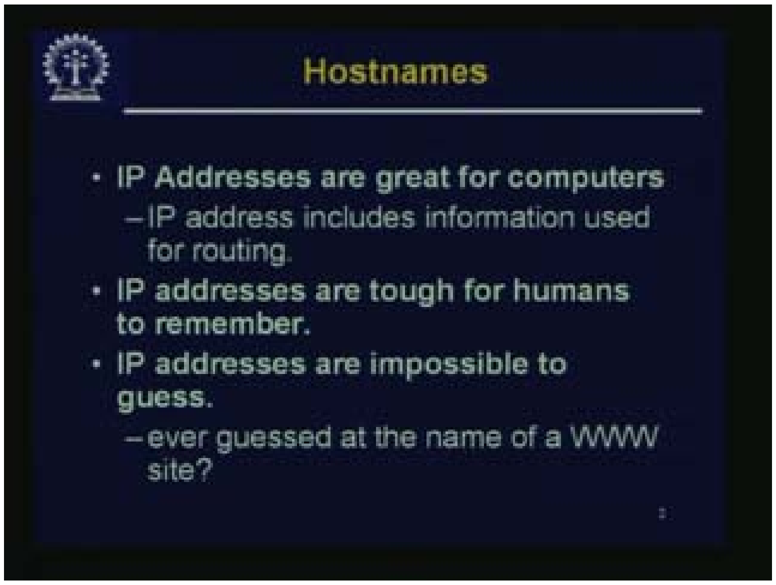
Computer Networks
Prof. S. Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture - 34
DNS & Directory

Good day. Today we will take up two topics, DNS and directory. First let us talk about DNS. DNS is the short form for Domain Name System.

(Refer Slide Time: 00:57-4:19)



Hostnames

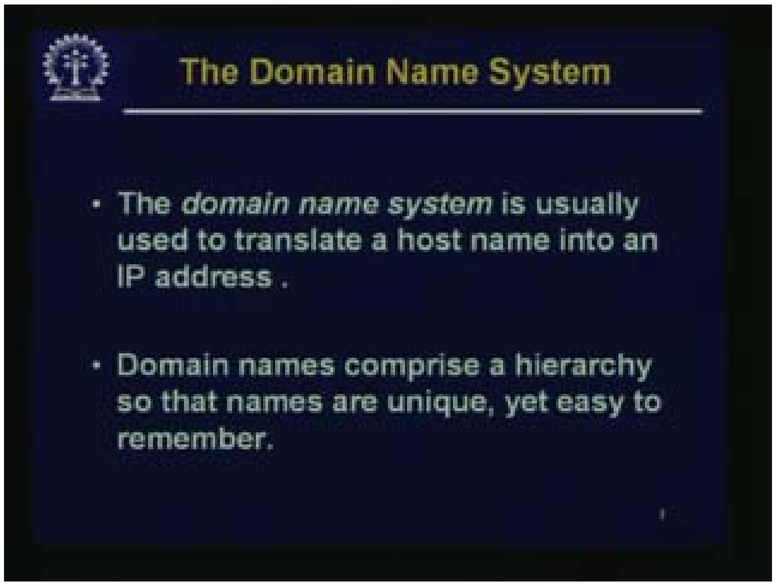
- IP Addresses are great for computers
  - IP address includes information used for routing.
- IP addresses are tough for humans to remember.
- IP addresses are impossible to guess.
  - ever guessed at the name of a WWW site?

We have seen two kinds of addresses. One kind is MAC address or the so called hardware address. In the case of Ethernet it is also called as Ethernet addressed which is used in the data link layer for direct communication. Then we have seen IP addresses used for communication between two end points. The end points can be anywhere in the network. So, IP address includes information used for routing. IP is used for routing where there is a network part and post part etc. But unfortunately, this IP address is tough for humans to remember.

We can remember only a few addresses that we really require for our own configuration, like our own IP address, address of gateway, address of mail server etc. Beyond that it is very difficult for human beings to remember IP addresses of other people. They are impossible to guess. What human beings find is, it much easier to remember and use the domain names. Domain names are used most of time by the people for surfing the web (www site name used for surfing). While surfing the web for a www site, sometimes you may not know the name exactly but you can make it out in 3 or 4 guesses, may be make some combination of .com, .net etc. The most important thing here is that it is much easier for humans to remember site names. The name is used to some particular machine, site etc. This is also some kind of an address and we have the third layer namely the domain names.

We will see how we use these domain names. Just as in the local area network, you require a mapping from IP addresses to MAC addresses done by the ARP protocol and the reverse MAC addresses to IP addresses using RARP. You need a mechanism for mapping the domain names to IP addresses. That is what this whole scheme of DNS is all about. May be sometimes we will look into the reverse query also.

(Refer Slide Time: 4:20-5:58)



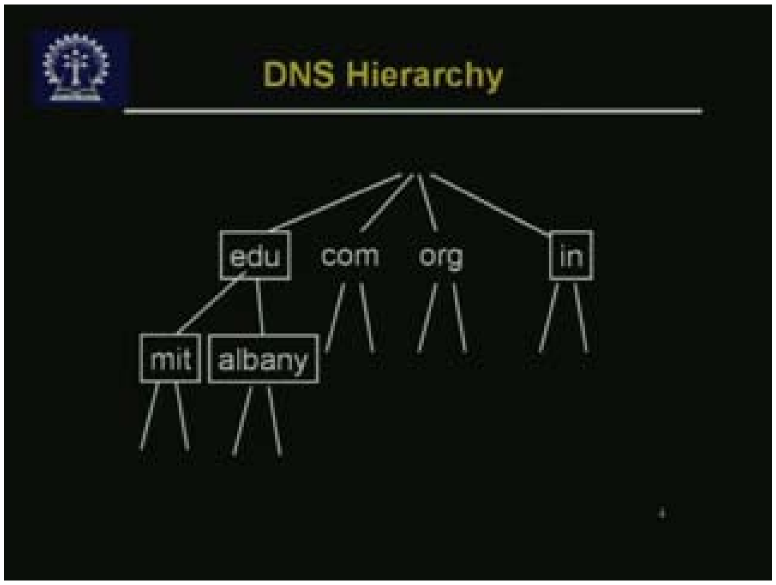
The Domain Name System

- The *domain name system* is usually used to translate a host name into an IP address .
- Domain names comprise a hierarchy so that names are unique, yet easy to remember.

- The domain name system is usually used to translate a host name into an IP address.
- Domain names comprise a hierarchy so that names are unique, yet easy to remember.

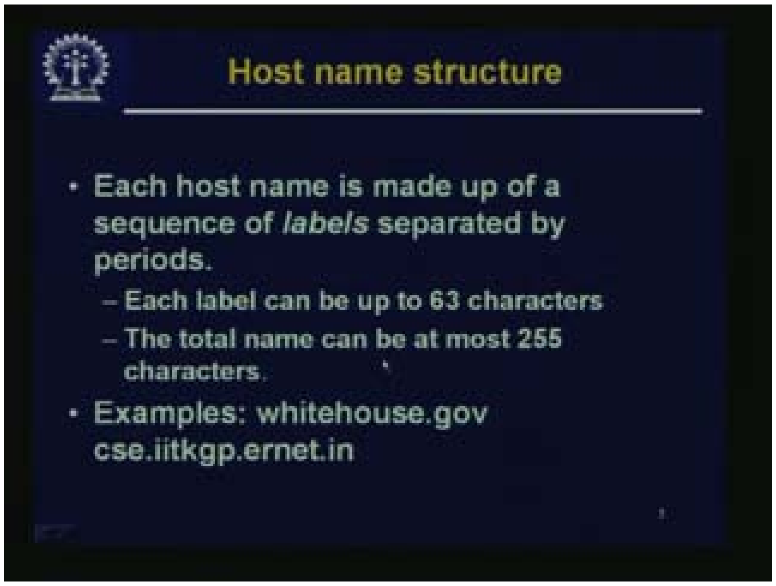
It is important to remember that, if it is an address it needs to be unique. In IP v4 we ran out of IP addresses because we had limited length. Here, we can go easily with the length because these are not to be used in high speed computations but may be used just once or twice in a session. This means you can allow longer names, that is, rather than 4 bytes you can use many bytes but then this also has to be unique. What people thought was, if we make a hierarchy of names which is a logical hierarchy which corresponds with the external world then not only they will be remembered easily but can be easily administered and we can make them unique. So, domain names comprise a hierarchy so that names are unique yet easy to remember.

(Refer Slide Time: 5:59-6:48)


DNS Hierarchy

This is what the hierarchy looks like. There is a root and from the root we have the top level domain. Here we have edu, com, org and so on. And finally these are for some nations. Under edu, we have mit, Albany and all other kinds of organizations. Under in, there may be ernet and all that. Under mit, also there may be cs or something. So there is a tree and a hierarchy. This is sort of a global hierarchy and is the DNS hierarchy.

(Refer Slide Time: 6:49-8:34)


Host name structure

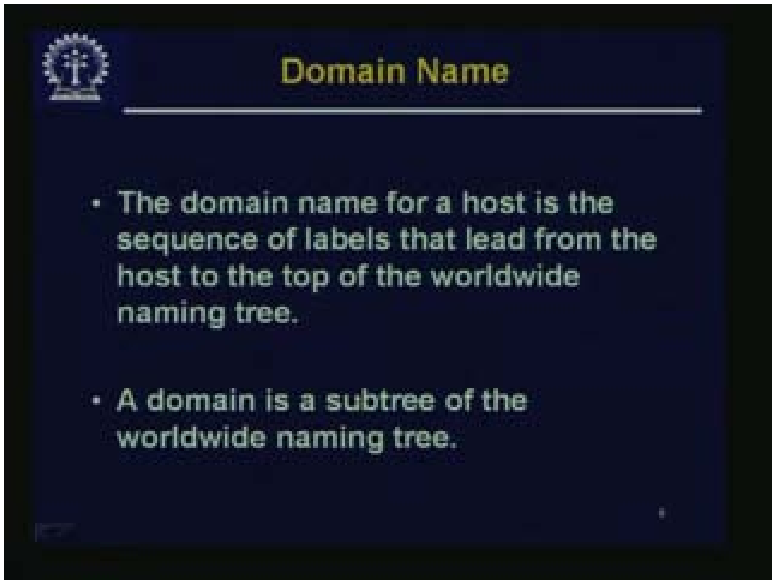
- Each host name is made up of a sequence of *labels* separated by periods.
  - Each label can be up to 63 characters
  - The total name can be at most 255 characters.
- Examples: whitehouse.gov cse.iitkgp.ernet.in

- Each host name is made up of a sequence of labels separated by periods.
o Each label can be up to 63 characters long

o The total name can be at most 255 characters. So, you have up to 255 bytes to code these names so you can go easy on the length

- Examples: Whitehouse.gov, .gov is for government and since it started in USA, it is US government and whitehouse.gov is a domain. Similarly,let us say csc.iitkgp.ernet.in, .in is a top level domain standing for India and ernet is an organization which comes in the next level in in. Under ernet, there is iitkgp which comes in the next level and under iitkgp there is csc which again comes in the next layer. So, starting from here you can go from the leaf of the tree right up to the root of the tree that is the DNS hierarchy tree. And, after giving a label at each level we put a dot and then go up to the next level. This is how we name hosts.

