memory qualifiers. A computation must be mapped to work- groups of work-items that can be executed in parallel on the compute units (CUs) and processing elements (PEs) of a compute device. A work-item is a single instance of a kernel function. For each kernel-call, an NDRange (*n*-dimensional range) specifies the dimension, number, and shape of the work-groups. Global synchronization during the execution of a kernel is unavailable. Work-items inside a work-group can be synchronized. OpenCL provides a complex memory model with a relaxed consistency.
5. **The CUDA (Compute Unified Device Architecture) programming model:** The CUDA programming model is a parallel programming model that provides an abstract view of how processes can be run on underlying GPU architectures. The evolution of GPU architecture and the CUDA programming language have been quite parallel and interdependent. Although the CUDA programming model has stabilized over time, the architecture is still evolving in its capabilities and functionality. GPU architecture has also grown in terms of the number of transistors and number of computing units over years, while still supporting the CUDA programming model.

➔MESSAGE PASSING PROGRAMMING

In computer science, **message passing** is a technique for invoking behavior (i.e., running a program) on a computer. The invoking program sends a message to a process (which may be an actor or object) and relies on that process and its supporting infrastructure to then select and run some appropriate code. Message passing differs from conventional programming where a process, subroutine, or function is directly invoked by name. Message passing is key to some models of concurrency and object-oriented programming.

Message passing is ubiquitous in modern computer software. It is used as a way for the objects that make up a program to work with each other and as a means for objects and systems running on different computers (e.g., the Internet) to interact. Message passing may be implemented by various mechanisms, including channels.

The message-passing

A message transfer is when data moves from variables in one sub- program to variables in another sub-program. The message consists of the data being sent. The message passing system has no

interest in the value of this data. It is only concerned with moving it. In general the following information has to be provided to the message passing system to specify the message transfer.

- Which processor is sending the message?
- Where is the data on the sending processor?
- What kind of data is being sent?
- How much data is there?
- Which processor(s) are receiving the message?
- Where should the data be left on the receiving processor?
- How much data is the receiving processor prepared to accept?

In general, the sending and receiving processors will cooperate in providing this information. Some of this information provided by the sending processor will be attached to the message as it travels through the system and the message passing system may make some of this information available to the receiving processor.

As well as delivering data, the message passing system has to provide some information about progress of communications. A receiving processor will be unable to use incoming data if it is unaware of its arrival. Similarly, a sending processor may wish to find out if its message has been delivered. A message transfer therefore provides synchronisation information in addition to the data in the message.

The essence of message passing is communication and many of the important concepts can be understood by analogy with the methods that people use to communicate, phone, fax, letter, radio etc. Just as phones and radio provide different kinds of service different message passing systems can also take very different approaches. For the time being we are only interested in general concepts rather than the details of particular implementations.

The message-passing programming model

The sequential paradigm for programming is a familiar one. The programmer has a simplified view of the target machine as a single processor which can access a certain amount of memory. He or she therefore writes a single program to run on that processor. The paradigm may, in fact, be implemented in various ways, perhaps in a time-sharing environment where other processes share the processor and memory, but the programmer wants to remain above such implementation-dependent details, in the sense that the program or the underlying algorithm could in principle be ported to any sequential architecture -- that is after all the point of a paradigm.

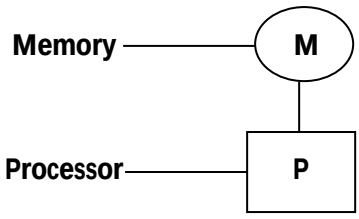


Figure: The sequential programming paradigm

The message-passing paradigm is a development of this idea for the purposes of parallel programming. Several instances of the sequential paradigm are considered together. That is, the programmer imagines several processors, each with its own memory space, and writes a program to run on each processor. So far, so good, Parallel programming by definition requires co-operation between the processors to solve a task, which requires some means of communication. The main point

of the message-passing paradigm is that the processes communicate by sending each other messages. Thus the message-passing model has no concept of a shared memory space or of processors accessing each other's memory directly -- anything other than message-passing is out with the scope of the paradigm. As far as the programs running on the individual processors are concerned, the message passing operations are just subroutine calls. Those with experience of using networks of workstations, client-server systems or even object-oriented programs will recognise the message- passing paradigm as nothing novel.

Single Program Multiple Data Streams (SPMD)

SPMD mode is a method of parallel computing, its processors run the same program, but execute different data. SPMD could get better computing performance through increasing the number of processors. This also increased power consumption, and had problems of heat dissipation at high clock speeds. Previously, computing performance was increased through clock speed scaling. Parallel computing allow more instructions to complete in a given time through parallel execution. Nowadays, parallel computing has entered main stream use, following the introduction of multi-core processors.

Advantages of SPMD

1. Locality: Data locality is essential to achieving good performance on large-scale machines, where communication across the network is very expensive.
2. Structured Parallelism: The set of threads is fixed throughout computation. It is easier for compilers to reason about SPMD code, resulting in more efficient program analyses than in other models.
3. Simple runtime implementation: SPMD belongs to MIMD, it has a local view of execution and parallelism is exposed directly to the user, compilers and runtime systems require less effort to implement than many other MIMD models.

Disadvantages of SPMD

1. SPMD is a flat model, which makes it difficult to write hierarchical code, such as divide-and-conquer algorithms, as well as programs optimized for hierarchical machines.
2. The second disadvantage may be that it seems hard to get the desired speedup using SPMD.

The advantages of SPMD are very obvious, and SPMD is still a common use on many large-scale machines. Many scientists have done researches to improve SPMD, such as the recursive SPMD, which provides hierarchical teams. So, SPMD will still be a good method for parallel computing in the future.

➔DEPENDENCE ANALYSIS

Dependencies are the relationships that exist between the constituent parts, or entities, of a complete system. A single dependency represents a directional relationship, and therefore a sequence, between a pair of system entities.

For example, if the system were a journey, there might be a walking dependency from a flight to a taxi. If this is applied to a software context, in a codebase, a function may depend on (invoke) another

function. As you can infer, dependencies can have types, or behaviour, as well as a direction, indicating how the transition occurs and in what order. Dependency analysis is the process of extracting the set of entities, their dependencies and their types and direction, from the system so that the system structure can be analysed, understood and improved.

Dependency analysis

When examining an artifact for re-use you might want to understand what it depends on. Developing a service that has a dependency on a large number of other distinct systems is likely to result in something that has to be revalidated every time each of those dependencies changes (which might therefore be quite often). To undertake a typical dependency analysis, perform the following **steps**:

1. Identify the artefact with dependencies you want to analyze.
2. Trace through any relationships defined on that artefact and identify the targets of the relationships. This impact analysis thus results in a list of "dependencies" that the selected artefact depends on.
3. If these "dependencies" also depend on other artefacts, then the selected artefact will also have an indirect dependency. The impact analysis must therefore act recursively looking for relationships from any of the "dependencies".

This process continues until a complete graph is obtained starting at the selected artefact and finishing with artefacts that have no further dependencies. The selected artefact might have a dependency on any artefact in the graph. Kindly note that an object can exist multiple times in the graph if it can be reached in different ways.

How dependencies are found

When impact analysis is started, it does not change the direction of processing through the graph. For example, an object, B, has a dependency on object C, and object B is depended on by objects A and D as shown in Figure below.

