Important Key Concepts and Terminologies – Learn System Design

> **System Design** is the core concept behind the design of any distributed systems. System Design is defined as a process of creating an architecture for different components, interfaces, and modules of the system and providing corresponding data helpful in implementing such elements in systems.

In this article, we'll cover the standard terms and key concepts of system design and performance, such as:

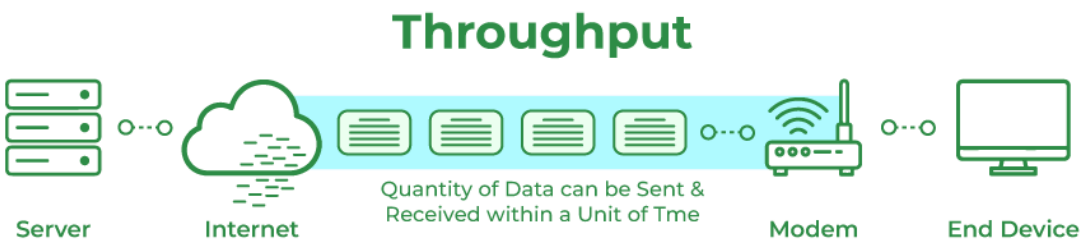
- Latency,
- Throughput,
- Availability,
- Redundancy,
- Time
- CAP Theorem
- Lamport's Logical Clock Theorem.

Important Key Concepts and Terminologies In System Design – Learn System Design

Let us see them one by one.

Throughput in System Design

> **Throughput** is defined as the **measure of amount of data transmitted** successfully in a system, in a certain amount of time. In simple terms, throughput is considered as **how much data is transmitted** successfully over a period of time.

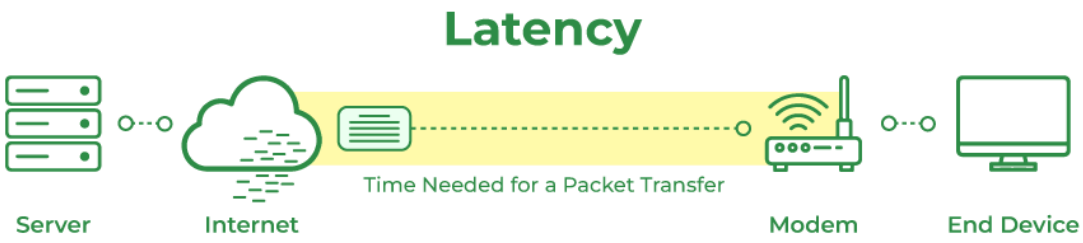


Throughput

The unit of measure for throughput is **bits per second or bps.**

Latency in System Design

> **Latency** is defined as the **amount of time** required **for a single data to be delivered** successfully. Latency is measured in **milliseconds (ms).**



Latency

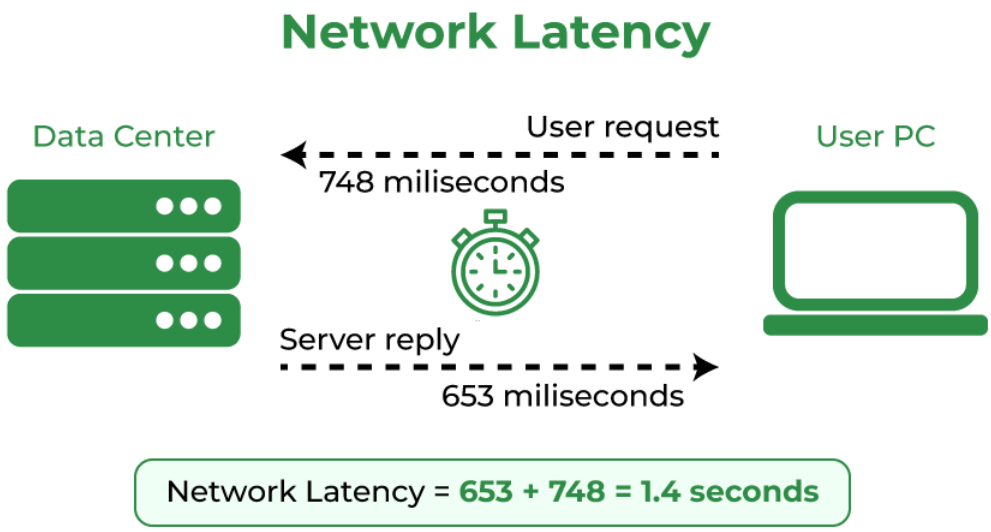
We all have encountered situations where websites and web applications take longer to respond, or there is buffering while playing a video despite having good network connectivity. Then the latency for such systems is said to be comparatively **high**.