(Refer Slide Time: 8:35-9:04)



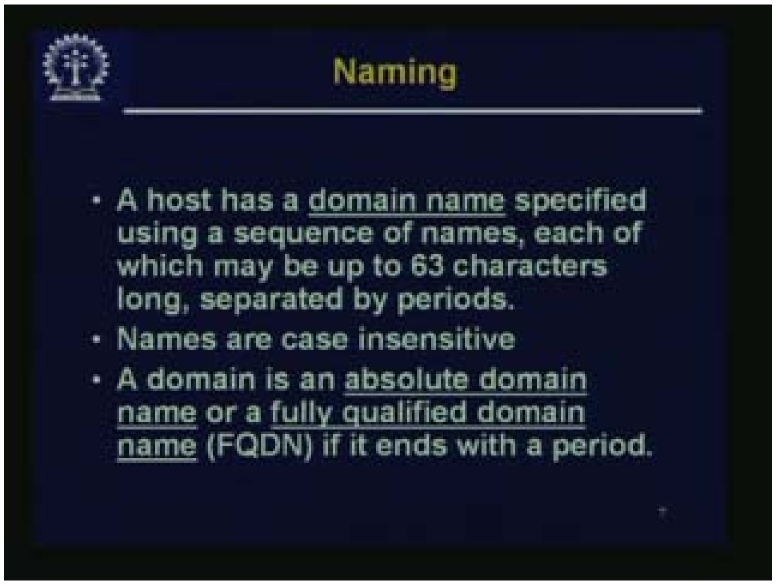
Domain Name

- The domain name for a host is the sequence of labels that lead from the host to the top of the worldwide naming tree.
- A domain is a subtree of the worldwide naming tree.

- The domain name for a host is the sequence of labels that lead from the host to the top of the worldwide naming tree.
- A domain is a subtree of the worldwide naming tree

So, mit is a domain, ernet is a domain and so on. There is a subtree under ernet and there will be many sub domains. So any domain is a subtree of the worldwide naming tree.

(Refer Slide Time: 9:05-9:25)

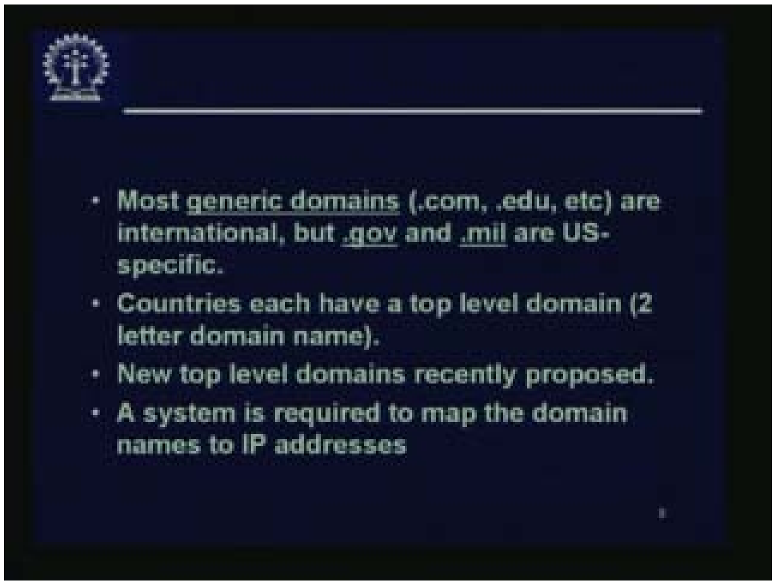


Naming

- A host has a domain name specified using a sequence of names, each of which may be up to 63 characters long, separated by periods.
- Names are case insensitive
- A domain is an absolute domain name or a fully qualified domain name (FQDN) if it ends with a period.

-	A host has a domain name specified using a sequence of names, each of which may be up to 63 characters long, separated by periods.
-	Names are case sensitive.
-	A domain is an absolute domain name or a fully qualified domain name (FQDN), if it ends with a period.

(Refer Slide Time: 9:26-10:41)

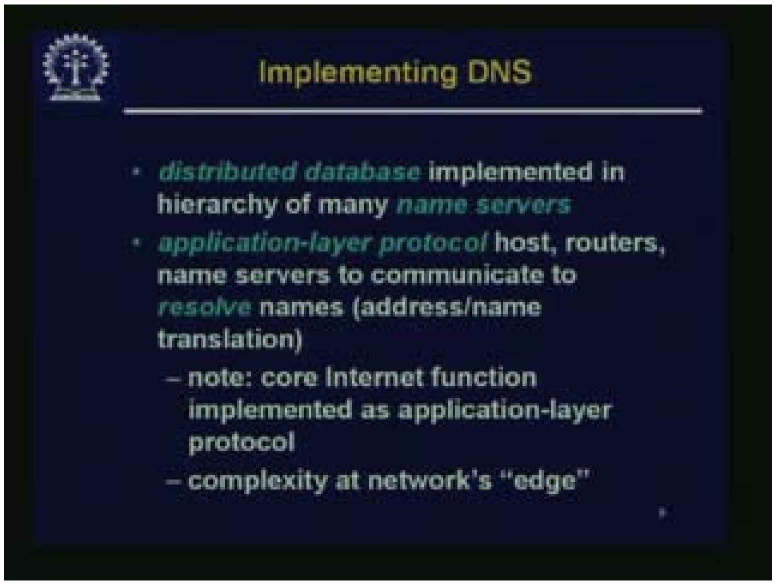


- Most generic domains (.com, .edu, etc) are international, but .gov and .mil are US-specific.
- Countries each have a top level domain (2 letter domain name).
- New top level domains recently proposed.
- A system is required to map the domain names to IP addresses

-	Most generic domains (.com, .edu, etc) are international, but there are some like .gov for government and .mil for military are US-specifications. So when you say .gov, it actually means US government and .mil is for US military. If you remember, the history of internet started in USA from the ARPA net and it evolved. At that time, it was very US centric but now the whole world has embraced it.

- New top level domains recently been proposed, though they are not very popular yet.
- Countries each have a top level domain (2 letter domain name). For example: in for India, jp for Japan, uk for UK and so on.
- A system is required to map the domain names to IP addresses. Just like we have ARP for IP address to MAC or RARP for MAC to IP address, we need a system to map domain name to IP address that is the chief one and may be the reverse also.

(Refer Slide Time: 10:42- 12:45)



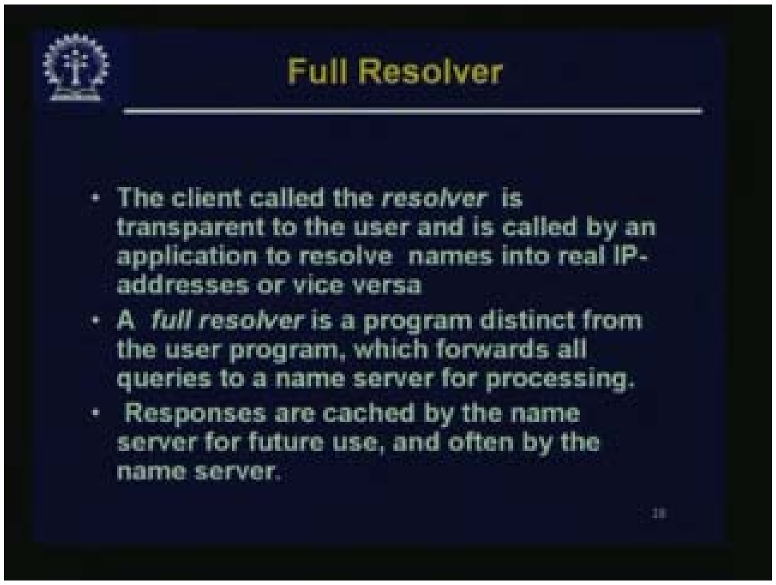
Implementing DNS

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
  - note: core Internet function implemented as application-layer protocol
  - complexity at network's "edge"

Implementing DNS:

- Distributed database implemented in hierarchy of many name servers. This is distributed because this is widely used and you cannot have a centralized database. Centralized database will be much more difficult to administer, Contribute to a single point of failure and the network traffic at that node will be tremendous. You will not be able to handle the network traffic if everybody tries to login to the same central name server. That will not scale and it needs to be distributed. Later on we will study the way it is distributed and the way it is administered. What are distributed are the name servers (servers with gives you the mapping from the domain name system to the IP address).
- There is an application-layer protocol host, routers, name servers, which sort of all combine and communicate to resolve names (provide this address/name translation).
  - o Note: Core internet function is implemented as application-layer protocol
  - o Complexity at network's "edge" so that at the network's core where the traffic is very high and very heavy this is not present. If you had to do the domain name servers at the core routers then it will be of much strain on the routers so they have been put to the edge of the network.

(Refer Slide Time: 12:46-14:29)



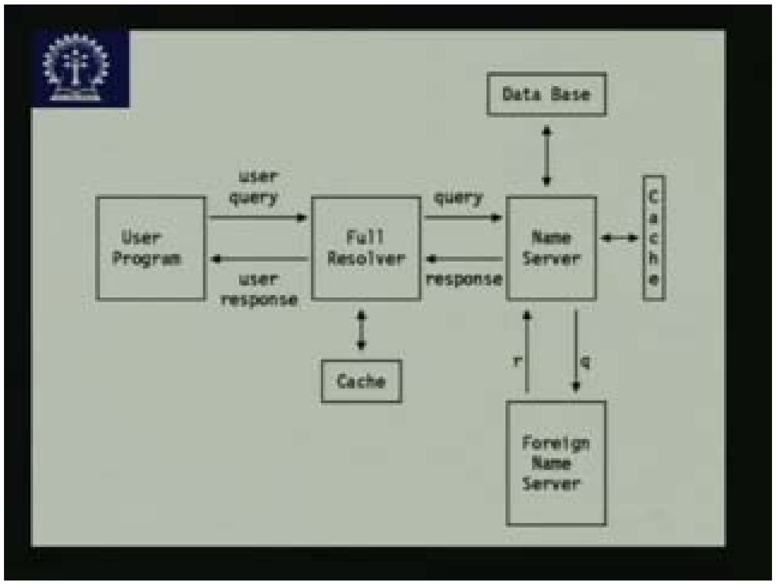
Full Resolver

- The client called the *resolver* is transparent to the user and is called by an application to resolve names into real IP-addresses or vice versa
- A *full resolver* is a program distinct from the user program, which forwards all queries to a name server for processing.
- Responses are cached by the name server for future use, and often by the name server.

Full Resolver: For resolving we use resolvers, and there are full resolvers and stub resolvers.