If object A is selected for analysis, the results list includes object B and object C. Despite object D also having a dependency on objects B and C the analysis keeps tracing down through the dependencies and will not find any objects which are backwards in the dependency hierarchy that are not directly linked to the selected object, so object D will not be in the results list.

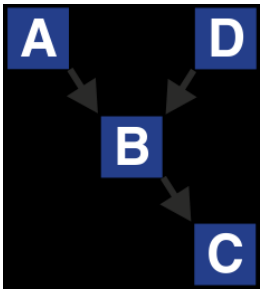


Figure: Dependency analysis graph

Why use dependency analysis

A codebase may have been initially designed with a modular or layered architecture, but over time, that clean structure could have been eroded, introducing unexpected coupling between elements, which can damage or even destroy the architectural intent. This can lead to ongoing development or maintenance of the codebase becoming more complex and time consuming to do, which can result in additional errors and complications creeping into the code. The longer these problems go unchecked,

the more the problems can be compounded into the codebase and therefore more expensive in time and money to address. Beyond dependency analysis tools providing a way for you to address such problems, they may also offer the ability to enforce structural rules dictating how the different modules/layers or external entities such as third party libraries may interact with and within the system respectively. Once in place these rules typically allow the codebase to be automatically checked for adherence and where violations are detected, notifications of said problems can be immediately brought to the attention of team members so the problems can be more quickly identified and addressed before the issues become too embedded.

Improve Refactoring

Software architecture evaluation and refactoring should be a standard activity in any development process because it is a way to reduce risk. It is relatively inexpensive, and it pays for itself in the reduction of costly errors or schedule delays. As software teams grow and/or become more distributed, understanding software architecture becomes even more vital. Dependency analysis allows everyone on the team to have a clear understanding of the architecture (what components depend on other components, etc.).

Reduce Technical Debt

Technical debt is a common concept in modern software development practices that happens when there is no core set of well-defined and enforceable design rules for a code base combined with a culture that does not value creating technical wealth. By establishing and iteratively improving design rules, using dependency analysis, technical debt is paid down and the code base is easier to understand and maintain. Development accelerates and this establishes technical wealth.

Understand Impact of Change

Automatic impact analysis, a feature of dependency analysis, will highlight which parts of the application will be affected by planned codebase changes, such as replacement of modules or third party libraries. Understanding the impact of changes enables teams to quickly and accurately respond to change requests. With impact analysis, teams can be responsive while maintaining control over scope and customer expectations. Impact analysis helps developers calculate the impact of change.

How dependency analysis works

From the perspective of software dependency analysis, there must be an initial parsing phase to gather all the data. There can be different qualities of parsing depending on the approach taken. Some approaches rely on a scan of directories containing the codebase in question whilst other mechanisms exist where the codebase is compiled, resulting in a more complete and precise picture. Once the “database” of information exists, it is loaded into some tool that provides a suitable way of visualising the structure of the codebase, and functionality to assist the user in understanding and improving the architecture.

- **Extract:** Dependencies can be extracted using a variety of tools and techniques depending on the language or software that is being analysed. This can be as simple as scanning directories for languages such as C# or Java, to using static analysis tools for languages such as C and C++.
- **Import:** Most enterprise class dependency analysis tools have more than one import

mechanism for dependencies. This will generally include a number of different supported languages, possibly with the ability for the definition of custom import processes.

- **Interact:** Once you have the data imported dependency analysis tools allow you to interact with the dependencies to visualise the implementation against a 'reference' architecture. Features such as impact analysis and change logs can be used to help ensure there is no 'architecture creep' in the implementation.

OPENMP PROGRAMMING

OpenMP is a library for parallel programming in the SMP (symmetric multi-processors, or shared-memory processors) model. When programming with OpenMP, all threads share memory and data. OpenMP supports C, C++ and Fortran. The OpenMP functions are included in a header file called omp.h .

OpenMP program structure: An OpenMP program has sections that are sequential and sections that are parallel. In general an OpenMP program starts with a sequential section in which it sets up the environment and initializes the variables.

When run, an OpenMP program will use one thread (in the sequential sections), and several threads (in the parallel sections). There is one thread that runs from the beginning to the end, and it's called the *master thread*. The parallel sections of the program will cause additional threads to fork. These are called the *slave* threads.

A section of code that is to be executed in parallel is marked by a special directive (omp pragma). When the execution reaches a parallel section (marked by omp pragma), this directive will cause slave threads to form. Each thread executes the parallel section of the code independently. When a thread finishes, it joins the master. When all threads finish, the master continues with code following the parallel section. Each thread has an ID attached to it that can be obtained using a runtime library function (called omp_get_thread_num()). The ID of the master thread is 0.

Introduction to Open Specification for Multi-Processing (OpenMP)

Open MP means Open specifications for MultiProcessing via collaborative work between interested parties from the hardware and software industry, government and academia. It is an Application Program Interface (API) that is used to explicitly direct multi-threaded, shared memory parallelism. API components include Compiler directives, Runtime library routines and Environment variables. Portable because API is specified for C/C++ and Fortran & Implementations on almost all platforms including Unix/Linux and Windows. Standardization is ensured by Jointly defined and endorsed by major computer hardware and software vendors and it is possible to become ANSI standard.

Brief History of OpenMP

In 1991, Parallel Computing Forum (PCF) group invented a set of directives for specifying loop parallelism in Fortran programs. **X3H5**, an ANSI subcommittee developed an ANSI standard based on PCF. In 1997, the first version of OpenMP for Fortran was defined by OpenMP Architecture Review Board. Binding for C/C++ was introduced later. Version 3.1 of it was available since 2011.

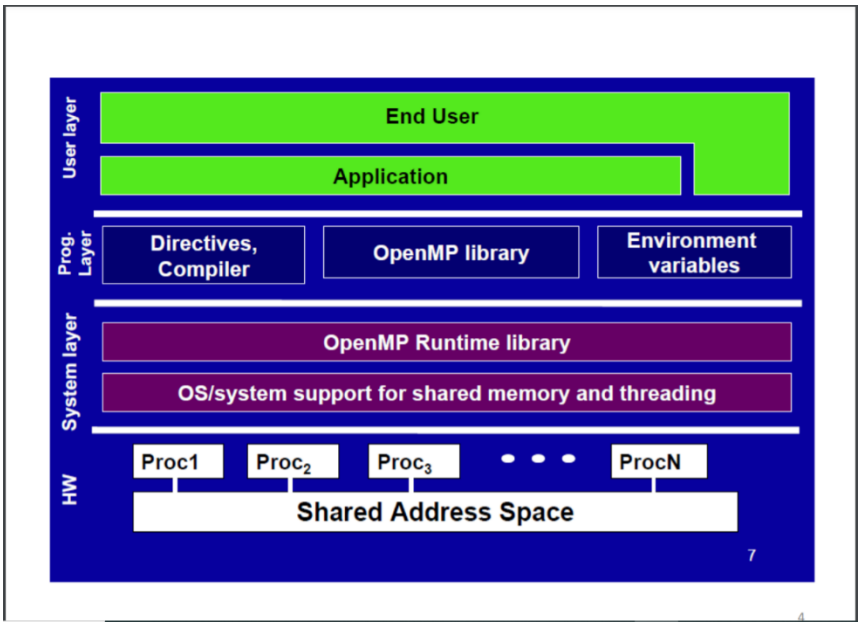


Figure 3: Open Specification for MultiProcessing (OpenMP)

Thread

A process is an instance of a computer program that is being executed. It contains the program code and its current activity. A thread of execution is the smallest unit of a process that can be scheduled by an operating system. Thread model is an extension of the process model where each process consists of multiple independent instruction streams (or threads) that are assigned computer resources by some scheduling procedures. Threads of a process share the address space of this process. Global variables and all dynamically allocated data objects are accessible by all threads of a process. Each thread has its own run-time stack, register, program counter. Threads can communicate by reading/writing variables in the common address space.

