1. End-to-end arguments in systems design

Delivery Guarantees

The point that stood the most out from the paper to us was the chapter on Delivery Guarantees.

A system must send some sort of an acknowledgement after receiving a message. However, a general message saying “I got the message” is not particularly useful to the sender. The paper talks about messages that tell the sender that the receiver has done something with the message. We agree with the fact that these sorts of messages are much more useful than the generic acknowledge messages.

It is important for systems to receive some sort of an acknowledgement after sending a message to a secondary system. There is a purpose for a middleware on the receivers end that takes over the responsibility of sending that acknowledgement to the sender, making sure that the sender always receives something that tells them that the message has been received. Due to common misconceptions made on networks, i.e., reliability, there is a need for a procedure that makes sure that the file transfer has been successful, and both the receiver and the sender know of it.

By using a middleware to solve this problem we can eliminate most of the difficulties that come up in file transfer protocols. However, by placing the service that provides the delivery guarantees (middleware) on top of the file transfer application, forces other applications using the shared communication system to use its own middleware for reliability enhancement.

If the delivery guarantee system is placed closer to the top of the pipeline, i.e., in the communication system, the other applications do not need to worry about providing their own implementation of the delivery guarantee. Now, the receiving receiver is not using a middleware for guarantee of delivery and the responsibility for letting the sender know that the file/data has been delivered is on the communication system.

In the end, it is up to the designer of the system, if a middleware should be used in the application system or not. There are pros and cons to each implementation, placing the delivery guarantee into the communication system may be redundant for applications that find the cost of this enhancement not to be worth it, but it is not up for them to decide on this. However, with many applications using the communication system heavily, it would benefit most, if not all, of the applications to have the delivery guarantee on the communication system.

2. A note on distributed computing

When using a distributed system, the user might want to access a certain resource without mentioning the location of that specific resource. This sounds great in practice but in reality, this is something that is easier said than done.

Due to many difficulties regarding distributed systems (latency, memory access, partial failure, and concurrency), it can be extremely difficult to program an interface where remote calls act the same way as local calls. There are only two ways to do this, either by programming the system as if all calls, both local and remote, act as if they are remote. However, this causes major problems when detecting errors withing the system. A component of a distributed system might fail, causing another component (which is using the first component) to stop running since the control is never returned to the caller. This is a problem that never occurs with local computing and handling these errors with distributed computing can be extremely cumbersome.

To truly make distributed systems act as local systems, there needs to be a layer that allows for replacement of pointers to objects with object references. This forces the programmers of the system to use a unified model for object interaction.

However, to force the programmers to use the model, the programmer cannot use address space relative pointers. This causes the programmers of the system to adapt a new way of programming which gives up the whole point of location transparency in local and distributed systems.

Attached to this assignment is an example in Go that shows how cumbersome this is in the real world and not worthwhile for programmers and enterprises to achieve.