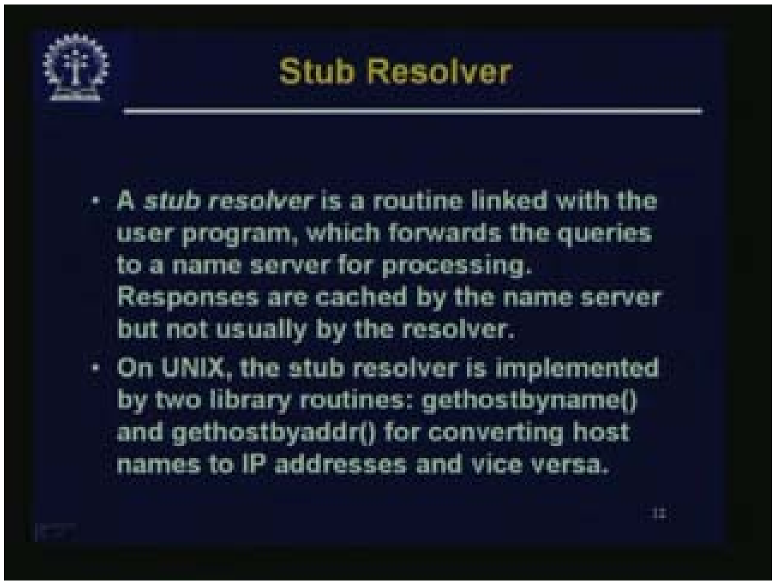
- The client for this naming system is called resolver. This is transparent to the user and is called by an application to resolve names into real IP-addresses or vice versa. When you type a domain name in your browser, the browser is a client application program for http and so its calls the resolver to translate these domain names to corresponding IP addresses. The browser will send the http request to the web server using that IP address. So it gets it from the resolver.
- A full resolver is a program distinct from the user program, which forwards all queries to a name server for processing. It knows about the name server.
- Responses are cached by the name server for future use, and often by the name server. The local full resolver in your host may connect to the local name server. If the local names server does not have the resolution, it is going to contact other name servers and finally will get the resolution. We will look into the details later.

(Refer Slide Time: 14:30-15:19)



In this diagram, the user program gives a user query to the full resolver. The full resolver gives this query to the name server. This name server has its own database, so it may look up in its own database. If it is not there then it may send the request to foreign name servers and finally it will give a response. The full resolver also maintains a cache and it will cache this response so that if it gets another request to resolve the same name then it can find from the cache and give the response to the User. Name server also maintains its own cache.

(Refer Slide Time: 15:20-16:16)



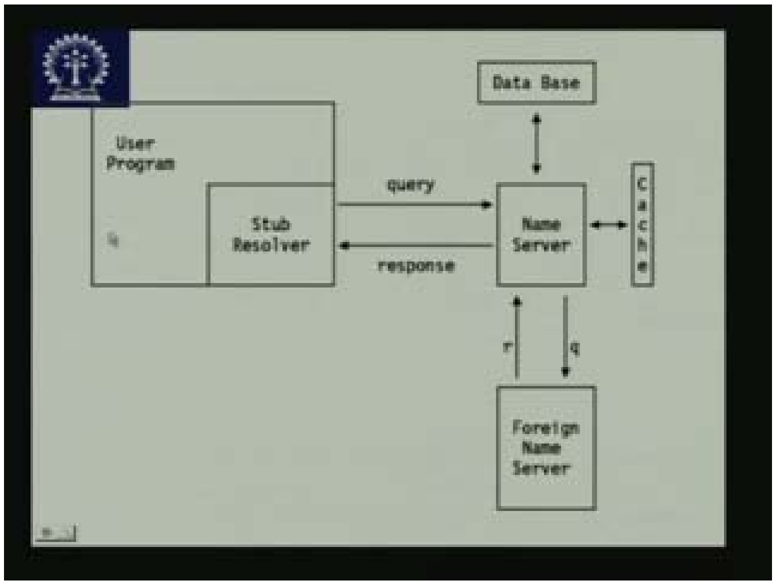
Stub Resolver

- A *stub resolver* is a routine linked with the user program, which forwards the queries to a name server for processing. Responses are cached by the name server but not usually by the resolver.
- On UNIX, the stub resolver is implemented by two library routines: gethostbyname() and gethostbyaddr() for converting host names to IP addresses and vice versa.

12

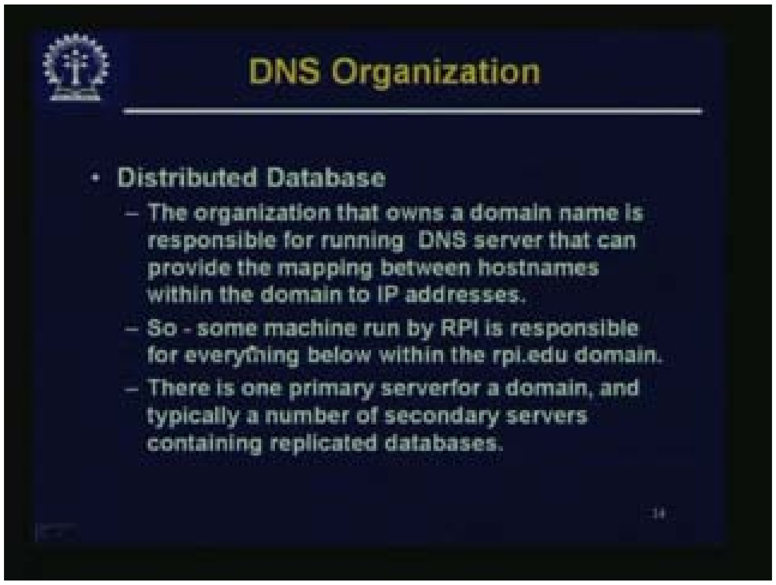
- A stub resolver is a routine linked with the user program which forwards the queries to a name server for processing. Responses are cached by the name server but not usually by the resolver. There are two differences between a stub resolver and a full resolver. One is that, full resolver has a cache but stub resolver usually does not, and secondly, stub resolver has to be linked with the user program where as the full resolver runs by itself.
- On UNIX, the stub resolver is implemented by two library routines: gethostbyname() and gethostbyaddr() for converting host names to IP addresses and vice versa.
-

(Refer Slide Time: 16:17-16:34)



Stub resolver is a part of the user program which gets linked and this routine sends the query to the local name server and gets the response and passes it on. Naturally the user program gets it.

(Refer Slide Time: 16:35-20:04)


DNS Organization

- Distributed Database
  - The organization that owns a domain name is responsible for running DNS server that can provide the mapping between hostnames within the domain to IP addresses.
  - So - some machine run by RPI is responsible for everything below within the rpi.edu domain.
  - There is one primary serverfor a domain, and typically a number of secondary servers containing replicated databases.

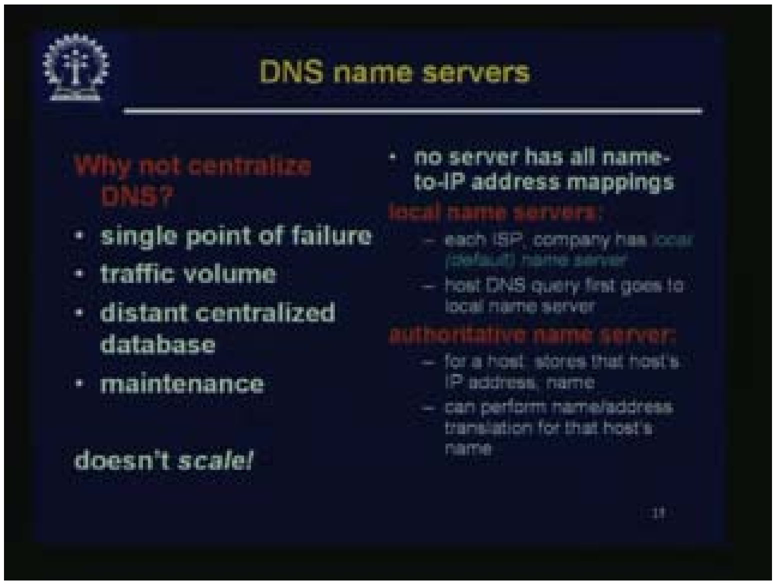
DNS Organization: Is a distributed organization.

- Distributed Database
  - o The organization that owns a domain name is responsible for running DNS server that can provide the mapping between hostnames within the domain to IP addresses.
  - o So some machine run by say an organization like RPI is responsible for everything below within the rpi.edu domain.
  - o There is one primary server for a domain, and typically a number of secondary severs containing replicated databases.

So, it is important to understand how this is organized. This is a global system, which means this is very big. So any effort to control this centrally would become quite difficult. Let us say csc.iitkgp.ernet.in is a domain. Now .in is a domain, which is for whole of India. There is one specific organization (in India) which looks after this domain. So whatever sub domains are there under this domain, it is the responsibility of that organization to keep track of who all can be given sub domains. That is, who all can be given names within these sub domains.in. Also, any name resolution query coming from anywhere else in the world will direct their query first to this organization who is maintaining the domain.in.

Also there are so many organizations maintaining their own sub domains under the domain .in. Either that organization will have it in its cache or it will forward the query to the particular domain. Suppose if it gives sub domain ernet, ernet again gives sub domains to iitkgp and again under iitkgp there may be many sub domains but ernet does not really bother about whatever sub domains are there under iitkgp. It only has to know that iitkgp is an organization under ernet which has got a name and a separate domain. It is for iitkgp to decide how it is going to break this domain into further sub domains or it may not break at all. This way the entire domain administration is decentralized and it is easy, and also these names are some kind of addresses that has been made unique. Since the names are broken into names which stand for actual organization at any level, naturally any particular organization administering a domain will not have two organizations having the same sub domain name under it. This way all the names automatically become distinct which is another good advantage.