A Process

A process contains all the information needed to execute the program.

- Process ID
- Program code
- Data on run time stack
- Global data
- Data on heap

Each process has its own address space. In multitasking, processes are given time slices in a round robin fashion. If computer resources are assigned to another process, the status of the present process has to be saved, in order that the execution of the suspended process can be resumed at a later time.

Differences between threads and processes

A thread is contained inside a process. Multiple threads can exist within the same process and share resources such as memory. The threads of a process share the latter's instructions (code) and its context (values that its variables reference at any given moment). Different processes do not share these resources.

OpenMP Programming Model

OpenMP is based on the existence of multiple threads in the shared memory programming paradigm. A shared memory process consists of multiple threads.

1. **Explicit Parallelism:** In Explicit Parallelism, a Programmer has full control over parallelization. OpenMP is not an automatic parallel programming model.
2. **Compiler Directive Based:** Most OpenMP parallelism is specified through the use of compiler directives which are embedded in the source code. OpenMP is not necessarily implemented identically by all vendors. It is meant for distributed-memory parallel systems (it is designed for shared address spaced machines) but guaranteed to make the most efficient use of shared memory. Required to check for data dependencies, data conflicts, race conditions, or deadlocks. Required to check for code sequences, meant to cover compiler-generated automatic parallelization and directives to the compiler to assist such parallelization. Designed to guarantee that input or output to the same file is synchronous when executed in parallel.
3. **Fork-Join Parallelism:** OpenMP program begin as a single process: the master thread. The master thread executes sequentially until the first parallel region construct is encountered. When a parallel region is encountered, master thread create a group of threads by FORK and becomes the master of this group of threads, and is assigned the thread id 0 within the group. The statement in the program that are enclosed by the parallel region construct are then executed in parallel among these threads.

➔EVALUATION OF PROGRAMS

Program evaluation is a systematic method for collecting, analyzing, and using information to answer questions about projects, policies and programs, particularly about their effectiveness and efficiency. In both the public and private sectors, stakeholders often want to know whether the programs they are funding, implementing, voting for, receiving or objecting to are producing the intended effect. While program evaluation first focuses around this definition, important considerations often include how much the program costs per participant, how the program could be improved, whether the program is worthwhile, whether there are better alternatives, if there are unintended outcomes, and whether the program goals are appropriate and useful. Evaluators help to answer these questions, but the best way to answer the questions is for the evaluation to be a joint project between evaluators and stakeholders.

Evaluation is the systematic application of scientific methods to assess the design, implementation, improvement or outcomes of a program (Rossi & Freeman, 1993; Short, Hennessy, & Campbell, 1996). The term "program" may include any organized action such as media campaigns, service provision, educational services, public policies, research projects, etc. (Center for Disease Control and Prevention [CDC], 1999). The purpose of Programming Evaluation includes:

- Demonstrate program effectiveness to funders
- Improve the implementation and effectiveness of programs
- Better manage limited resources
- Document program accomplishments
- Justify current program funding
- Support the need for increased levels of funding
- Satisfy ethical responsibility to clients to demonstrate positive and negative effects of program participation.
- Document program development and activities to help ensure successful replication

Barriers

Program evaluations require funding, time and technical skills: requirements that are often perceived as diverting limited program resources from clients. Program staff are often concerned that evaluation activities will inhibit timely accessibility to services or compromise the safety of clients. Evaluation can necessitate alliances between historically separate community groups (e.g. academia, advocacy groups, service providers. Mutual misperceptions regarding the goals and process of evaluation can result in adverse attitudes.

Overcoming Barriers

Collaboration is the key to successful program evaluation. In evaluation terminology, stakeholders are defined as entities or individuals that are affected by the program and its evaluation. Involvement of these stakeholders is an integral part of program evaluation. Stakeholders include but are not limited to program staff, program clients, decision makers, and evaluators. A participatory approach to evaluation based on respect for one another's roles and equal partnership in the process overcomes barriers to a mutually beneficial evaluation. Identifying an evaluator with the necessary technical skills as well as a collaborative approach to the process is integral. Programs have several options for identifying an evaluator. Health departments, other state agencies, local universities, evaluation associations and other programs can provide recommendations. Additionally, several companies and university departments providing these services can be located on the internet. Selecting an evaluator entails finding an individual who has an understanding of the program and funding requirements for evaluations, demonstrated experience, and knowledge of the issue that the program is targeting.

Types of Evaluation

Various types of evaluation can be used to assess different aspects or stages of program development. As terminology and definitions of evaluation types are not uniform, an effort has been made to briefly introduce a number of types here.

1. **Context Evaluation:** Investigating how the program operates or will operate in a particular social, political, physical and economic environment. This type of evaluation could include a community needs or organizational assessment.
2. **Formative Evaluation:** Assessing needs that a new program should fulfil (Short, Hennessy, & Campbell, 1996), examining the early stages of a program's development (Rossi & Freeman, 1993), or testing a program on a small scale before broad dissemination (Coyle, Boruch, & Turner, 1991). Sample question: Who is the intended audience for the program?
3. **Process Evaluation:** Examining the implementation and operation of program components. Sample question: Was the program administered as planned?
4. **Impact Evaluation:** Investigating the magnitude of both positive and negative changes produced by a program (Rossi & Freeman, 1993). Some evaluators limit these changes to those occurring immediately (Green & Kreuter, 1991). Sample question: Did participant knowledge change after attending the program?
5. **Outcome Evaluation:** Assessing the short and long-term results of a program. Sample question: What are the long-term positive effects of program participation?

Performance or Program Monitoring

Similar to process evaluation, differing only by providing regular updates of evaluation results to stakeholders rather than summarizing results at the evaluation's conclusion (Rossi & Freeman, 1993; Burt, Harrell, Newmark, Aron, & Jacobs, 1997).

Evaluation Standards and Designs

Evaluation should be incorporated during the initial stages of program development. An initial step of the evaluation process is to describe the program in detail. This collaborative activity can create a mutual understanding of the program, the evaluation process, and program and evaluation terminology. Developing a program description also helps ensure that program activities and objectives are clearly defined and that the objectives can be measured. In general, the evaluation should be feasible, useful, culturally competent, ethical and accurate. Data should be collected over time using multiple instruments that are valid, meaning they measure what they are supposed to measure, and reliable, meaning they produce similar results consistently. The use of qualitative as well as quantitative data can provide a more comprehensive picture of the program. Evaluations of programs aimed at violence prevention should also be particularly sensitive to issues of safety and confidentiality.

Experimental designs are defined by the random assignment of individuals to a group participating in the program or to a control group not receiving the program. These ideal experimental conditions are not always practical or ethical in "real world" constraints of program delivery. A possible solution to blending the need for a comparison group with feasibility is the quasi-experimental design in which an equivalent group (i.e. individuals receiving standard services) is compared to the group participating in the target program. However, the use of this design may introduce difficulties in attributing the causation of effects to the target program. While non-experimental designs may be easiest to implement in a program setting and provide a large quantity of data, drawing conclusions of program effects are difficult.

