Computer Networks Summary with a Focus on OSI Interconnectivity Model

Part 1: Networking

The OSI model is the Open Systems Interconnection model which describes the architecture that allows for various machines to communicate with each other using standardized protocols. The OSI model can be broken up into 7 abstracted layers, each with one being layered on top of the other, each with their own unique function for universal communication. Each layer can only communicate with the layers it is directly adjacent to, so each communication up or down. The layers from top to bottom are: Application layer, Presentation layer, Session layer, Transport layer, Network layer, Data Link layer, and finally the physical layer.

The application layer is the highest level layer and is used for interaction with the actual software application and the user controlling the machine. However due to this being a network architecture the application layer does not actually include the softwares the end user uses, but the components and protocols that connect the user to the network. These can include things such as HTTP, STMP, FTP, etc. The application layer receives this data from the presentation layer.

The presentation layer's main function is to package or present data taken or given from the application layer. It does so by encrypting, decrypting, compressing, or decompressing the data given to it depending on the layer that needs the data. For example, the presentation layer may receive data from the application layer that is too large to be sent over the network so it would be the presentation layer's job to package the data neatly in the required size before sending it to the layer below it. It may also decode messages sent from the layer below it so that it can be presented on the application layer in a form readable to the end user.

The session layer is the layer below the presentation layer, it creates the communication between two devices such that it will initialize, maintain, and terminate the connection between devices. It is known as the session layer as a "session" is the duration of communication between two machines. Many authentication protocols will also utilize the session layer.

This layer also allows for segmenting of data so in the case a crash occurs, we do not have to download the entire file again and only the missing segments. The transport layer is used to pass these messages from machine to machine.

The transport layer is used for communication of data between machines in the form of connectionless communication such as UDP or connected-oriented communication such as TCP. This data can be in variable-length formats and also be received at variable time lengths. This layer is also responsible for both error control and flow control. Error control means the transport layer ensures that the data it's receiving is not corrupted, for example by the use of checksums and asking for a repeat sending of data if anything is incorrect. Flow control ensures that a machine is not receiving more data than it can manage, for example when establishing a TCP connection the receiving machine will acknowledge the reception of data, this reception is done by the transport layer.

The network layer's function is to actually send the data that was packaged in the transport layer as packets to the network. Packets are a segmented form of data in smaller units that, upon reception, a computer can reassemble in order to have readable, in-order data. This layer also performs routing, which is to find the least cost path to the destination machine.

The data link layer is a node-to-node transfer. This means it has a similar use-case to the network layer however the data link layer provides communication between machines within the same network. Similar to the transport layer, the data link layer provides the functionality for communication and flow control between machines.

Lastly the physical layer is the actual real transmission of data bits such as 1s and 0s through some sort of data transfer device such as a switch. This layer will also establish a signal convention between the two devices to ensure that it receives the correct information, such as a high signal is a 1 and low signal is a 0.

Part 2:

Compute Resource Consolidation Pattern

In a distributed system it often makes sense to logically partition resources into separate computation units, however when it comes to scaling this up we can sometimes find that this will lead to wasted computing power. If the separation is too much, then some units within a distributed system will be idle or underutilized. To solve this problem we can use the Compute Resource Consolidation Pattern.

In this pattern, we take similar resources and combine them into one consolidated computation unit to ensure that we get as much use as possible. By consolidating resources we can increase utilization and decrease overhead and costs. A common approach is to look at resources that are similar in both processing requirements and lifetime, meaning that they will scale similarly.

Compute Partitioning Guidance

We can partition data in a distributed system to improve scalability, performance, security, flexibility, and availability. Partitioning data is essentially when split up data into sections that are accessed and managed separately from each other. This improves scalability by distributing the data across multiple computers so there is no memory limit. This also increases the availability and performance since smaller data sets being spread out makes access more efficient and less prone to complete failures. At the same time, certain partitions can be made harder to access than others, making it more secure. Within partitions there are three main strategies: Horizontal partitioning, Vertical partitioning, and Functional partitioning.

Horizontal partitioning, also known as sharding, is storing tables in different databases while maintaining the same schema, each data partition being referred to as a shard in this case. This helps to reduce the amount of rows in a specific table due to the tables being spread out. The difficulty of partitioning horizontally is in the sharding strategy itself, meaning that we want our databases to be evenly accessed and distributed. For example, all commonly accessed pieces of data being stored in one shard would mean this shard is accessed much more than the others, what we would want to do instead is distribute these commonly accessed pieces of data through the system.

Vertical partitioning is when you break up the columns of a table into several databases, meaning that the same key exists in several databases but the full row is split up.

Functional partitioning is when you separate data based on its function to the distributed system. For example, partitioning data by read-only or write-only would be an example of functional partitioning.