(Refer Slide Time: 20:05-20:54)



DNS name servers

Why not centralize DNS?
- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't *scale!*

- no server has all name-to-IP address mappings

local name servers:
- each ISP, company has local (default) name server
- host DNS query first goes to local name server

authoritative name server:
- for a host: stores that host's IP address, name
- can perform name/address translation for that host's name

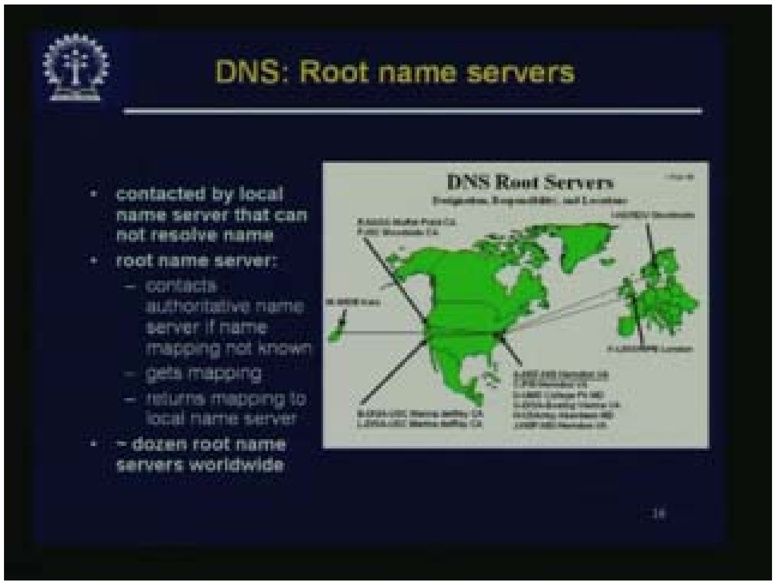
11

Why not centralize DNS?
- Single point of failure.
- Traffic volume.
- Distant centralized database should not work
- Maintenance should be a problem
- Does not scale so new server so its distributed server
- No server has all names to IP address mappings

Local Name Servers:
- Each ISP Company has Local Name Server (Default Name server)
- Host DNS query first goes to the local name server

Authoritative name server:
- Sometimes we may come across something that has been given by authoritative server
- For a host stores that hosts IP address, Name
- Can perform name/address translation for that host's name

(Refer Slide Time: 20:55-21:27)

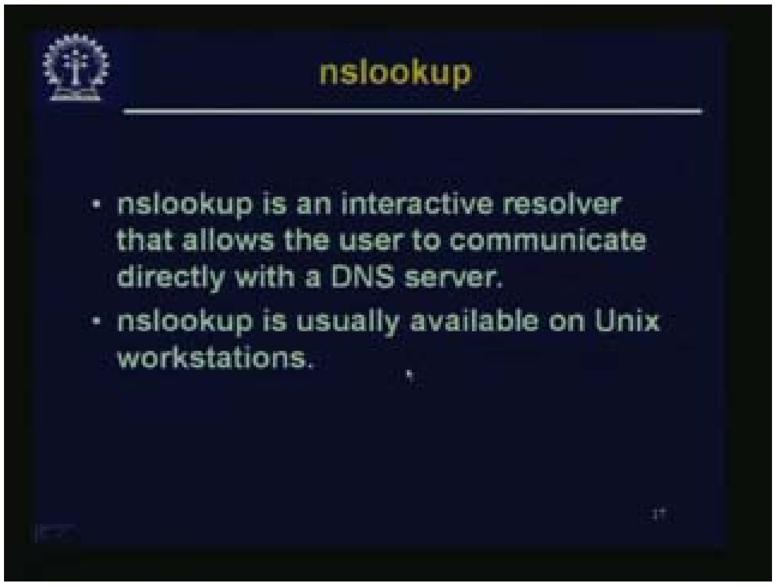

DNS Root Servers

DNS: Root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server
- ~ dozen root name servers worldwide

There are some root name servers. The root name servers are actually still mostly US centric and have some of the biggest name servers. So, root name servers are all in USA but of course that depends on what root it is and therefore they could also be distributed.

(Refer Slide Time: 21:28-21:54)



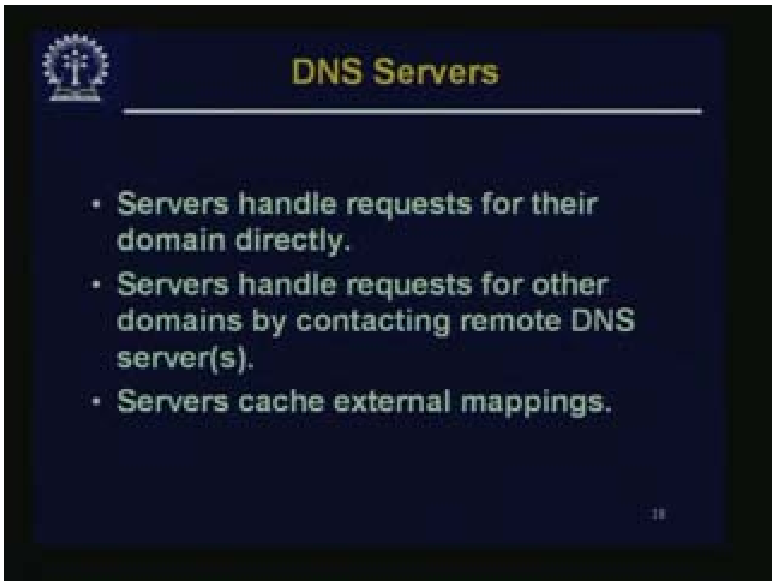
nslookup

- nslookup is an interactive resolver that allows the user to communicate directly with a DNS server.
- nslookup is usually available on Unix workstations.

nslookup:
- nslookup is an interactive resolver that allows the user to communicate directly with a DNS server.
- From the OS, you can use this nslookup and give a name query so this is actually name/server look up. This is why it is called nslookup

- nslookup is usually available on UNIX workstations.

(Refer Slide Time: 21:55-22:12)



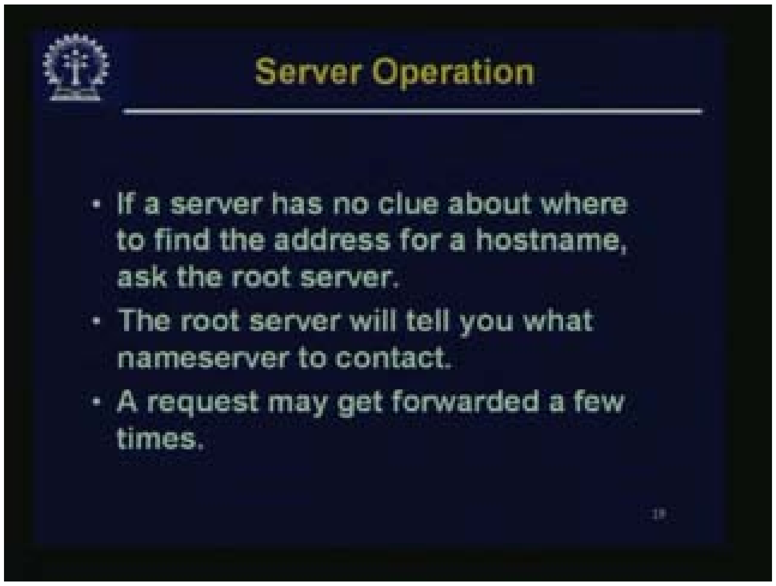
DNS Servers

- Servers handle requests for their domain directly.
- Servers handle requests for other domains by contacting remote DNS server(s).
- Servers cache external mappings.

18

- Servers handle request for their domain directly
- Servers handle request for other domains by contacting remote DNS server(s)
- Servers cache external mappings

(Refer Slide Time: 21:13-23:43)



Server Operation

- If a server has no clue about where to find the address for a hostname, ask the root server.
- The root server will tell you what nameserver to contact.
- A request may get forwarded a few times.

19

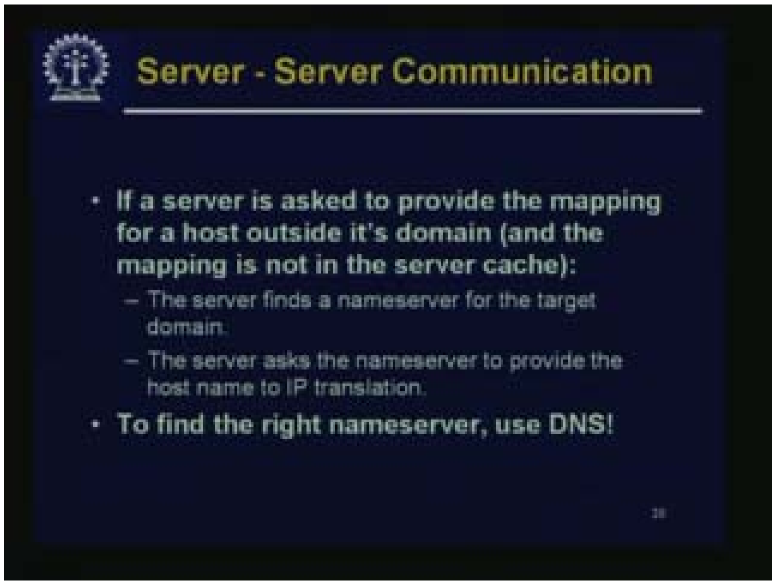
Server Operation:

- If a server has no clue about where to find the address for a hostname, it asks the root server

- The root server will tell you what name server to contact
- A request may get forwarded a few times

Let us say the iitkgp has a name server. A request for a particular domain name translation has come to it. It does not know to whom to connect the name server.. So it can always transfer it to the next higher level, namely, ernet. If the ernet also does not know where in .in it is there, then it can contact in. in will definitely know all the sub domains under it. It has to know because it is administering that domain. If it is for some address which is from outside you can send it directly to the root of that particular domain. Suppose from India you trying to contact something for Japan, you can send it the jp root name server and then jp root name server would know which name server to contact. DNS queries will go back and forth few times and finally the name will be resolved.

(Refer Slide Time: 23:44-24:03)



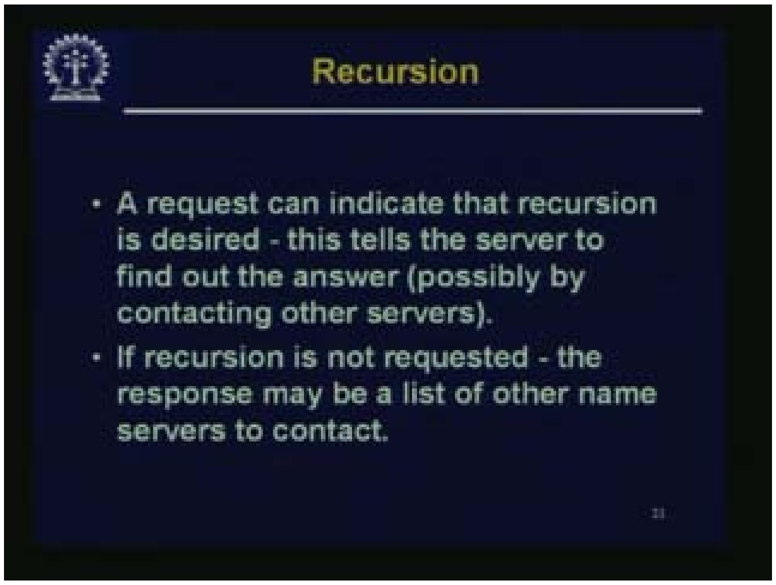
Server - Server Communication

- If a server is asked to provide the mapping for a host outside it's domain (and the mapping is not in the server cache):
  - The server finds a nameserver for the target domain.
  - The server asks the nameserver to provide the host name to IP translation.
- To find the right nameserver, use DNS!