Logic Models

Logic models are flowcharts that depict program components. These models can include any number of program elements, showing the development of a program from theory to activities and outcomes. Infrastructure, inputs, processes, and outputs are often included. The process of developing logic models can serve to clarify program elements and expectations for the stakeholders. By depicting the sequence and logic of inputs, processes and outputs, logic models can help ensure that the necessary data are collected to make credible statements of causality.

Communicating Evaluation Findings

Preparation, effective communication and timeliness in order to ensure the utility of evaluation findings. Questions that should be answered at the evaluation's inception include: what will be communicated? to whom? by whom? and how? The target audience must be identified and the report written to address their needs including the use of non-technical language and a user-friendly format (National Committee for Injury Prevention and Control, 1989). Policy makers, current and potential funders, the media, current and potential clients, and members of the community at large should be considered as possible audiences. Evaluation reports describe the process as well as findings based on the data

➔ DISTRIBUTED SYSTEMS

A Distributed system is a computing environment in which various components are spread across multiple computers (or other computing devices) on a network. These devices split up the work, coordinating their efforts to complete the job more efficiently than if a single device had been responsible for the task. Distributed systems reduce the risks involved with having a single point of failure, bolstering reliability and fault tolerance. Modern distributed systems are generally designed to be scalable in near real-time and additional computing resources can be added on the fly to increasing performance and further reducing time to completion.

Earlier, distributed computing was expensive, complex to configure and difficult to manage, but Software as a Service (SaaS) platforms has offered expanded functionality, distributed computing has become more streamlined and affordable for businesses, large and small, all types of computing jobs be it database management, video games or Softwares cryptocurrency systems, scientific simulations, blockchain technologies and AI platforms all use Distributed Systems platforms.

How a distributed system works

Distributed systems have evolved over time but today's most common implementations are largely designed to operate via the internet and, more specifically, the cloud

For Example:

- A distributed system begins with a task, such as rendering a video to create a finished product ready for release.
- The web application, or distributed applications, managing this task — like a video editor on a client computer:
- splits the job into pieces
- An algorithm gives one frame of the video to each of a dozen different computers (or nodes) to complete the rendering
- Once the frame is complete, the managing application gives the node a new frame to work on
- This process continues until the video is finished and all the pieces are put back together

Distributed Systems turns a task that might have taken days for a single computer to complete into one that is finished in a matter of minutes.

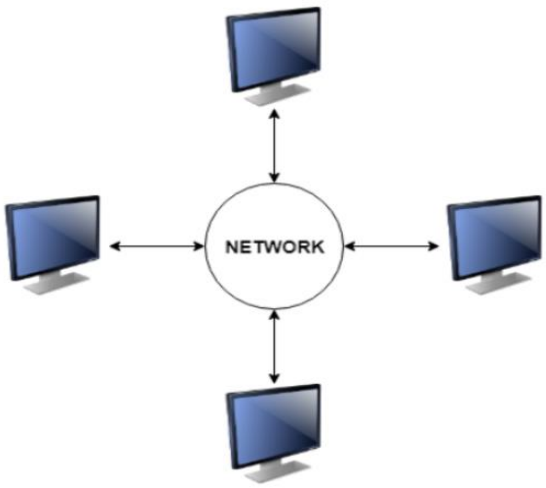


Figure 3.1.1: Distributed Operating Systems

There are many models and architectures of distributed systems in use today.
Client-server systems, the most traditional and simple type of distributed system, involve a multitude of networked computers that interact with a central server for data storage, processing or other common goal. Cell phone networks are an advanced type of distributed system that share workloads among handsets, switching systems and internet-based devices. Peer-to-peer networks, in which workloads are distributed among hundreds or thousands of computers all running the same software, are another example of a distributed system architecture. The most common forms of distributed systems in the enterprise today are those that operate over the web, handing off workloads to dozens of cloud- based virtual server instances that are created as needed, then terminated when the task is complete.

Key Characteristics of a Distributed System

Distributed systems are commonly defined by the following key characteristics and features:

- Scalability: The ability to grow as the size of the workload increases is an essential feature of distributed systems, accomplished by adding additional processing units or nodes to the network as needed.
- Concurrency: Distributed system components run simultaneously. They are also characterized by the lack of a "global clock," when tasks occur out of sequence and at different rates.
- Availability/fault tolerance: If one node fails, the remaining nodes can continue to operate without disrupting the overall computation.
- Transparency: An external programmer or end user sees a distributed system as a single computational unit rather than as its underlying parts, allowing users to interact with a single logical device rather than being concerned with the system's architecture.
- Heterogeneity: In most distributed systems, the nodes and components are often asynchronous, with different hardware, middleware, software and operating systems. This allows the distributed systems to be extended with the addition of new components.
- Replication: Distributed systems enable shared information and messaging, ensuring consistency between redundant resources, such as software or hardware components, improving fault tolerance, reliability and accessibility.

Distributed Tracing

Distributed tracing, sometimes called distributed request tracing, is a method for monitoring applications — typically those built on a microservices architecture — which are commonly deployed on distributed systems. Distributed tracing is essentially a form of distributed computing in that it is commonly used to monitor the operations of applications running on distributed systems.
In software development and operations, tracing is used to follow the course of a transaction as it travels through an application — an online credit card transaction as it winds its way from a customer's initial purchase to the verification and approval process to the completion of the transaction, for example. A tracing system monitors this process step by step, helping a developer to uncover bugs, bottlenecks, latency or other problems with the application.
Distributed tracing is necessary because of the considerable complexity of modern software architectures. A distributed tracing system is designed to operate on a distributed services infrastructure, where it can track multiple applications and processes simultaneously across numerous concurrent nodes and computing environments. Without distributed tracing, an

application built on a microservices architecture and running on a system as large and complex as a globally distributed system environment would be impossible to monitor effectively.

Benefits of Distributed Systems

Distributed systems offer a number of advantages over monolithic, or single, systems, including:

- Greater flexibility: It is easier to add computing power as the need for services grows. In most cases today, you can add servers to a distributed system on the fly.
- Reliability: A well-designed distributed system can withstand failures in one or more of its nodes without severely impacting performance. In a monolithic system, the entire application goes down if the server goes down.
- Enhanced speed: Heavy traffic can bog down single servers when traffic gets heavy, impacting performance for everyone. The scalability of distributed databases and other distributed systems makes them easier to maintain and also sustain high-performance levels.
- Geo-distribution: Distributed content delivery is both intuitive for any internet user, and vital for global organizations.

What are some challenges of distributed systems?

Distributed systems are considerably more complex than monolithic computing environments, and raise a number of challenges around design, operations and maintenance. These include:

- **Increased opportunities for failure**: The more systems added to a computing environment, the more opportunity there is for failure. If a system is not carefully designed and a single node crashes, the entire system can go down. While distributed systems are designed to be fault tolerant, that fault tolerance isn't automatic or foolproof.
- **Synchronization process challenges:** Distributed systems work without a global clock, requiring careful programming to ensure that processes are properly synchronized to avoid transmission delays that result in errors and data corruption. In a complex system — such as a multiplayer video game — synchronization can be challenging, especially on a public network that carries data traffic.
- **Imperfect scalability**: Doubling the number of nodes in a distributed system doesn't necessarily double performance. Architecting an effective distributed system that maximizes scalability is a complex undertaking that needs to take into account load balancing, bandwidth management and other issues.
- **More complex security**: Managing a large number of nodes in a heterogeneous or globally distributed environment creates numerous security challenges. A single weak link in a file system or larger distributed system network can expose the entire system to attack.
- **Increased complexity**: Distributed systems are more complex to design, manage and understand than traditional computing environments.

