COP 5615 - Distributed Operating Systems Principles
Fall 2022 - Weekly Report 1 (6th September - 8th September)

Manish Kumar
gangula.m@ufl.edu

Lecture on 6th September 2022 -

The definition of distributed systems, which we looked at the start of the lecture on Tuesday , is that a distributed system is a collection of autonomous computer parts that appears to its users as a single coherent system. Following that, we had a discussion about nodes, which are autonomous computer components. There won't be a global clock because each node is autonomous, giving each one a unique sense of time. The overlay network, which we previously mentioned, is what happened next. In an overlay network, each node in the collection only communicates with the nodes that make up its neighbourhood.

Structured and unstructured overlays are the two different types. Then we dove deep into the middleware of distributed systems, where frequently used components and services that do not need to be built by each application and how the middleware helps us to achieve openness, scalability, transparency, and resource sharing. We thoroughly addressed the use of middleware to accomplish these qualities. The topic of putting policies and mechanisms into place was then discussed.

In our discussion of scale in distributed systems, we found that it depends on three factors: size scalability, geographic scalability, and administrative scalability. After discussing various scaling approaches, we began by discussing the issues with each scale. Techniques like shifting calculations to clients, DNS, duplicate file servers and databases, and mirrored web pages were covered.

The last topic we covered was ubiquitous systems, which is defined as an idea in software engineering, hardware engineering, and computer science that defines how computing can exist at any time and in any location. There are four core elements in ubiquitous systems which are listed below.

1. Distribution - Involves transparently available gadgets that are dispersed, networked, and accessible.
2. Interaction with a minimum of user-device interaction
3. Autonomy - A level of self-management where the gadgets operate without human intervention and on their own.
4. The ability for the system as a whole to handle a variety of dynamic actions and interactions

Lecture on 8th September 2022 -

The lecture started with the architectures and architectural styles in Distributed Operating System Implementation and continued to discuss about Layered Architecture. Many components are arranged in layers via layered architecture and with the layer above it, each layer communicates by sending requests and getting responses. Using layered architecture, components are divided into units. It works well for communication. Direct communication between layers is not possible. We also looked at the analogy and connectivity between network layers and os architectures, and packet communication between layers in TCP/IP protocol. The true communication is in its physical form only in its lower stages, as a physical signal is being transmitted to the lower layers. The rest of the layer communication is just kind of an abstraction form of communication.

We continued to discuss TCP/IP protocol and went through server side and client side code for TCP/IP. During this discussion we happened to address the problem of using TCP/IP (issues slipover) and how that's not our case in the project as we use high-level abstraction. We continued to look into 3-layered architecture in Operating Systems principles. They are

1. Application Interface Layer - This layer contains the units/resources that are responsible for communicating with end users or external applications outside the system
2. Processing Layer - Processing Layer refers to the area where data transformations happen. The potential for data transformation depends on the many data kinds, data sources, and ingestion techniques that are used.
3. Data Layer - The protocol layer in charge of data and its transfer is known as the data layer. The data layer offers the functional and procedural means for data transfer across network entities and may also do so in a way that allows mistakes that might happen in the physical layer to be detected and possibly fixed.

We started to learn layering in a different approach with an example of a web application architecture. In a web application or a browser based search engine, we have a user interface that acts as an 'Application-Interface Layer', taking inputs from the user and transferring in right formats to the next layer i.e the Processing Layer. The processing layer in browser based scenario is a keyword to query generator in forward flow and a ranking algorithm as well as a html/ui generator that helps in rendering the result in browser in the reverse flow. The generated html is then transferred into the interface layer to be presented to the end user as a response.

After checking the layered architecture, we moved on to the '**Object Based Architecture**'. Instead of seeing a system as a collection of procedures or instructions, object-oriented architecture sees it as a network of interdependent objects. .In a streamlined and ordered way, objects gracefully encapsulate the numerous characteristics and tasks. Through clearly specified interfaces, objects are able to interact, communicate, and work together. As a result, the object-oriented architectural style has supplanted other styles in the development of object-oriented software programs.

We also happened to discuss in short about the Representational State Transfer (REST) architecture, also known as RESTful architecture, in which we have four different constraints to be satisfied to create a uniform architecture. They are -

1. Resource Identification - All resources must be identified with a unique naming convention
2. Resource Manipulation - To make sure that the same interfaces are used for all the services
3. Self-Descriptive Messages - Messages that are sent to or received from a service should be self descriptive in nature
4. Post execution of service, the details of the caller are removed from the component

In RESTful, we use different methods such as PUT(create), POST(modify), DELETE(remove),GET(obtain) etc; to interact with data sent or received from applications.