28

Server – Server Communication:
- If a server is asked to provide the mapping for a host outside it's domain (and the mapping is not in the server cache):
  - o The server finds a name server for the target domain
  - o The server asks the nameserver to provide the host name to IP translation
- To find the right nameserver, once again it can use DNS

(Refer Slide Time: 24:04-25:20)



Recursion

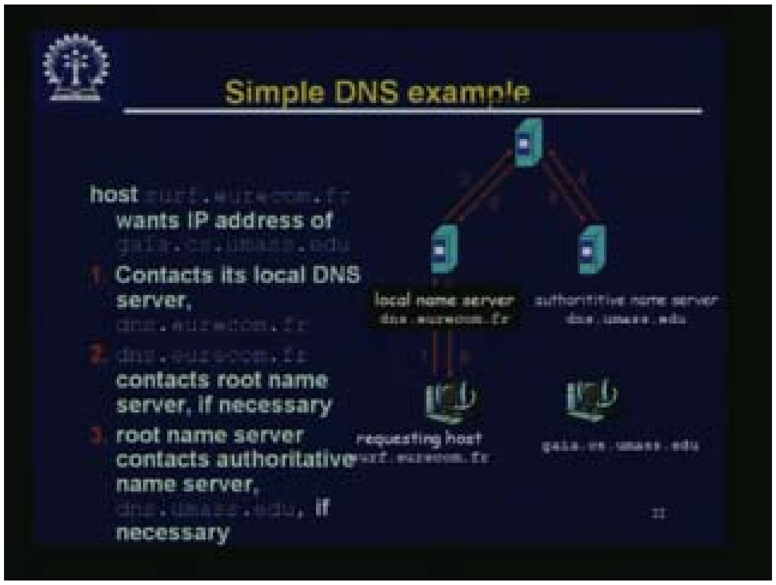
- A request can indicate that recursion is desired - this tells the server to find out the answer (possibly by contacting other servers).
- If recursion is not requested - the response may be a list of other name servers to contact.

21

Recursion:
- A request can indicate that recursion is desired - this tells the server to find out the answer (possibly by contacting other servers)
- If recursion is not requested - the response may be a list of other name servers to contact

So there are two versions of this DNS server, the recursive server and iterative server. If the recursion is requested and that is honored, suppose I am a name server, I will send the request to a name server whom I think may have a clue about how to do this request recursion. Now, he will make all the other contacts, the next level or may be another level contact as necessary and finally send me the answer. So this is the recursive version. In the iterative version what happens is that, he will simply give me a list of name servers to whom I can possibly contact directly to find out more. Then I make some more requests and finally get the name resolved.

(Refer Slide Time: 25:21-25:48)



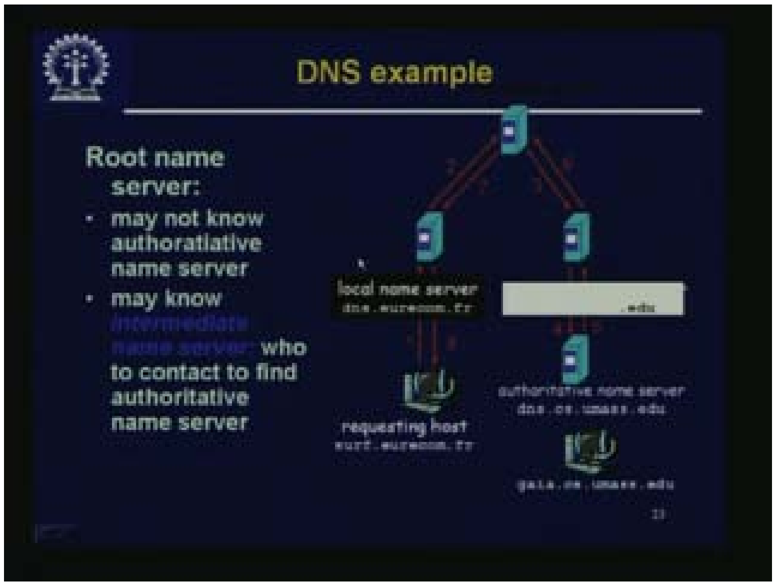
Simple DNS example

host surf.eurecom.fr wants IP address of gaia.cs.umass.edu

1. Contacts its local DNS server, dns.eurecom.fr
2. dns.eurecom.fr contacts root name server, if necessary
3. root name server contacts authoritative name server, dns.umass.edu, if necessary

For example, host1 makes a request to the local names server. Then the local name server may send the request for recursion to another name server. So, it will make the necessary request, get the response and then give the response.

(Refer Slide Time: 25:49-26:13)



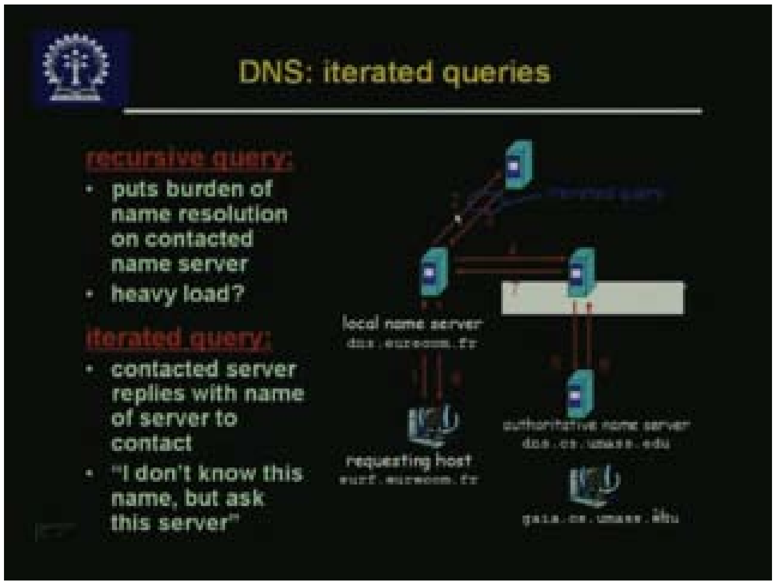
DNS example

Root name server:

- may not know authoratiative name server
- may know intermediate name server who to contact to find authoritative name server

In the iterative case, it knew in one shot but it may not be in one shot. When recursion is requested this may go deep down and finally find out and then send the request.

(Refer Slide Time: 26:14-26:37)



DNS: iterated queries

recursive query:

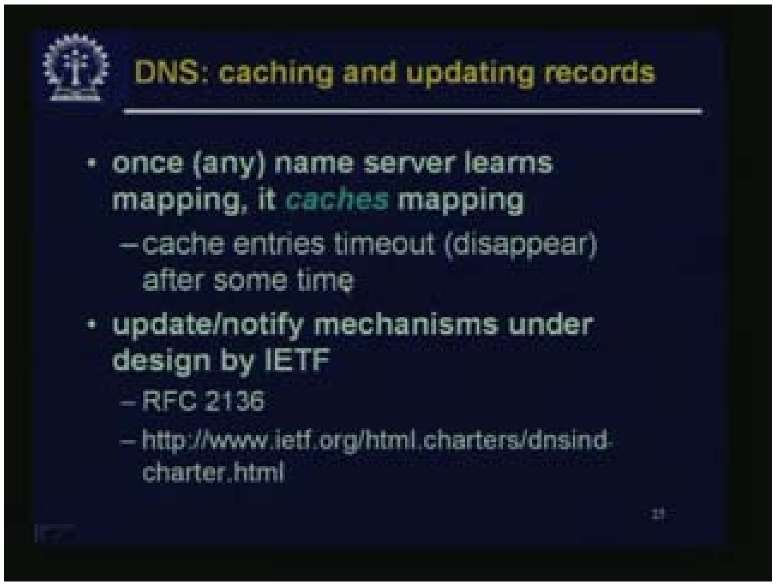
- puts burden of name resolution on contacted name server
- heavy load?

iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"

Whereas in the iterated queries, he gives a query and gets some list, he gives it to the next name server and gets some more response. Then finally he will get it and send it for you and then it will be resolved. This is an iterated query.

(Refer Slide Time: 26:38-26:59)

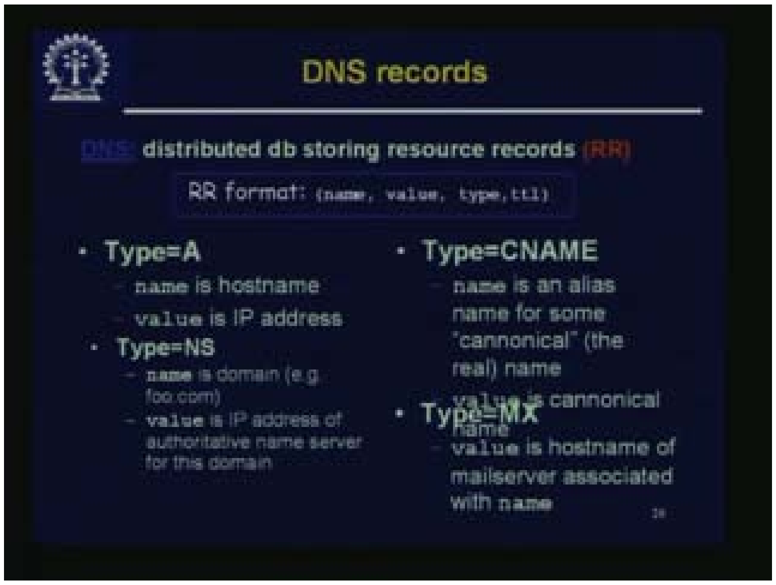


DNS: caching and updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time
- update/notify mechanisms under design by IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

DNS caching and updating of records:
- Once (any) name server learns mapping, it caches mapping
- Cache entries timeout (disappear) after some time
- Update/notify mechanisms under design by IETF (how the cache is to be managed)

(Refer Slide Time: 27:00-27:46)



DNS records

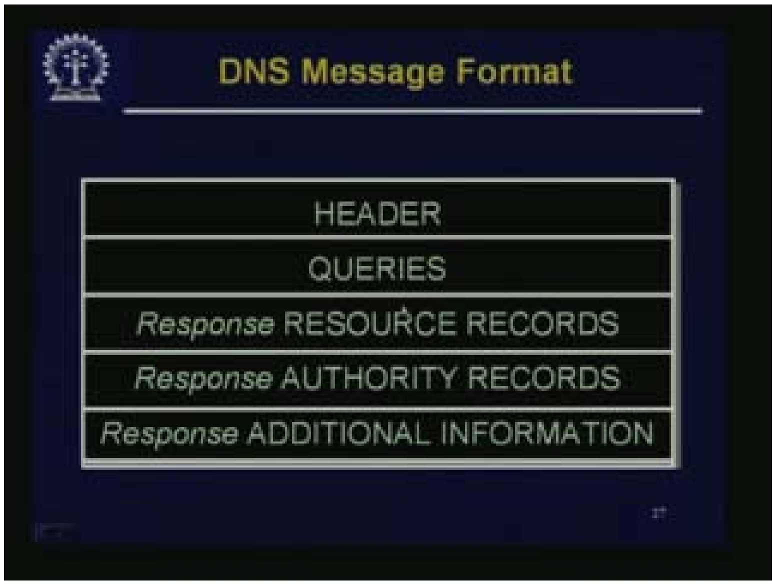
DNS: distributed db storing resource records (RR)

RR format: (name, value, type,ttl)

- Type=A
  - name is hostname
  - value is IP address
- Type=NS
  - name is domain (e.g. foo.com)
  - value is IP address of authoritative name server for this domain
- Type=CNAME
  - name is an alias name for some "cannonical" (the real) name
  - value is cannonical name
- Type=MX
  - value is hostname of mailserver associated with name

We will not going into all the details, although they are given in the slide. These are the so called resource records or RR. The RR (resource records) have a format Type equal to A where the name is hostname and value is IP address. This is the most common one. Type equal to NS where name is domain and value is IP address of authoritative name server for this domain. Type equal to CNAME where name is an alias name for some canonical (the real) name and value is canonical. Type equal to MX is for mail server records, so these are the various resource records which are handled by the DNS.