The risks of distributed systems

The challenges of distributed systems as outlined above create a number of correlating risks. These include:

- Security: Distributed systems are as vulnerable to attack as any other system, but their distributed nature creates a much larger attack surface that exposes organizations to threats.
- Risk of network failure: Distributed systems are beholden to public networks in order to transmit and receive data. If one segment of the internet becomes unavailable or overloaded, distributed system performance may decline.

- Governance and control issues: Distributed systems lack the governability of monolithic, single-server-based systems, creating auditing and adherence issues around global privacy laws such as GDPR. Globally distributed environments can impose barriers to providing certain levels of assurance and impair visibility into where data resides.
- Cost control: Unlike centralized systems, the scalability of distributed systems allows administrators to easily add additional capacity as needed, which can also increase costs. Pricing for cloud-based distributed computing systems are based on usage (such as the number of memory resources and CPU power consumed over time). If demand suddenly spikes, organizations can face a massive bill.

➔ SYSTEMS MODELS

Systems modelling is the interdisciplinary study of the use of models to conceptualize and construct systems in business and IT development. A common type of systems modeling is function modeling, with specific techniques such as the Functional Flow Block Diagram and IDEF0. These models can be extended using functional decomposition, and can be linked to requirements models for further systems partition. Contrasting the functional modeling, another type of systems modeling is architectural modeling which uses the systems architecture to conceptually model the structure, behavior, and more views of a system.

A **system** is a simplified representation of reality. "System" is a common word, often used with loose meaning. Whereas in the real world, a "system" may seem at times an endless series of connected elements, we refer here to a system as a series of selected, chosen elements with specified boundaries and pre-determined time characteristics.

A 'simple' system could for instance be a nearby coffee shop. This coffee shop has customers who place orders and staff who process them. There may be at times very few customers, whereas at others, the place is very busy (say, because the coffee shop is just nearby the University, and has free wi-fi, which the students use while enjoying a coffee and chat with their friends). So, for the customers, and the staff too, time is not neutral. It is then useful to look at our coffee-shop-system over a series of sections of time (time steps) that make a day. Perhaps an appropriate time step of one hour is adequate: it is more than enough to encapsulate long hours when little really happens, but is just enough to capture events at peak time. So much, though, may happen in one hour over a cup of coffee, when the place is busy, people meet, many orders are placed, many messages received. Perhaps, a time step of 30 minutes, or even 15 minutes might then be better. So, although many near-empty 15-minute segments might be a waste of computing time, and lead to outputs that may be boring for some parts of the day, these might ensure that important events are not lost at peak time. Yet - so many things may still happen over a period of 15 minutes. Might a time step of 5 minutes be safer? This is obviously not an easy question.

At any rate, a decision must be made, and it is up to the modeler to make it. Each system, such as the coffee-shop-system, has a **time constant**, which we can simply define for the time being as the delay over which the system may strongly change, or, in systems analysis phrasing: over which the **state** of the system may change. One way to empirically choose a time constant is based on experience and knowledge of the system at hand. Note that in the coffee-shop-system, not all the elements are enclosed within the coffee shop itself, which are important for the coffee- shop-system: for instance, it has free wi-fi. We therefore can call it a semi- open system. Biological systems, phytopathological systems in particular, are **semi-open**: they receive and transmit information, components, biomass, or energy from and to their environment.

A model & **Systems Models**

A model is a computer program that describes the mechanics of the considered system. The encoding of a model can be made in many ways.

Systems Models

A system is a set of elements that relate to each other in some manner. The elements of a system can be objects, people, organizations, processes, descriptions or even ideas. The relationships between these elements can include different kinds of influence, flows of information, resources, associations, temporal relationships, or origins.

Models of systems therefore try to capture these relationships in a way that gives a perspective on how the system as a whole interacts.

System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system. It is about representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML). Models help the analyst to understand the functionality of the system; they are used to communicate with customers.

Models' Different perspectives:

- An external perspective, where you model the context or environment of the system.
- An interaction perspective, where you model the interactions between a system and its environment, or between the components of a system.
- A structural perspective, where you model the organization of a system or the structure of the data that is processed by the system.
- A behavioral perspective, where you model the dynamic behavior of the system and how it responds to events.

UML diagrams

Five types of UML diagrams that are the most useful for system modeling:

- **Activity** diagrams, which show the activities involved in a process or in data processing.
- **Use case** diagrams, which show the interactions between a system and its environment.
- **Sequence** diagrams, which show interactions between actors and the system and between system components.
- **Class diagrams**, which show the object classes in the system and the associations between these classes.
- **State** diagrams, which show how the system reacts to internal and external events.

Models of both new and existing system are used during **requirements engineering**. Models of the **existing systems** help clarify what the existing system does and can be used as a basis for discussing its strengths and weaknesses. These then lead to requirements for the new system.

Models of the **new system** are used during requirements engineering to help explain the proposed requirements to other system stakeholders. Engineers use these models to discuss design proposals and

to document the system for implementation.

Types of Systems Models

Context and process models

Context models are used to illustrate the operational context of a system, they show what lies outside the system boundaries. Social and organizational concerns may affect the decision on where to position system boundaries. Architectural models show the system and its relationship with other systems.

System boundaries are established to define what is inside and what is outside the system. They show other systems that are used or depend on the system being developed. The position of the system boundary has a profound effect on the system requirements. Defining a system boundary is a political judgment since there may be pressures to develop system boundaries that increase/decrease the influence or workload of different parts of an organization.

Context models simply show the other systems in the environment, not how the system being developed is used in that environment. **Process models** reveal how the system being developed is used in broader business processes. UML activity diagrams may be used to define business process models.

The example below shows a UML **activity diagram** describing the process of involuntary detention and the role of MHC-PMS (mental healthcare patient management system) in it.

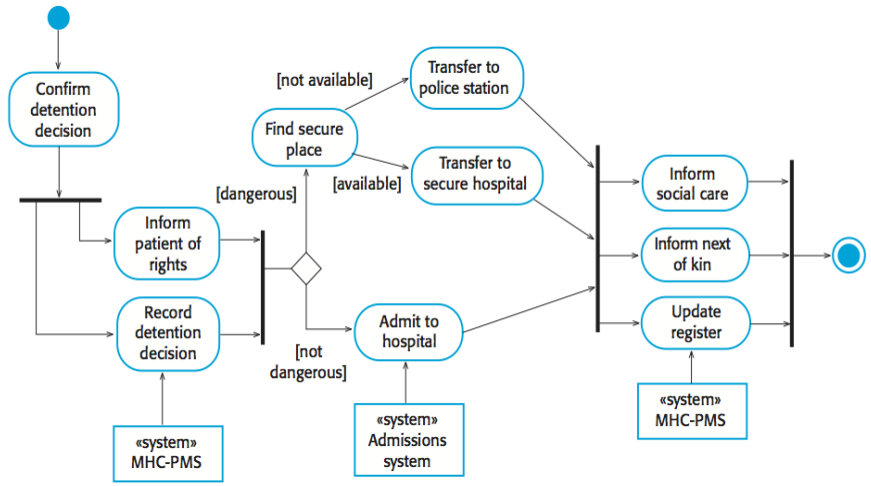


Figure: UML activity diagram for involuntary detention and the role of MHC-PMS

Interaction models

Types of interactions that can be represented in a model:

- Modeling user interaction is important as it helps to identify user requirements.
- Modeling system-to-system interaction highlights the communication problems that may arise.
- Modeling component interaction helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.

