CMPSCI 677 Operating Systems **Spring 2019**

Lecture 11: March 4

Lecturer: Prashant Shenoy *Scribe:* ***Akanksha Atrey***

Announcements

- Calendar is updated on the course webpage. The order of lectures had to be changed slightly.
- If you have signed up to be a scribe, you need to make sure which date the lecture you signed up for is going to happen.
- No class on Wednesday.

11.1 Stream Synchronization (Lecture 10 Cont'd)

Stream synchronization arises in multiple contexts where you have two or more streams, for instance an audio and video stream in video calls. When the output gets played out on the other end, the two streams need to synchronized. This can be done in many different ways and time stamping is the simplest method to do so. When two streams are captured on the same device (i.e. both streams share a common clock), on the receiver end, captured time stamp can be used to map both streams. If the two streams are captured by difference devices, stream synchronization will be a problem if the clocks are not synchronized.

11.2 Naming/Distributed Naming

Naming is a mechanism that allows you to map a name of an object to some value of the object. Essentially all resources in any distributed system or application are given names. Other entities in your system can access these resources using names. Thus, there must be something that can map the resource to the underlying entity of the resource. To do this, you are going to need a naming service.

For instance, if you have used RPCs or RMIs in Lab 1, you used a naming service to map the server to its IP address and the port number. The server has to have a name associated with it and when a client needs to bind to the server, it has to lookup the server's name. This allows you to specify an addressed server by name, and map it to, in this case, an IP address and port number that the server is listening on.

If you have a large distributed system, the number of names in your distributed system will be large. So you are not going to have one name server that will handle all the naming requests. The naming system itself has to be distributed in nature. We will use Domain Name System (DNS) as a distributed naming system example to see how to construct them. We will use machine name to IP address mapping as an example to see what kind of name has to be resolved. What you have to keep in mind is that you can have an arbitrary name in any distributed system.

So what happens if you type www.cnn.com in your browser? Your browser will do name/url lookup in DNS. DNS will then map that name to the IP address of the machine, which will then be used to create a HTTP

connection to that machine. Note, the user does not need to know anything other than the name of the server which, in this case, is the URL. This process is known as name resolution.

11.2.1 Naming Approaches

There are two approaches for distributed naming. When you have a lot of entities performing naming lookup, your distributed system is going to be distributed in nature. The way you are going to construct the distributed naming system is using one of two mechanisms. The first mechanism uses a distributed hash table (DHT) and a P2P-like (key-value) name lookup to resolves names. You can write your naming service on top of a P2P system of this sort where key is the name and value is whatever the name needs to resolve to. The other mechanism is hierarchical in nature where the naming system is broken down into hierarchies.

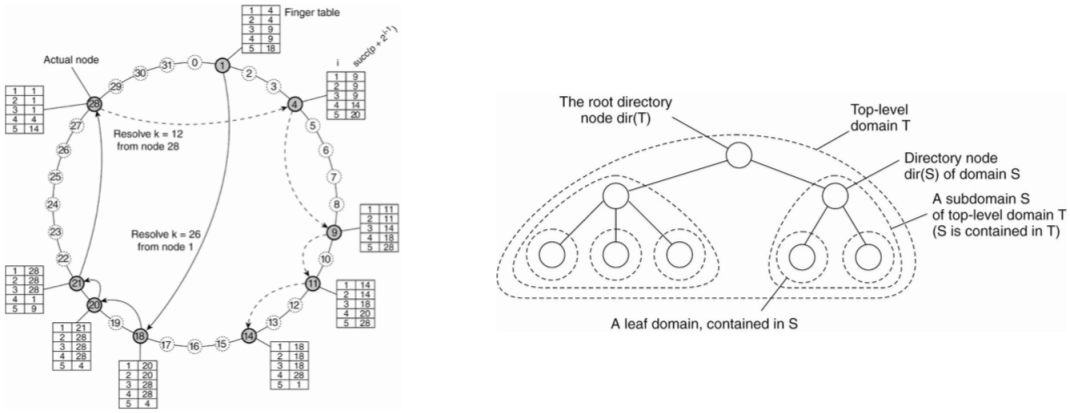

Figure 11.1: P2P vs Hierarchical Naming System.

Question: Traditionally, for ring topologies, does the traffic have to go one way or both ways?

Answer: The problem how the traffic flows through the topology is called query routing. That is, if you make a request, how does that query route through the network. There are many routing schemes. For example, if you say your routing is clockwise, that may not be optimal if the node you want is one hop away in the other direction. Typically, you might have some intelligence to determine which route is shorter and send the query along the shorter path. The query routing protocol that is layered on top of this network and the goal in a query routing protocol is to optimize the resources used.