(Refer Slide Time: 27:27-28:02)

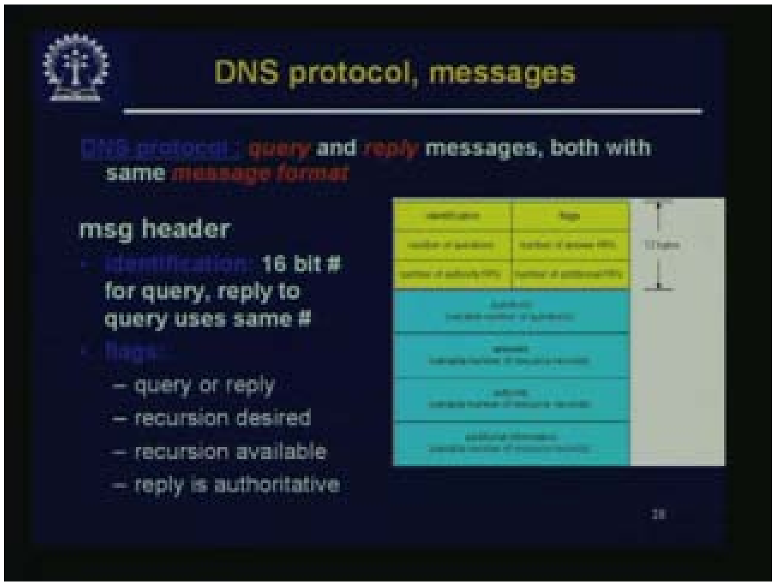


DNS Message Format

HEADER
QUERIES
Response RESOURCE RECORDS
Response AUTHORITY RECORDS
Response ADDITIONAL INFORMATION

This is the DNS message format. We have a header followed by some queries, followed by some resource records and some authority records if it is from alternative name server and may be some additional information. This is the DNS message format.

(Refer Slide Time: 28:03-28:18)



DNS protocol, messages

DNS protocol : query and reply messages, both with same message format

msg header

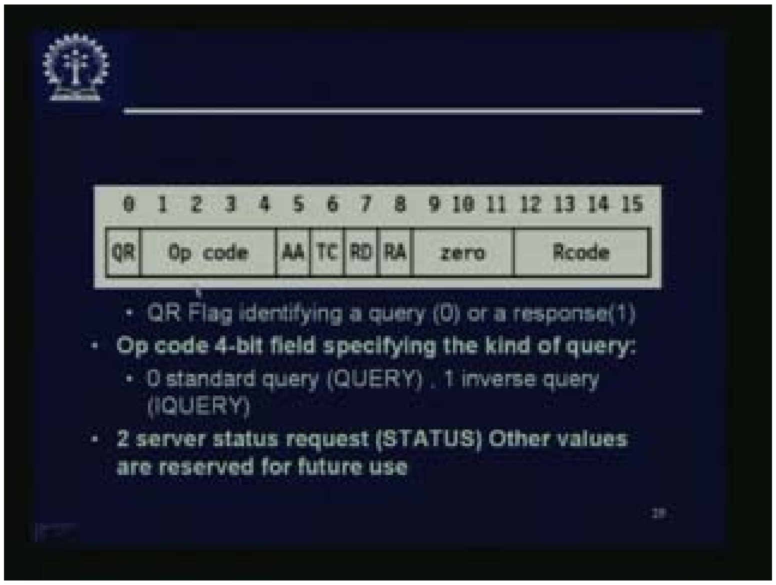
- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

identification
flags
number of questions
number of answer RRs
number of authority RRs
number of additional RRs
12 bytes
questions (variable number of questions)
answers (variable number of resource records)
authority (variable number of resource records)
additional information (variable number of resource records)

28

We have in the message header, a 16-bit # for identification, 16-bit # for query, and reply to query uses the same # and then we have a number of flags.

(Refer Slide Time: 28:19-28:44)



0	1 2 3 4	5	6	7	8	9 10 11	12 13 14 15
QR	Op code	AA	TC	RD	RA	zero	Rcode

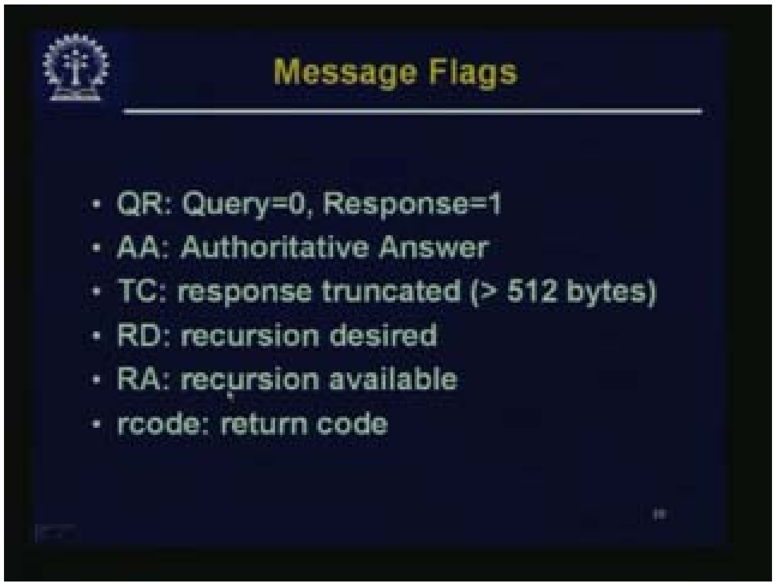
- QR Flag identifying a query (0) or a response(1)
- **Op code 4-bit field specifying the kind of query:**
  - 0 standard query (QUERY) , 1 inverse query (IQUERY)
- **2 server status request (STATUS) Other values are reserved for future use**

29

- QR flag identifying a query (0) or a response (1)
- Op code is a 4-bit field specifying the kind of query:

0 for standard query (QUERY); 1 for inverse query (QUERY)
- Server status (STATUS).
-

(Refer Slide Time: 28:45-29:10)

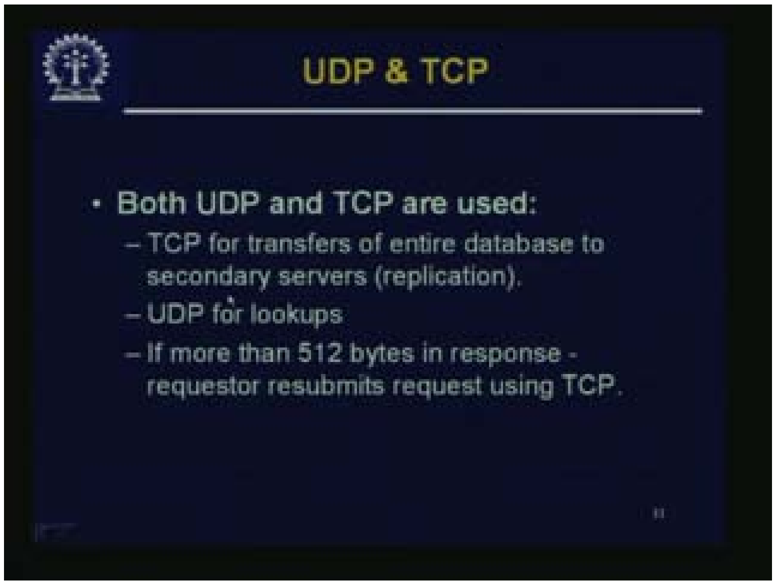


Message Flags

- QR: Query=0, Response=1
- AA: Authoritative Answer
- TC: response truncated (> 512 bytes)
- RD: recursion desired
- RA: recursion available
- rcode: return code

Other values are reserved for future use:
- AA: is authoritative answer that means this it is coming from an Authoritative Name Server
- TC: to see if the response has been truncated. Actually if it is truncated, it switches from UDP to TCP
- RD: recursion desired
- RA: recursion available etc
- rcode: is return code

(Refer Slide Time: 29:11-30:19)



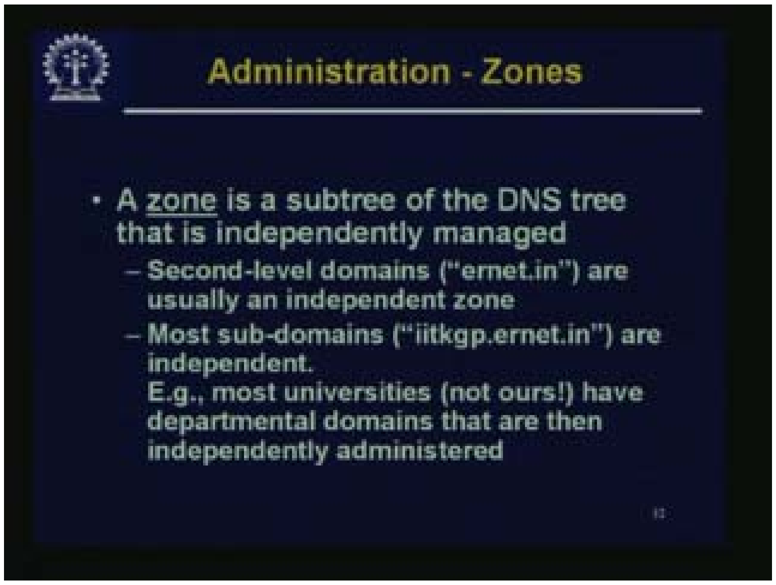
UDP & TCP

- Both UDP and TCP are used:
  - TCP for transfers of entire database to secondary servers (replication).
  - UDP for lookups
  - If more than 512 bytes in response - requestor resubmits request using TCP.

Both UDP and TCP are used by name server:

- TCP for transfers of entire database to secondary servers
- UDP for lookups. The lookups that we discussed about, usually use the UDP protocol. UDP protocol is used as it is a very simple protocol with no extra over head. But there is a limit to that UDP and in response to UDP you will just get one packet which is just 512 bytes
- If the response requires more than 512 bytes, then the requester resubmits request using TCP. If the message is truncated the flag is set and when the client sees the flag it opens a TCP connection with the corresponding name server and then resubmits the request so that it can get a longer request.