Structural models

Structural models of software display the organization of a system in terms of the components that make up that system and their relationships. Structural models may be **static** models, which show the

structure of the system design, or **dynamic** models, which show the organization of the system when it is executing. You create structural models of a system when you are discussing and designing the system architecture.

Behavioral models

Behavioral models are models of the dynamic behavior of a system as it is executing. They show what happens or what is supposed to happen when a system responds to a stimulus from its environment. Two types of stimuli:

- Some data arrives that has to be processed by the system
- Some event happens that triggers system processing. Events may have associated data, although this is not always the case.

Many business systems are data-processing systems that are primarily driven by data. They are controlled by the data input to the system, with relatively little external event processing.

Data-driven models show the sequence of actions involved in processing input data and generating an associated output. They are particularly useful during the analysis of requirements as they can be used to show end-to-end processing in a system. **Data-driven models** can be created using UML **activity diagrams** or UML **sequence diagrams.**

Real-time systems are often event-driven, with minimal data processing. For example, a landline phone switching system responds to events such as 'receiver off hook' by generating a dial tone.

Event-driven models shows how a system responds to external and internal events. It is based on the assumption that a system has a finite number of states and that events (stimuli) may cause a transition from one state to another. Event-driven models can be created using UML **state diagrams.**

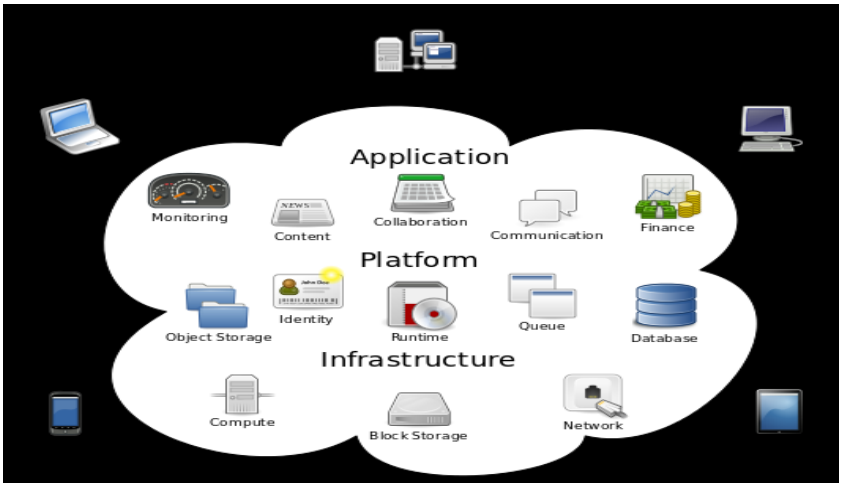
➔Mobile & Cloud Computing

Cloud Computing

Cloud Computing is the delivery of the computing services, such as servers, databases, storage and networking – over the Internet. These services, usually are offered by so called Cloud Providers, that usually charge based on usage.

Nowadays, everyone that is using a device connected to Internet, might be user of cloud services, even though we might not be aware of it. Almost every online service, including email, document editors or entertaining apps, might be running using cloud services.

Figure: Cloud Computing



Capabilities of Cloud Computing

Generally, these are a few of the capabilities of Cloud Computing:

- Create new apps and services
- Store, back up and recover data
- Deliver software
- Analyse data for pattern recognition
- Streaming.

Besides the capabilities that Cloud Computing provides, there are also a lot of benefits that it can offer.

- Cost – using cloud services lowers the costs that organizations need to spend for buying hardware and software tools for setting up the infrastructure for its needs.
- Speed – when the organization needs more resources, provisioning additional resources in cloud services can be done in minutes.
- Scaling – the ability to scale elastically on demand using cloud services appears as their main and most common use case – processing power, storage, bandwidth and whatever the demand is, in less than a minute.

Categories of Cloud Computing models

Depending on the type of service a Service Providers provides, there are several categories of Cloud Computing models, as listed:

1. **Software as a Service (SaaS):** The providers that provide this model of Cloud Computing solutions usually provide a web-based application where the users of the service can operate. In this model, the consumer does not have any control over the infrastructure in which the service is running, including the network, servers, storage or operating system. It removes the need that several organizations or companies will have to install and run their applications or services on their data centers or company computers. By this, the organizations save a lot of financial resources by saving money on the hardware they would need to run the application, the rent of space where the data center would be located on, or even software license for operating systems and depending software.
2. **Platform as a Service (PaaS):** Platform as a Service is another Cloud Computing model in which the third-party provider provides the necessary hardware and software tools usually required for development or research – over the Internet. In other words, all the programming languages, libraries, services and other programming tools provided by the provider are deployed in the cloud infrastructure that the provider provides. Similar as in the previous model, SaaS, the end user does not have any control nor have to manage any part of the infrastructure, such as network, operating systems and storage.
3. **Infrastructure as a Service (IaaS):** According to most of the information provided by different surveys, IaaS is the most common cloud-based model provided by the service providers. IaaS refers to the service providers who provide processing capability, storage, network and other fundamental computing resources, to the consumer who wants to run any type of software. Usually these services are made possible by using virtual machines as instances. Xen, Oracle VirtualBox, KVM or Hyper-V are typical examples of providers that offer great possibilities to run these VMs.

Mobile Cloud Computing (MCC)

In the consumer space, mobility players such as Apple, Google, and Microsoft all offer variants of cloud-based apps and private storage. However, the line between the individual and the professional is increasingly being blurred. Allowing employees access to company resources using private devices makes them expect access to your CRM system on their iPad, with (near) real-time business intelligence reports delivered by the touch of a finger while sharing analysis with their teams on the collaboration platform.

Most of the companies tend to move their apps and services in the cloud. Every company's mission is to grow and evolve. Considering this case, organizations face trouble with new coming employees, which bring their own devices, services and apps. This means that, it requires more efforts and time to integrate the data to the corporate cloud, in order to ensure support and control over usage of the same. When we add the complex format of making sure that corporate services are up to date, all this process becomes a mess and quite often it becomes a challenging task for the responsible employees.

Advantages of Mobile & Cloud Computing

Mobile Cloud Computing offers a bunch of advantages while using cloud services. Following are listed some of the most important ones:

- Flexibility – one of key advantages while using MCC is that the cloud information can be used anywhere, everywhere; all you need is a mobile device of any kind, which is paired or configured with the organization cloud platform.
- Real time available data – accessing the data in real time is no longer a challenge while you are out of the office.
- No upfront payments – last, but not least – payments. Commonly, cloud applications does not require payment without using it. It is mostly the case pay-for-use, which helps in growing the adoption of the model.

Disadvantages of Mobile & Cloud Computing (MCC)

Whenever there are advantages on any issue, it is sure there would be the disadvantages as well. The following are some listed and most important disadvantages of Mobile and Cloud Computing.