Lets take file name as a simple example and see how name resolution works for file names. When you are using a file system on your machine, your OS is performing name resolution. Internally, files are numbered (numeric IDs) and humans also give textual names to files. When you open an application and access a file, your OS performs name resolution by taking the name of the file, getting the numeric ID associated with the file and using that to access the file.

The file systems name space is typically hierarchical. Thus, the name resolution is going to involve multiple lookups if you have a directory structure. For example, if you are going to find */home/steen/mbox*, you will first look at the root directory / and find the subdirectory *home*. Then you will open the subdirectory *home* and continue descending to the next level of the subdirectory. Thus, you traverse the hierarchy based on the name and once you hit the file, you open the metadata and get the numeric ID of the file.

The same concept is used to do name resolution in other systems. The name server has hierarchical components and you are going to look at each of the components to try to find the resource the user wants or an

application is trying to address.

11.2.2 Resolving File Names Across Machines

In a distributed file system, some of the files are stored on the server machine instead of all being on the client machine. So, name distribution has to become distributed.

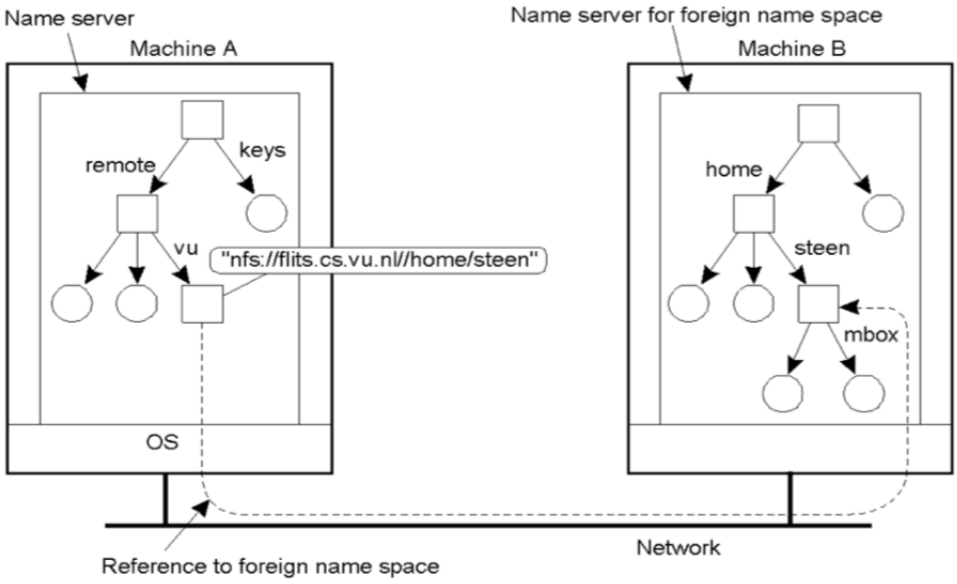

Figure 11.2: Resolving File Names Across Machines.

Lets take an example where you have machine A (which is the client) and machine B (which is the server). You have taken some file stored on the directory from the file server and you have mapped it on to the client using a mount protocol. Mount protocol simply establishes a mapping of a local directory to a remote directory. So when you open the local directory, you are actually seeing files in the remote directory through the newly constructed mount point. An example of a mount protocol is NFS. To do name resolution in this system, you will traverse down the root hierarchy as you would normally until you hit the mount point. Once this happens, the OS knows that this resolution has to happen on a server instead of locally. Thus, multiple machines are participating in this case. Part of the lookup is happening locally and once you figure out the file is remote, the request is sent to the remote machine. What we have generalized from the previous case is part of the name resolution is done at the client. Name resolution in this case is with multiple entities and thus, distributed.

11.2.3 Name Space Distribution

Naming in large distributed systems may be global in scope. If you look at the Internet, the naming system actually has millions of nodes. The naming system has to serve potentially millions of devices in a very large distributed systems. We use the Domain Name System (DNS) to map machine names to IP address in the Internet. DNS is distributed through three logical layers:

1. Global layer, highest level nodes and the root of hierarchy; mostly static.

2. Administrational layer: the organization that are part of each of the domain.
3. Managerial layer: lowest layer; are actual nodes where there are frequent changes.

As you go down in the naming hierarchy, there are more frequent changes to the naming system. Top level domains almost never go but more can enter. The image below shows an example of name space distribution.