(Refer Slide Time: 30:20-30:46)



Administration - Zones

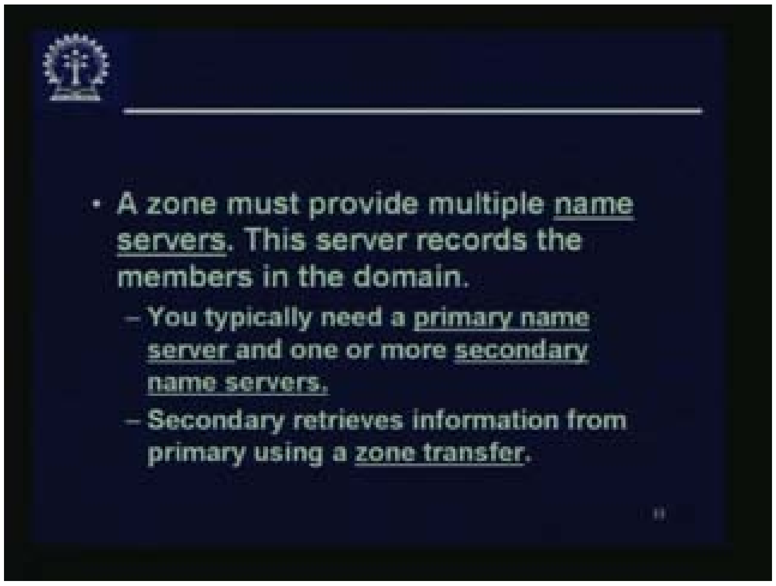
- A zone is a subtree of the DNS tree that is independently managed
  - Second-level domains ("ernet.in") are usually an independent zone
  - Most sub-domains ("iitkgp.ernet.in") are independent.
    E.g., most universities (not ours!) have departmental domains that are then independently administered

12

We have already discussed the administration zones.

- A zone is a sub tree of the DNS tree that is independently managed.
  - o Second level domains (ernet.in) are usually an independent zone
  - o Most sub domains (iitkgp.ernet.in) are also independent. Independent means that what happens under this sub-domain is their business
  - o Eg: Most universities have departmental domains that are then independently administered

(Refer Slide Time: 30:47-32:24)



- A zone must provide multiple name servers. This server records the members in the domain.
  - You typically need a primary name server and one or more secondary name servers.
  - Secondary retrieves information from primary using a zone transfer.

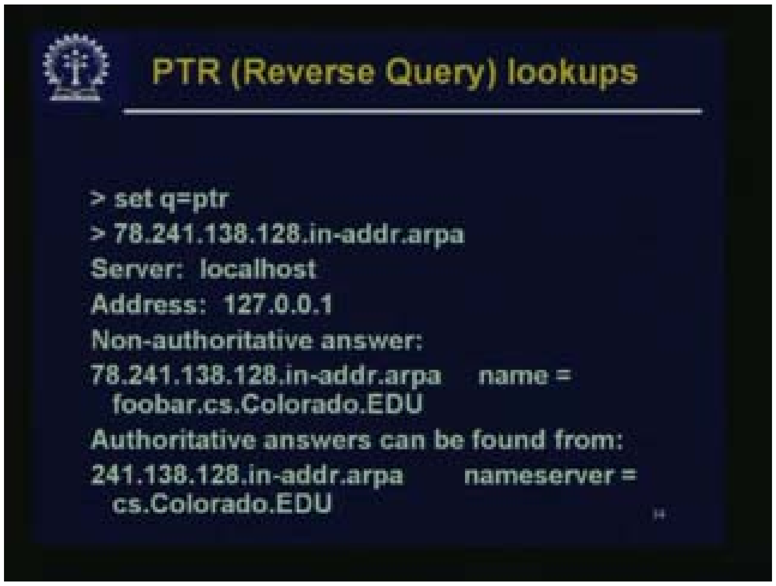
13

- A zone must provide multiple name servers. This server records the members in the domain
- You typically need a primary name server and one or more secondary name servers
- Secondary retrieves information from primary using a zone transfer, using a TCP connection

The reason why the secondary name server is required is sometimes more than one secondary server is kept and the reason is that it is very vital for everybody. Suppose if you are administering a domain and you want to be independent then you can create extra sub domains under you where you have to maintain your own name server. You cannot maintain just one name sever but you have to maintain multiple name servers so that when the primary name server fails the secondary can immediately take over.

From time to time the primary will cache the data and from time to time or the database will be shifted to the secondary one so that they remain more or less in sync. Therefore as soon as primary goes down the secondary can start acting as a primary and be the authoritative name server for this particular domain. There will be requests from other people to you for IP addresses in your domain and you are bound to give that. That is the reason you need to have these different name servers and good network connection to handle all these requests.

(Refer Slide Time: 32:25-33:02)



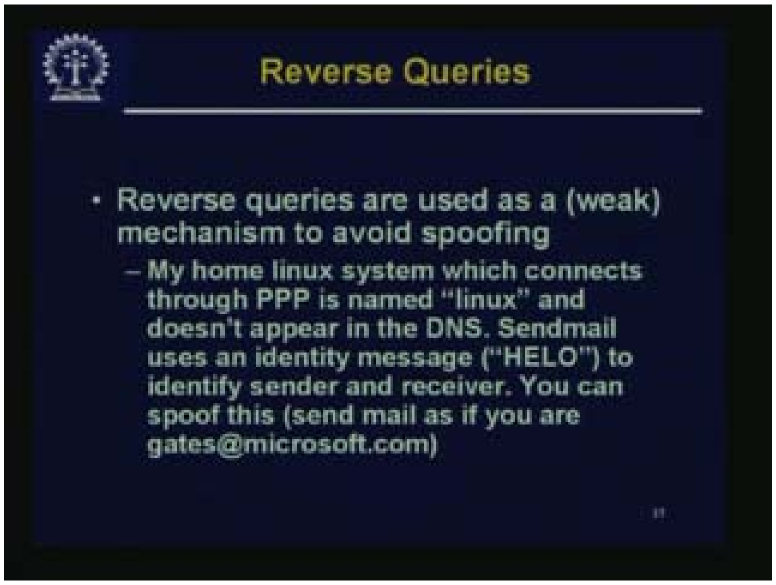
PTR (Reverse Query) lookups

```
> set q=ptr
> 78.241.138.128.in-addr.arpa
Server:  localhost
Address:  127.0.0.1
Non-authoritative answer:
78.241.138.128.in-addr.arpa     name =
  foobar.cs.Colorado.EDU
Authoritative answers can be found from:
241.138.128.in-addr.arpa        nameserver =
  cs.Colorado.EDU
```

There is a reverse query which is sometimes used as a sort of weak protection against spoofing.

- Set q equal to ptr i.e. ptr is the reverse query for lookup. So here what is done is that, an IP address is given and the domain name is found out. Just as we have ARP and RARP similarly we have query and reverse query.

(Refer Slide Time: 33:03-33:42)

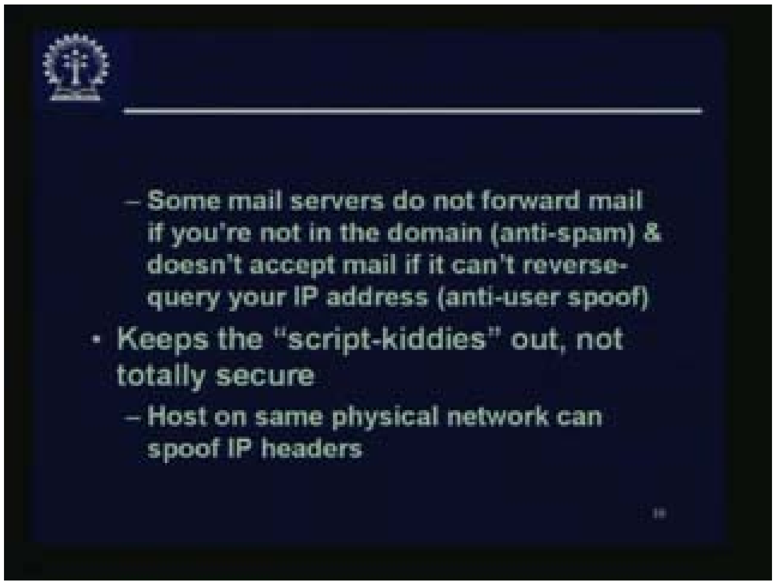


Reverse Queries

- Reverse queries are used as a (weak) mechanism to avoid spoofing
  - My home linux system which connects through PPP is named "linux" and doesn't appear in the DNS. Sendmail uses an identity message ("HELO") to identify sender and receiver. You can spoof this (send mail as if you are gates@microsoft.com)

Reverse queries are used as a weak mechanism to avoid spoofing. This is weak because those machines may have a name but if it connects to the network through a modem and is given a dynamic IP address then sometimes it may not get reflected. So sendmail uses an identity message to identify the sender and receiver but this can easily be spoofed.

(Refer Slide Time: 33:43-35:18)

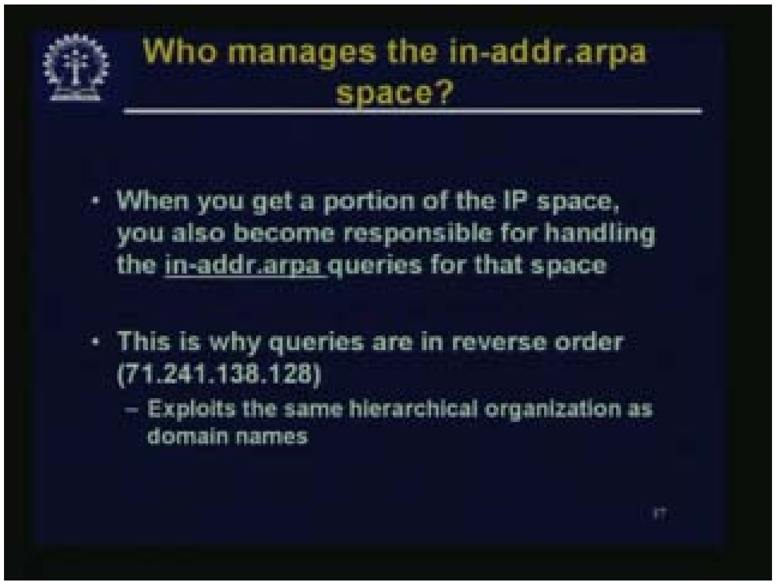


- Some mail servers do not forward mail if you're not in the domain (anti-spam) & doesn't accept mail if it can't reverse-query your IP address (anti-user spoof)
- Keeps the "script-kiddies" out, not totally secure
  - Host on same physical network can spoof IP headers

- Some mail servers do not forward mail if you are not in the domain. This is for anti-spam. Spams are mails which are spuriously generated mostly by programs/machines which are automatically generated and sent to millions of people. It is estimated that the majority of the mails which go through are actually spam. If the mail is not for my domain then I am not going to relay this to other mail servers. Sometimes you can do that but sometimes you have to relay because it is your responsibility to relay. And does not accept mail if it can reverse query your IP address. As soon as a mail comes it will find out from which IP address this mail has come from and send a reverse query. By reverse query the name looks responsible may be it is coming from some known University etc then accept the mail. If you cannot reverse query it then maybe it is coming from some spurious source and it does not accept.