- Security – a major concern with Cloud Computing is the security and data integration. When mobile is the subject, the attention must be two times higher: unprotected information can be easily sniffed.
- Internet connection – considering the flexibility of MCC, allowing the users to access the data from anywhere, requires Internet connection. Making sure that, when accessing data, the user have access to strong and stable Internet connection, often can cause headache, especially in non-metropolitan areas.
- Performance – considering smaller size and lower hardware performance, it is understandable that the performance with MCC will be in a much lower level.

Mobile Cloud Computing Security Concerns

One of the most significant concerns of Cloud Computing in general and Mobile and Cloud Computing particularly, is data security.

Mobile devices are at the top of the list of the most significant security risks. Confidentiality, integrity

and authenticity of information are the most particular threat. Confidentiality is considered a risk when unauthorized parties manage to intercept data transmission. Allowing such a thing, risks the integrity of the data. The authenticity is risked when these unauthorized parties can use the devices to trigger transactions. The latest trends of using mobile devices is by using free applications, which can be infected by malicious software. Using open channels over network threatens confidential information. Thus, these applications are often updated or upgraded, trying to provide as much security as possible.

The Top Threats in the usage of Mobile and Cloud Computing

1. **Data Loss:** Using Cloud Computing is more like outsourcing the data to the service provider. This means increasing the risk of exposing important data which were not issues in traditional computing. Since more of the service providers provide shared resources, it is more likely for the transactions to crash and data to be lost. Recently, there has been a lot of unintentional deletion of data by the providers. Also, a bad line code can mess up access keys, and the data is lost. The following solutions can lower the risk:
    - Encryption of data while transmission;
    - Using access control tools
    - Time-to-time back up
2. **Untrusted service providers** Known as *malicious insiders*, they are the people who have access and authorization to manage data in the care of the service providers, offering cloud services. These people can either be working for other companies or they do it for their personal intentions.
3. **Insecure API:** Usually, the communication between a client (in this case, a mobile device which is handled by the company's employee) and the server (which is somewhere in the cloud) is done by an Application Programming Interface. In order to keep data integration and security in a higher level, the company providing the API should secure the communication channels and the information transmitted. Avoiding insecure APIs can be achieved by using the following techniques:
    - Applying authentication and access control tools on data transmission channels
    - Implementing the proper security model according to service provider's security protocols

➔ **Wireless Local Area Network (WLAN)**

A wireless local-area network (WLAN) is a group of colocated computers or other devices that form a network based on radio transmissions rather than wired connections. A Wi-Fi network is a type of WLAN.

Although some may use the terms "Wi-Fi" and "WLAN" interchangeably but they are not the same. A "Wi-Fi connection" refers to a given wireless connection that a device uses, the WLAN is the network itself. "Wi-Fi" is a superset of the IEEE 802.11 standard and is sometimes used interchangeably with that standard. However, not every Wi-Fi device actually receives Wi-Fi Alliance certification, although Wi-Fi is used by more than 700 million people through about 750,000 Internet connection hot spots. The hot spots themselves also constitute WLANs, of a particular kind.

A wireless local area network (WLAN) is a wireless distribution method for two or more devices.

WLANs use high-frequency radio waves and often include an access point to the Internet. A WLAN allows users to move around the coverage area, often a home or small office, while maintaining a network connection. A WLAN is sometimes called a local area wireless network (LAWN).

WLANs and Access Points

Every component that connects to a WLAN is considered a station, and falls into one of two categories: access points (APs) and clients.

- **Access points or APs** transmit and receive radio frequency signals with devices that are able to receive transmitted signals; they normally function as routers.
- **Clients**, on the other hand, may include a variety of devices, such as desktop computers, workstations, laptop computers, IP phones and other cell phones and smartphone devices.

All stations able to communicate with each other are called basic service sets (BSSs), of which there are two types: independent and infrastructure. Independent BSSs (IBSS) exist when two clients communicate without using APs, but cannot connect to any other BSS. Such WLANs are called a peer-to-peer or an ad-hoc WLANs. The second BSS is called an infrastructure BSS. It may communicate with other stations but only in other BSSs and it must use APs.

Emerging WLANs and its Ubiquity

In the early 1990s, WLANs were very expensive, and were only used when wired connections were strategically impossible. By the late 1990s, most WLAN solutions and proprietary protocols were replaced by IEEE 802.11 standards in various versions (versions "a" through "n"). WLAN prices also began to decrease significantly. As technology progressed, WLANs became easier and easier to set up and administrate. That led to the emergence of the ISP WLAN, where so many small local home networks are mostly coordinated by the Internet Service Provider, and not engineered by the end-user on-site.

In these types of ISP WLAN setups, the ISP’s modem is the access point. It's also the router. All that the consumer has to do is plug in the router, use provided security passwords, and connect home devices to the home WLAN. You could call this “wireless local area network as a service” (WLANaaS) or refer to a “plug-and-play” or abstracted wireless local area network model. In any case, it’s ultimately very convenient for the household. Although ISPs don't usually advertise their products as home LANs, that’s what they are. Some types of ISP services talk about using the modem as a “gateway” to the Internet, which implies that your WLAN is on the other side of that gateway. Users of home WLANs are more frequently connecting devices such as phones, televisions, computers and printers to evolved WLAN systems where the ISP will offer some type of dashboard visualization for the WLAN in question.

There's also been some innovation toward peer-to-peer WLANs that work without a defined access point. In other words, all of the devices are independently operated to network together. This challenges the traditional idea that the WLAN was made of access points and clients, as discussed above. At the same time, in the client/server architecture, where a similar approach is used to engineer Internet services, peer-to-peer systems are also challenging that traditional build as well. As the IoT paves the way for advanced connectivity, the WLAN provides that “sub- network” and the convenience of local Wi-Fi operation.

How a WLAN works

Like broadcast media, a WLAN transmits information over radio waves. Data is sent in packets. The packets contain layers with labels and instructions that, along with the unique MAC (Media Access Control) addresses assigned to endpoints, enable routing to intended locations.

Configuration of WLAN

A WLAN can be configured in one of two ways:

a. **Infrastructure**

A home or office Wi-Fi network is an example of a WLAN set up in infrastructure mode. The endpoints are all connected and communicate with each other through a base station, which may also provide internet access.

A basic infrastructure WLAN can be set up with just a few parts: a wireless router, which acts as the base station, and endpoints, which can be computers, mobile devices, printers, and other devices. In most cases, the wireless router is also the internet connection.

b. **Ad hoc**

In this setup, a WLAN connects endpoints such as computer workstations and mobile devices without the use of a base station. Use of Wi-Fi Direct technology is common for an ad hoc wireless network. An ad hoc WLAN is easy to set up and can provide basic peer-to-peer (P2P) communication. An ad hoc WLAN requires only two or more endpoints with built-in radio transmission, such as computers or mobile devices. After adjusting network settings for ad hoc mode, one user initiates the network and becomes visible to the others.

How Roaming works on a WLAN

For any sized network, access points can extend the area of access. Wi-Fi standards are designed to allow a non-stationary user's connection to jump from one access point to another, though some users and applications may experience brief dropouts. Even with non-overlapping access points, a user's connection is simply paused until connection with the next access point. Additional access points can be wired or wireless. When access points overlap, they can be configured to help optimize the network by sharing and managing loads.