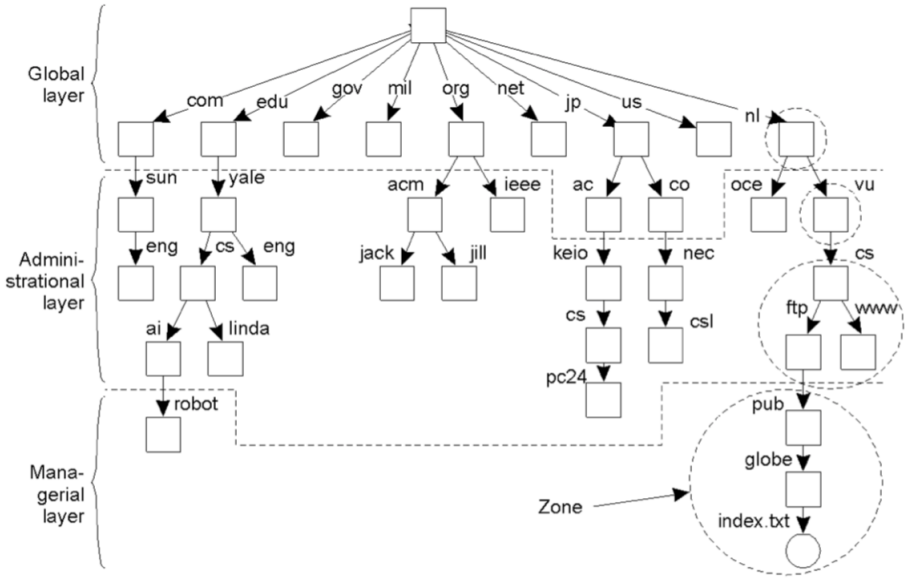

Figure 11.3: Example of Name Space Distribution.

Question: The naming service is distributed, so why is there a head node at the top in the image?

Answer: What this is showing is the name space, not the servers in DNS. Name space is how the names are constructed and organized. This does not necessarily mean the servers are organized this way. But in many cases, the name space also gives us a hierarchical organization of the servers.

The below chart illustrates some of the requirement of some of the domains:

Item	Global	Administrational	Managerial
Geographical scale of network	Worldwide	Organization	Department
Total number of nodes	Few	Many	Vast numbers
Responsiveness to lookups	Seconds	Milliseconds	Immediate
Update propagation	Lazy	Immediate	Immediate
Number of replicas	Many	None or few	None
Is client-side caching applied?	Yes	Yes	Sometimes

Figure 11.4: Requirements of Some of the Domains.

The more stable a layer, the longer the lookups are valid (and can be cached longer). As you go downs the hierarchy, you need a faster lookup. If you make changes to the domain names, it will typically take 2-24 hours for the changes to be visible.

11.2.4 How does DNS work?

DNS can be seen as a large distributed database that does key-value lookups. The key is machine name and the value is IP address. The servers that are part of the DNS will have several types of records (databases of records). The table below shows different possible records. The most common record is A record which takes in a host name and maps it to an IP address. MX record refers to a mail server and is responsible for the email service for that machine. NS record is the IP address of the name server for that domain. CNAME is an alias record which allows you to give multiple names to a machine. Just as files can have multiple names, a machine can have multiple names as well. You can use *nslookup* to look at that.

Type of record	Associated entity	Description
SOA	Zone	Holds information on the represented zone
A	Host	Contains an IP address of the host this node represents
MX	Domain	Refers to a mail server to handle mail addressed to this node
SRV	Domain	Refers to a server handling a specific service
NS	Zone	Refers to a name server that implements the represented zone
CNAME	Node	Symbolic link with the primary name of the represented node
PTR	Host	Contains the canonical name of a host
HINFO	Host	Holds information on the host this node represents
TXT	Any kind	Contains any entity-specific information considered useful

Figure 11.5: Different Types of Records.

11.2.5 Name Resolution in DNS

There are two common ways to do name resolution in DNS: iteratively and recursively. In iterative name resolution, you are given a domain name and you split it into pieces from the root domain all the way to the machine name. Then, you iteratively go on and look for domain name servers of each of these levels of hierarchy. Once you reach the final machine name, you will get the IP address. Essentially, you iteratively look for the name by descending down the hierarchies, similar to what we did in the file system case.

In recursive name resolution, you just give the name to the root domain server and it figures out what the next name server is by forwarding that request down the tree. The request will traverse down recursively to the domain naming server at each level until you find the machine and then the response is returned up the tree to the request server. Here, the name servers are propagating the request themselves instead of returning answers one at a time and the client machine talking to the next level name server.

There are pros and cons of both types. In the recursive method, the client makes only one request and gets only one response. This reduces the round time request. In the iterative approach, the round trip time for each of the request is going to be large, especially when you consider domain names that are geographically far.

However, the recursive method will cause heavy load on the root name server, especially when we think about there being millions of naming requests. This is not an issue in the iterative approach because you can use caching to reduce the load at higher level name servers. You can cache the previous requests that have been resolved at your local machine on the local name server. Moreover, you will be able to cache the replies for high level name servers for longer because they don't change as often.