There is a certain amount of time required for user input over the website and there is a certain amount of time for the response from the web application to the user. So the **delay between user input and web application response to the same input is known as latency**.

Reasons for high Latency

Now you must be wondering about the factors that are responsible for delays. So high latency mainly depends on 2 factors:

1. Network Delays
2. Mathematical Calculation Process Delays



Network Latency

In **monolithic architecture**, as we know there is only a single block and all network calls are local within hence network delay is zero and hence latency equals computational delays only (which if not latency equals zero in monolithic systems)

> Latency = Mathematic Calculation Delays

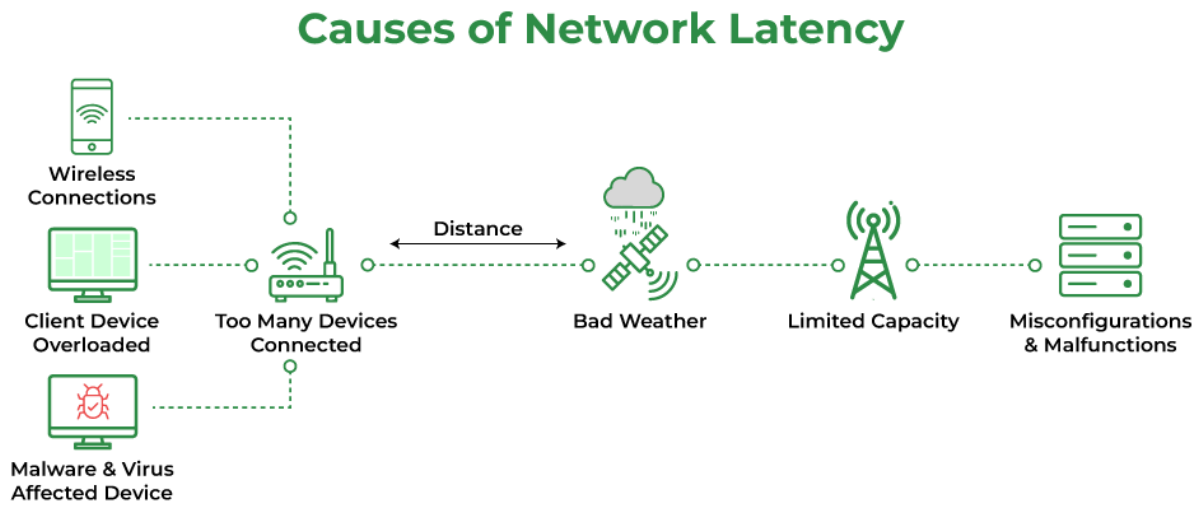
In **distributed systems**, there is a networks over which signals are passed to and fro hence there will for sure be network delay.

> Latency = Mathematic Calculation Delays + Network Delays

Let us finally discuss are components affecting latency:

Components Affecting Latency:

1. **Packet Size:** Smaller the packet chunk size faster the transmission and the lower the latency.
2. **Packet Loss:** Transmission of huge packets of various sizes in medium losses to very few losses in packets.
3. **Medium of transmission:** Optical fiber is the fastestway of transmission.
4. **Distance between Nodes:** Poor signal will increase latency and great connectivity decreases to a greater extent.
5. **Signal strength:** Good signal strength reduces latency.
6. **Storage delays:** Stored information in a database and fetching from it requires very little time which supports increasing latency.



