CMPSCI 677 Operating Systems **Spring 2013**

Lecture 11: February 27

Lecturer: Prashant Shenoy *Scribe: Timothy Wang*

11.1 Review

Streaming is very different from message-oriented communication. After the user sends a request to the server, it continues to push data to the user. In streaming, you tend not to use protocol like TCP, because retransmission can cause the data to arrive late, and late data is as bad as no data.

11.2 HTTP streaming

We mentioned that using UDP as transmission protocol might be inherently a better idea. Yet, almost all streaming on the internet goes through http, which runs through TCP. Having said that, we will introduce how streaming actually works today. Some UDP servers are particularly developed, but none of them actually took off. Nowadays, you can simply send a http request for a video file. Most browsers today wait for the whole file to finish downloading before playing, so http partitions a file into smaller files. Your player request the first file, and plays it when requesting for the next. Http streaming can deal with heterogeneous clients with different bandwidth. The server provides three versions of the same file. The client-side video player can request whatever quality of video it wants. This is referred to direct adaptive streaming over http, or DASH streaming. As the file is downloading, the player can keep statistics of the download process, and request different versions of the next file according to streaming rate. The user can witness the fluctuation of quality if the network is unstable. Note that this is all through TCP. The reasons that http streaming works is that you have large buffers at the client end, and the network speed is reasonable high now.

11.3 Stream Synchronization

When you are streaming data, the data at least has two components, video and audio. To get good quality, these two components have to be lip-sync, in which the audio is played when the lips of the speaker is moving. This is especially a problem when doing Skype. The camera captures images, and the microphone captures audio. You cannot simply send them to the other end, because they could easily become asynchronized. You have to make sure the frame and sample that were captured at the same time were played at the same time as well. This is referred to as "audio-video synchronization". To implement synchronization, you can time stamp the frame and sample. On the other end, you play the frames and samples with the same time stamps. This is a problem more commonly seen in live video, where video and audio stream are coming independently in the network. Note that video and audio do not stream at the same rate, so synchronization is even more important.

11.4 Naming

The main issue here is how to figure out where are resources that you try to access in a distributed system. We assume that every resource has a name, through which we find the resources. Every name is registered to a directory service, and the directory service resolves the name for you, telling you the location of the resource. In distributed systems, the directory service might be distributed, because a single one could not handle the load in a very large distributed system.

11.4.1 Approaches

- **Hierarchical approach:**
  We will explain more of this in later sections.

- **P2P approach:** you can construct the directory system where the name is the key to look up, and the value is whatever the name resolves to. When you want to look up, just inject a request into the P2P system.

11.4.2 How to name objects

File name are constructed using a hierarchical structure. The name of the file must have the whole file directory prefixed to it, e.g. home/steen/mbox. The name with the directory is called a fully qualified file name. This is a way that conventional OSs name files. If you present a fully qualified name to the OS, it has to resolve it. The OS parses the name and looks downwards each layer of the tree structure for the correct file name.

11.4.3 Resolving File Names across Machines

The name resolution process introduced above can work across machines. Here we look at NFS, or network file system. When a client tries to resolve a remote file name, it sends the file name to the server to resolve it.

11.5 Name Space Distribution

In a large distributed system, the name server itself is assumed to be hierarchical. There are three layers:

- **Global layer:** contains top level domains including countries or .edu, .com. This layer rarely changes.

- **Administrational layer:**his layer is controlled by the organization that owns the machine. Example: .ibm,.umass. They can have subdomains under these names, such as .cs, .eng. This layer changes more frequently than nodes in global layers, but still not very frequent.

- **Managerial layer:**This layer contains the leaves of this structure, which are end point machines. Nodes in this layer change a lot.

11.5.1 DNS Name Space

DNS is a form of directory look up service.You can create different kinds of records in DNS service. The following four are the most important ones:

- **A record:** contains the IP address of the host this node represents
- **CName record:** you can create a alias for a machine. One reason to create aliases is to share the load. DNS returns the list of machines with the same alias in a scrambled order, and the client picks the first one on the list.
- **NS record** the name server for this organization
- **MX record** Refers to a mail server to handle mail addressed to this node. Any mail coming to this organization should come to this node

11.5.2 Name resolution methods

- **Iterative Resolution:** This resolution technique iteratively goes to each level of the name space layers to request the IP of that name. Each level requires a round-trip-time. Once you resolve some of the names, DNS allows you to cache some of the name. Next time you won't have to go through the whole process. How long you cache a name depends on what level of name server are you caching. You cache high level name servers longer, because they are unlikely to change. pros:You are able to cache intermediate servers for future use. cons:Must tolerate higher round-trip-time
- **Recursive Resolution:** Simply send the request to the root name server. It figures out what is the next level server to pick, and forwards the request. The next-level server does the same thing until the request the lowest-level server. Note that the highest-level server locations are stored in the local name server. These IP of the high-level servers were preconfigured when local DNS servers were set up. The root name servers are well-known, and also heavily-loaded. **pros:**If all name servers are far away, the round-trip-time for recursive resolution is lower **cons:**Clinet cannot cache the intermediate servers because the client does not involve resolutions. Every recursive iteration goes to the root, so the root server must have high performance.

11.6 X.500 Directory Service

DNS is simply a name server system. It is a key-value look up, in which you specify a key and gets a value as reply. However, there are actually more general-purposed of lookup system. For example, you don't look for a specific person when you need a plumber. LDAP is a variant of X.500. It is widely used in most OSs. OpenLDAP in Unix and ADS in Windows are both examples of LDAP. You can use them to do a variety of lookups. You can use them for user names and passwords. You can also use them to store the list of printers in an apartment. The name space in LDAP is also hierarchical.