WLAN Architecture

a. **Stations:** Stations are components that connect wirelessly to networks. They are either access points or endpoints, each identified with a unique network address.
b. **Basic Service Set (BSS)**: A BSS is a group of stations that connects to the network. In ad hoc networks, the group of stations is called an Independent BSS (IBSS). A set of connected BSSs, as in a network with multiple access points, is called an Extended Service Set (ESS).
c. **Distribution system:** The distribution system connects access points in an ESS. The connections can be wired or wireless. A wireless distribution system (WDS) can use mesh or its own WDS protocol. Fixed wireless is a specialized form of radio transmission for connecting a geographically distant access point.
d. **Access point**: The access point is the base station that serves as a hub to which other stations connect. The "access" is that of the stations to the network, but it may also mean internet access, since many routers double as internet modems. In an ESS, access points may be

connected with Ethernet cables or wirelessly.

e. **Bridge**: The bridge is used to connect a WLAN to a LAN or to an access point.
f. **Endpoint:** The endpoint is any end-user station, such as a computer, mobile device, printer, or Internet of Things (IoT) device.

Benefits of a WLAN

1. Extended reach: WLANs enable computing to happen anywhere, even when carrying high data loads and advanced web applications.
2. Device flexibility: A WLAN supports use of a wide range of devices, such as computers, phones, tablets, gaming systems, and IoT devices.
3. Easier installation and management: A WLAN requires less physical equipment than a wired network, which saves money, reduces installation time, and takes up less of a footprint in office settings.
4. Scalability: A WLAN is easy to scale. Adding users is as simple as assigning login credentials.
5. Network management: Nearly all management of a WLAN can be handled virtually. A single software interface can provide visibility, manage users, monitor network health, and collect data.

IEEE 802.11 Standard

IEEE 802.11 is the set of technical guidelines for implementing Wi-Fi. Selling products under this trademark is overseen by an industry trade association by the name of the Wi-Fi Alliance.

IEEE 802.11 has its roots from a 1985 decision by the U.S. Federal Commission for Communication that opened up the ISM band for unlicensed use. The standard was formally released in 1997. That original standard was called IEEE 802.11-1997 and is now obsolete.

It's common to hear people refer to "802.11 standards" or the "802.11 family of standards." However, to be more precise, there is only one standard (IEEE 802.11-2007) but many amendments. Commonly known amendments include 802.11a, 802.11b, 802.11g, and 802.11n.

Bluetooth Technology

Bluetooth is a short-range wireless communication technology that allows devices such as mobile phones, computers, and peripherals to transmit data or voice wirelessly over a short distance. The purpose of Bluetooth is to replace the cables that normally connect devices, while still keeping the communications between them secure.

The "Bluetooth" name is taken from a 10th-century Danish king named Harald Bluetooth, who was said to unite disparate, warring regional factions. Like its namesake, Bluetooth technology brings together a broad range of devices across many different industries through a unifying communication standard.

Developed in 1994, Bluetooth was intended as a wireless replacement for cables. It uses the same 2.4GHz frequency as some other wireless technologies in the home or office, such as cordless phones and WiFi routers. It creates a 10-meter (33-foot) radius wireless network, called a personal area network (PAN) or piconet, which can network between two and eight devices. This short-range network allows you to send a page to your printer in another room, for example, without having to run an unsightly cable.

Bluetooth uses less power and costs less to implement than Wi-Fi. Its lower power also makes

it far less prone to suffering from or causing interference with other wireless devices in the same 2.4GHz radio band. Bluetooth range and transmission speeds are typically lower than Wi-Fi (the wireless local area network that you may have in your home). Bluetooth v3.0 + HS i.e Bluetooth high-speed technology devices, can deliver up to 24 Mbps of data, which is faster than the 802.11b WiFi standard, but slower than wireless-a or wireless-g standards. As technology has evolved, however, Bluetooth speeds have increased. The Bluetooth 4.0 specification was officially adopted on July 6, 2010. Bluetooth version 4.0 features include low energy consumption, low cost, multivendor interoperability, and enhanced range.

The hallmark feature enhancement to the Bluetooth 4.0 spec is its lower power requirements; devices using Bluetooth v4.0 are optimized for low battery operation and can run off of small coin-cell batteries, opening up new opportunities for wireless technology. Instead of fearing that leaving Bluetooth on will drain your cell phone's battery, for example, you can leave a Bluetooth v4.0 mobile phone connected all the time to your other Bluetooth accessories.

Connecting With Bluetooth

Many mobile devices have Bluetooth radios embedded in them. PCs and some other devices that do not have built-in radios can be Bluetooth- enabled by adding a Bluetooth dongle, for example. The process of connecting two Bluetooth devices is called "pairing." Generally, devices broadcast their presence to one another, and the user selects the Bluetooth device they want to connect to when its name or ID appears on their device. As Bluetooth-enabled devices proliferate, it becomes important that you know when and to which device you're connecting, so there may be a code to enter that helps ensure you're connecting to the correct device. This pairing process can vary depending on the devices involved. For example, connecting a Bluetooth device to your iPad can involve different steps from those to pair a Bluetooth device to your car.

Bluetooth Limitations

There are some downsides to Bluetooth. The first is that it can be a drain on battery power for mobile wireless devices like smartphones, though as the technology (and battery technology) has improved, this problem is less significant than it used to be. Also, the range is fairly limited, usually extending only about 30 feet, and as with all wireless technologies, obstacles such as walls, floors, or ceilings can reduce this range further. The pairing process may also be difficult, often depending on the devices involved, the manufacturers, and other factors that all can result in frustration when attempting to connect.

Security and Bluetooth

Bluetooth is considered a reasonably secure wireless technology when used with precautions. Connections are encrypted, preventing casual eavesdropping from other devices nearby. Bluetooth devices also shift radio frequencies often while paired, which prevents an easy invasion.

Devices also offer a variety of settings that allow the user to limit Bluetooth connections. The device-level security of "trusting" a Bluetooth device restricts connections to only that specific device. With service-level security settings, you can also restrict the kinds of activities your device is permitted to engage in while on a Bluetooth connection. As with any wireless technology, however, there is always some security risk involved. Hackers have devised a variety of malicious attacks that use Bluetooth networking. For example, "bluesnarfing" refers to a hacker gaining authorized access to information

on a device through Bluetooth; "bluebugging" is when an attacker takes over your mobile phone and all its functions. For the average person, Bluetooth doesn't present a grave security risk when used with safety in mind (e.g., not connecting to unknown Bluetooth devices). For maximum security, while in public and not using Bluetooth, you can disable it completely.

Personal Area Network (PAN)

A personal area network (PAN) is the interconnection of information technology devices within the range of an individual person, typically within a range of 10 meters. For example, a person traveling with a laptop, a personal digital assistant (PDA), and a portable printer could interconnect them without having to plug anything in, using some form of wireless technology. Typically, this kind of personal area network could also be interconnected without wires to the Internet or other networks.

Conceptually, the difference between a PAN and a wireless LAN is that the former tends to be centered around one person while the latter is a local area network (LAN) that is connected without wires and serving multiple users.

In another usage, a personal area network (PAN) is a technology that could enable wearable computer devices to communicate with other nearby computers and exchange digital information using the electrical conductivity of the human body as a data network. For example, two people each wearing business card-size transmitters and receivers conceivably could exchange information by shaking hands. The transference of data through intra-body contact, such as handshakes, is known as **linkup**.