Causes of high network Latency

How to Reduce latency:

1. **Use a content delivery network (CDN):** CDNs help to cut down on latency. In order to shorten the distance between users and reduce the amount of time that data must travel over great distances, CDN servers are situated at various locations.
2. **Upgrading computer hardware/software:** Improving or fine-tuning mechanical, software, or hardware components can help cut down on computational lag, which in turn helps cut down on latency.
3. **Cache:** A cache is a high-speed data storage layer used in computers that temporarily store large amounts of transient data. By caching this data, subsequent requests for it can be fulfilled more quickly than if the data were requested directly from its original storage location. This lessens latency as well.

Applications of Latency

1. Vehicles
2. Capital Market

Availability in System Design

> **Availability** is the **percentage of time the system is up** and working for the needs.

It is a very important factor when for tech companies to provide services while designing systems. As recorded, Meta went down for 6 hours, corresponding to loss of estimated 60 million dollars.

There are levels associated with availability with respect to the service the system is offering. For instance, air traffic control requires a higher level of availability in comparison to the restaurant reservation system.

$$\text{Availability} = \frac{\text{Uptime}}{(\text{Uptime} + \text{downtime})}$$

How availability is measured?

How availability is measured?

Now you must be thinking about what are these levels and how they are measured. Levels in availability are measured via downtime per year via order of ‘nines’. More ‘nines’ lead to lesser downtime.

It is as shown below via table as follows:

Availability(%)	Downtime/Year
90	~36.5 days
99	~3.65 days
99.9	~8.7 Hours
99.99	~52 Minutes
99.999	~6 Minutes

> **Note:** 5 nines is considered as the golden standard of availability the system is available(up) to perform tasks.

How to increase Availability?

1. Eliminate SPOF(major and important)
2. Verify Automatic Failover
3. Use Geographic Redundancy
4. Continue upgrading and improving

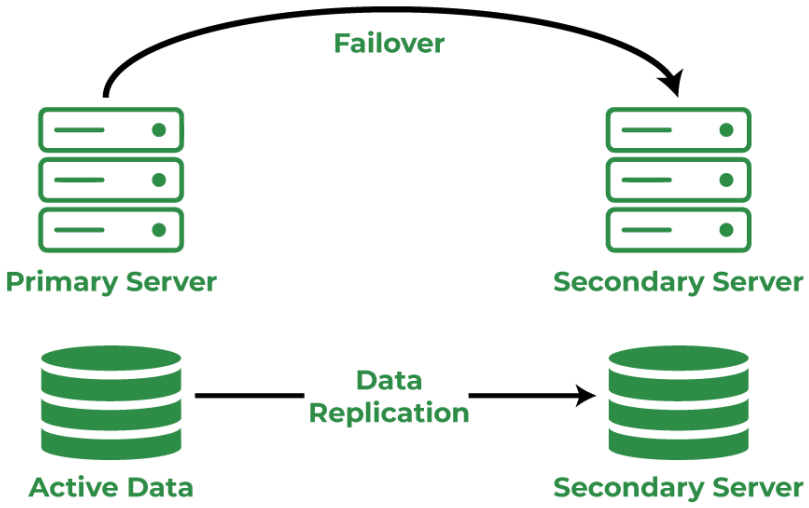
From the above understanding, we can land up with two conclusions:

1. **Availability is low in monolithic architecture** due to SPOF.
2. **Availability is high in distributed architecture** due to redundancy.

Redundancy in System Design

> **Redundancy** is defined as a concept where certain entities are **duplicated with aim to scale up the system and reduce over all down-time**.

For example, as seen in the image below, we are duplicating the server. So if one server goes down, then we have a **redundant server** in our system to balance the load.



Redundancy in System Design

Now from the above image, you must be wondering how these connections are handled meaning how to get all load over to another and not let to connect to server/s that are already down. Here we introduce a new term known as the **load balancer**.

Load Balancer

A load balancer works as a “traffic cop” sitting in front of your server and routing client requests across all servers. It simply distributes the set of requested operations (database write requests, cache queries) effectively across multiple servers and ensures that no single server bears too many requests that lead to degrading the overall performance of the application. A load balancer can be a physical device or a virtualized instance running on specialized hardware or a software process.
Consider a scenario where an application is running on a single server and the client connects to that server directly without load balancing. It will look something like the one below.