So, if your client and name servers are far, the recursive way should be better from a performance standpoint. From the infrastructure standpoint, in the recursive case, the root node is much more heavily loaded in the hierarchy. So, the machine should be very powerful to handle very large amounts of requests. For this reason, iterative name resolution is more common for scalability purposes.

Question: How does the client know where to look up for the root server?

Answer: The local DNS resolver is configured to know the root levels.

Question: Is it possible to cache in recursive name resolution?

Answer: Yes, if the user wants the exact same name, you don't have to send the request to the name server. However, this reduces load lower in the hierarchy but that is not where the problem is.

Question: Can you replicate?

Answer: Yes, you can replicate records. But you have to then maintain consistency and DNS uses a very loose form of consistency because they exchange information about once a day. So that is an issue with replication in this case.

Some advanced DNS features are:

1. Machines can have multiple names through CNAME and you can register them through DNS.
2. Same machine has two A records (star.cs.vu.nl) with two IP address. Such a thing can be used to cluster webservers. Browsers can connect to either of the two and either IP address will give you the service you want. This can be used to achieve load balance.
3. DNS can see who is making the DNS request and returning different responses based on where the nearest server is geographically. This is called geographic load balancing. So, the response time will be lower.

Question: What are the two parameters of MX records?

Answer: This is to set priority levels on what you want to access first. If one fails, it goes down the priority list and tries accessing that.

Question: Are there any security implications of registering an IP address to an arbitrary domain?

Answer: Typically, you have to be authorized to make changes to the DNS server. So, the mechanism will not allow you to change that.

Question: You have two A records for a machine and you make one the alias of the other, what would happen?

Answer: Typically, that would be an error because you cannot have both types of records for the same machine.

11.2.6 X.500 Directory Service

DNS provides pure key-value lookup. You specify machine names and get the corresponding IP address. You cannot do any advanced lookups like general queries. For instance, you cannot ask show me all entities that match x attribute and expect to get a list of those entities.

X.500 allows you do more generalized lookup. You can generally ask for any matching attribute and then it will show you all the entities with that attribute. X.500 is the standard that was developed to perform a

more advanced directory services. For this reason, you can also use X.500 for different tasks such as looking up plumbers in yellow pages or getting all users with a certain last name.

Question: How does X.500 do secondary indexing (lookup)?

Answer: This is an implementation specific detail. They specify interfaces and the implementation is left to the designer.

However, X.500 was a commerical failure.

11.2.7 LDAP

What is now used more widely is Lightweight Directory Access Protocol (LDAP). It is a more specialized and simplified form of X.500. Essentially the underlying concept is the same but they got rid of some complexities that X.500 had. For example, X.500 could run on any network protocol. LDAP on the other hand only runs on TCP/IP. LDAP also allows only one type of encoding – OS encoding, whereas X.500 allowed arbitrary types of encoding of databases.

The LDAP name space is also hierarchical. It typically starts with the geographic domain. Moreover, LDAP is used at an organizational level. Unlike DNS servers, all the LDAP servers are not connected together. LDAP servers within an organization can talk to each other but LDAP servers across organizations do not need to. For this reason, this is widely used in most organizations. For instance, UMass people finder uses the LDAP service.

11.2.8 Naming with Mobile Nodes

How does naming work when nodes are mobile? Today most nodes are in the Internet are phones. This means nodes can move from location to the other, resulting in different IP addresses. To address this, the concept of Mobile IP was introduced. Although not widely used, it attaches a home location to each node and if the node moves, it has to register the new IP address with its home server. So when you try to do a lookup or connect to that machine, you first connect to the home server and then get redirected to the IP address of where the node is.

Question: How is this different from adding another layer in the hierarchy?

Answer: The problem is of changing IP addresses and how to connect to nodes when this occurs.

11.3 Clock Synchronization

Next several lectures of the course will be looking at canonical problems in distributed systems. One such problem is of clock synchronization. Overtime clocks drift from real time and this makes it difficult to reason about order of events. For this reason, you want clocks to be synchronized.

For instance, *make* uses the timestamp of the source code file and the timestamp of the last compilation of the file. If it has changed since it was compiled last, it will re-compile. This is used to save computational overhead. If you do this on a single machine, this is not a problem. The problem arises when the build machines are different from development machines.

Question: If you are compile on one machine and build on one machine, is there an overhead of transferring the file?

Answer: Typically, these machines have a shared network so you don't have to transfer files.

11.3.1 Physical Clocks

The most accurate clocks are atomic clocks. Typical clocks, which use crystal oscillators, are not as accurate because factors like temperature can impact how fast/slow the crystal oscillators perform. This results in clock drift. To correct for this, you have to reset your clock by accessing a more authorized clock source periodically. To do this, there are various clock synchronization algorithms which we will cover in subsequent classes.