Load Balancer

How to handle the unavailability of a Load Balancer?

If the load balancer becomes unavailable, then the corresponding server will become unavailable and the system will go into downtime. In order to handle such cases, we do opt for two techniques:

1. **Way 1:** Using backup load balancer technique: It contains primary and secondary load balancers involving concepts of ‘floating ID’ and ‘Health check’.
2. **Way 2:** Using DNS Server: For now newbies to understand associate this works quite similar to the redundancy principle.

> **Note:** Geeks do remember that DNS does not keep track of whether load balancer is working so do we introduce **Monitor** to keep track in case of DNS.
>
> **Tip:** In system designing failures are inevitable, we could not eliminate them completely can only work in minimizing them.

While considering the availability factor while designing systems is directly proportional to geographic location. For instance: For a corresponding service if a system goes down in a particular location (be India) the whole service is available at some other location to make the service operational. In the real world, we are having the complete hardware available across various locations so as not to hamper service at any cost.

Consistency in System Design

> **Consistency** is referred to as **data uniformity in systems**.

When a user requests data, the system always returns the same data, regardless of the user’s location, time, etc. Before a user receives data from any node, the server at which the data is updated or changed should successfully replicate the new data to all the nodes.

In order to understand consistency in simpler terms:

- Consider three nodes **X, Y, and Z** in a system, where very little **t1** time is taken for data transmission for replication from **X** to **Y** and **t2** from **X** to **Z**.

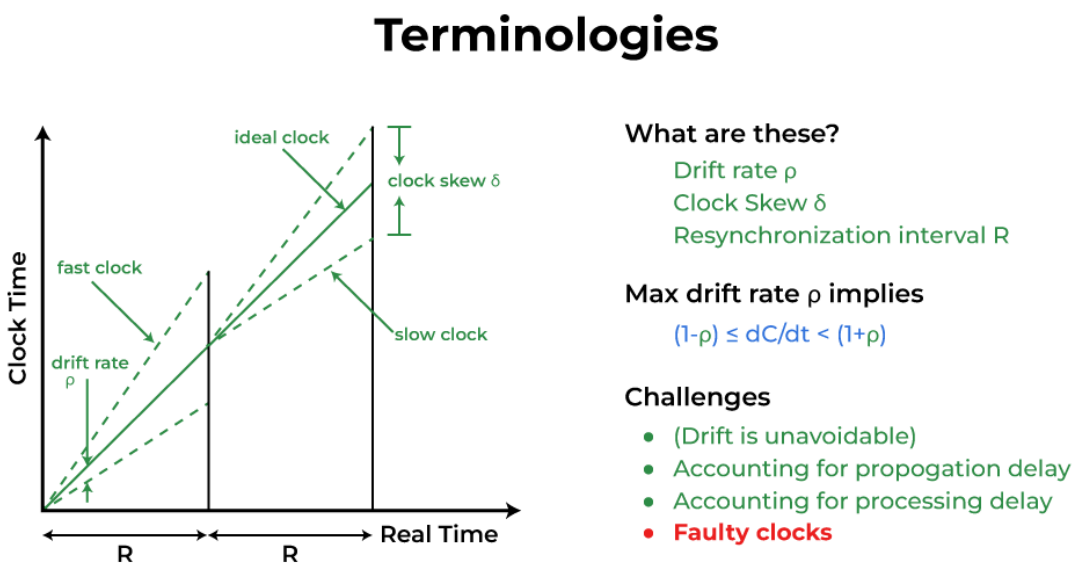
- Now the maximum among **t1** and **t2** is taken where no user is supposed to fetch info so as to avoid to get inconsistency in data by providing older records.

Example: Account Transactions

> **Note:** If the users gets the data somehow because of inconsistency in systems, it is known as **Dirty Reading.**

Time in System Design

> **Time** is a measure of sequences of events happening which is measured here in seconds in its SI unit.
>
> It is measured using a clock which is of two types:
>
> 1. **Physical Clock:** responsible for the time between systems.
> 2. **Logical Clock:** responsible for the time within a system.



Time in System Design

Illustration:

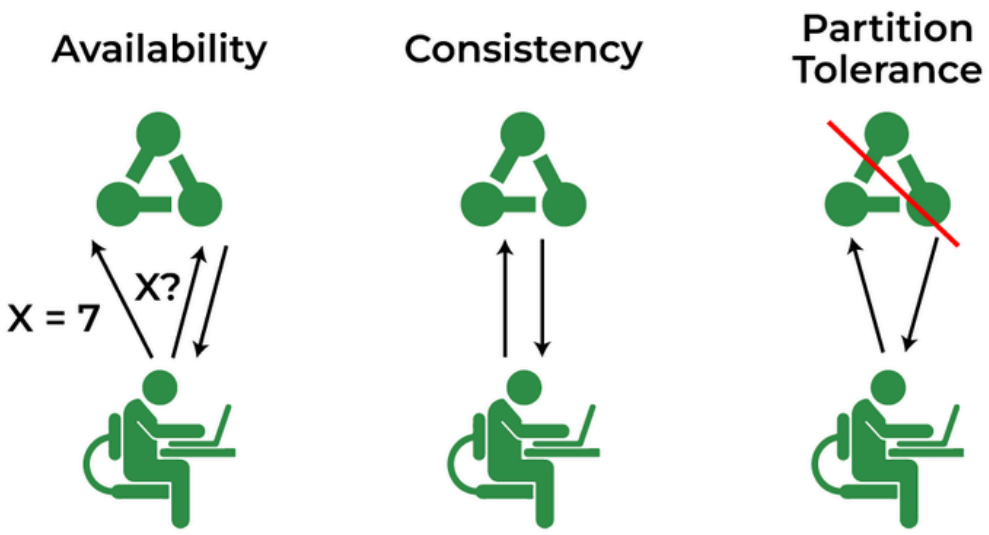
List out all colleges in the world. So here the time taken between colleges for communication is a measure of the physical clock. Whereas the time taken within the hospital is a measure of a logical clock.

CAP Theorem In System Design

Three desirable characteristics of distributed systems with replicated data are referred to as CAP: partition tolerance, availability, and consistency (among replicated copies) (in the face of the nodes in the system being partitioned by a network fault). According to this theorem, in a distributed system with data replication, it is not possible to ensure all three

of the required properties—consistency, availability, and partition tolerance—at the same time. It claims that only two of the three properties stated below can be supported strongly by networked shared-data systems:

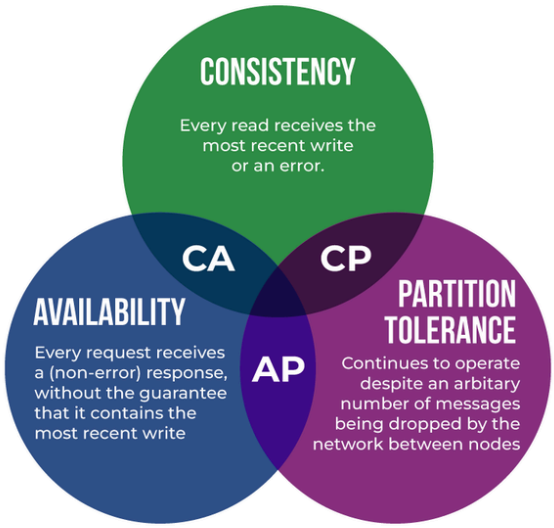
- **C**onsistency
- **A**vailability
- **P**artition Tolerance



CAP Theorem

The different combinations and their use cases are as follows:

1. **CA: Consistency and Availability:**
2. **AP: Availability and Partition Tolerance:**
3. **CP: Consistency and Partition Tolerance:**



CAP Theorem

Lamport’s Logical Clock Theorem

Lamport’s Logical Clock is a process to ascertain the sequence in which events take place. It acts as the foundation for the more complex Vector Clock Algorithm. A logical clock is required because a distributed operating system (Lamport) lacks a global clock.

> **Remember:** Leslie Lamport came up with the idea for Lamport’s Logical Clock.

Tip: Refer here to read more and understand better the Lamport Logical Clock Theorem.