- This is not totally secure because hosts on same physical network can spoof IP headers of domains which are sort of respectable, but anyway this maybe of some use.

(Refer Slide Time: 35:19-35:35)



Who manages the in-addr.arpa space?

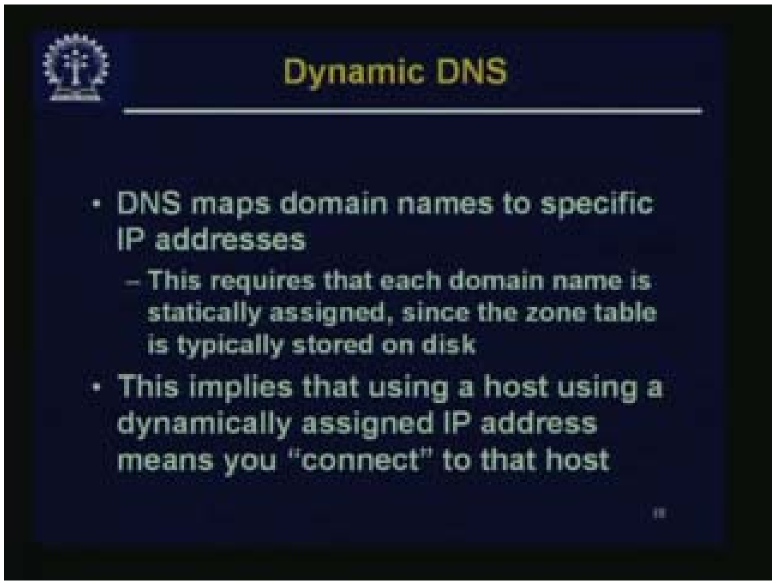
- When you get a portion of the IP space, you also become responsible for handling the in-addr.arpa queries for that space
- This is why queries are in reverse order (71.241.138.128)
  - Exploits the same hierarchical organization as domain names

17

Who manages the in-addr arpa space?

- When you get a portion of the IP space you also become responsible for handling the in-addr.arpa queries for that space.
- This is why queries are in reverse order.

(Refer Slide Time: 35:36-35:57)



Dynamic DNS

- DNS maps domain names to specific IP addresses
  - This requires that each domain name is statically assigned, since the zone table is typically stored on disk
- This implies that using a host using a dynamically assigned IP address means you "connect" to that host

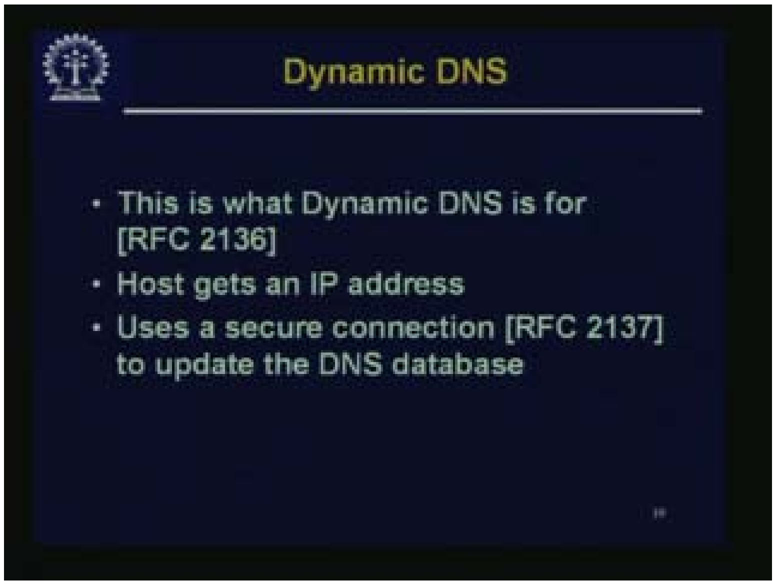
Dynamic DNS:

- DNS maps domain names to specific IP addresses.

o This requires that each domain name is statically assigned, since the zone table is typically stored on disk.

- This implies that a host using a dynamically assigned IP addresses means you connect to that host.

(Refer Slide Time: 35:58-36:34)



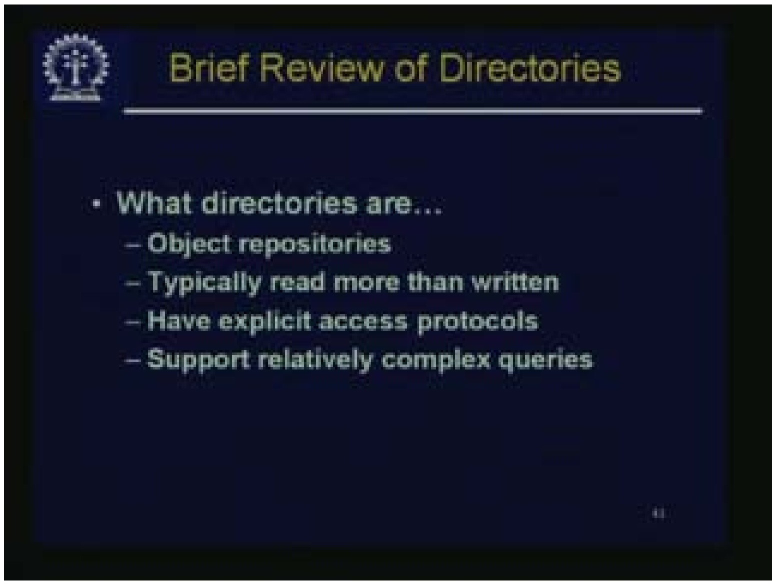
Dynamic DNS

- This is what Dynamic DNS is for [RFC 2136]
- Host gets an IP address
- Uses a secure connection [RFC 2137] to update the DNS database

This is what dynamic DNS is for. This is not widely implemented but there is an RFC for this and it uses a secure connection for doing a dynamic DNS update. You have to sort of update the name server database and anybody cannot use this name server and so you have to use a secure connection to update the DNS database.

Next, we take up the topic of directories. This is similar to DNS but more general. Actually the directories sometimes uses DNS,. Many of you have used mails and email clients where you have your own address book from where you can look up the email address of those who have sent you mails. One example of a directory service could be some kind of a global address book where we can find the email address of anybody. This was the original intention for creating the directory. But of course people who produce spam mails have sort of killed this idea. So nobody wants his email address to be in a directory which is universally available so that everybody can send junk mails. Anyway, directory can be used for many other purposes as well.

(Refer Slide Time: 37:59-38:48)



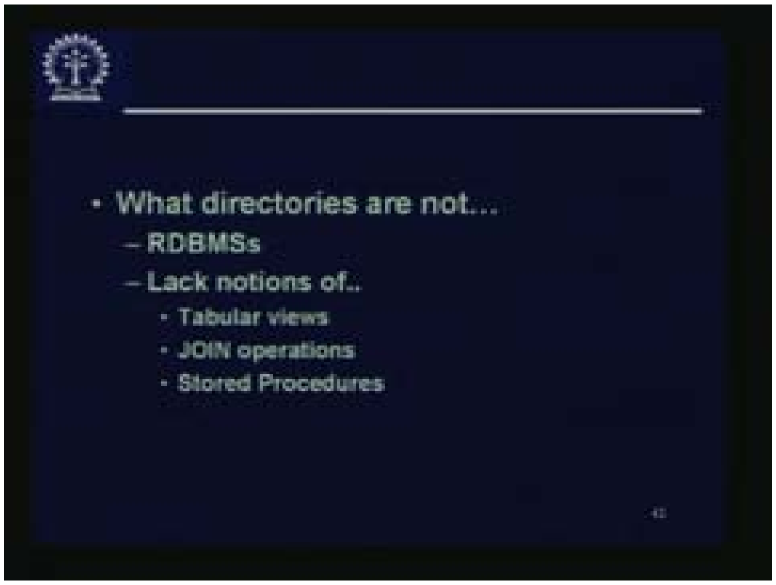
Brief Review of Directories

- What directories are…
  - Object repositories
  - Typically read more than written
  - Have explicit access protocols
  - Support relatively complex queries

- What directories are...?
  - They are object repositories
  - Typically read more than written
  - Have explicit access protocols
  - Support relatively complex queries

DNS queries are simple but it supports relatively complex queries. For example, you can have something like give me the email addresses (assuming that email addresses are still available) of all people who live in Delhi whose name contains ram etc. You may be looking for somebody so you can have more complex queries than you have in a DNS.

(Refer Slide Time: 38:49-39:03)

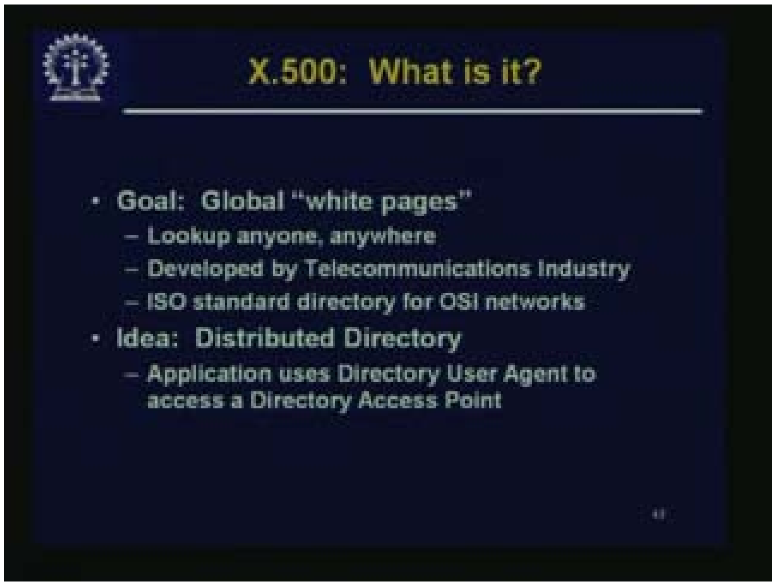


- What directories are not…
  - RDBMSs
  - Lack notions of..
    - Tabular views
    - JOIN operations
    - Stored Procedures

42

But directories are not meant to be RDBMSs, they are just for looking up. So, lack notions of tabular views, join operations, stored procedures etc. They are not regular RDBMS but they are just for this particular service.

(Refer Slide Time: 39:04-39:41)



X.500: What is it?

- Goal: Global "white pages"
  - Lookup anyone, anywhere
  - Developed by Telecommunications Industry
  - ISO standard directory for OSI networks
- Idea: Distributed Directory
  - Application uses Directory User Agent to access a Directory Access Point

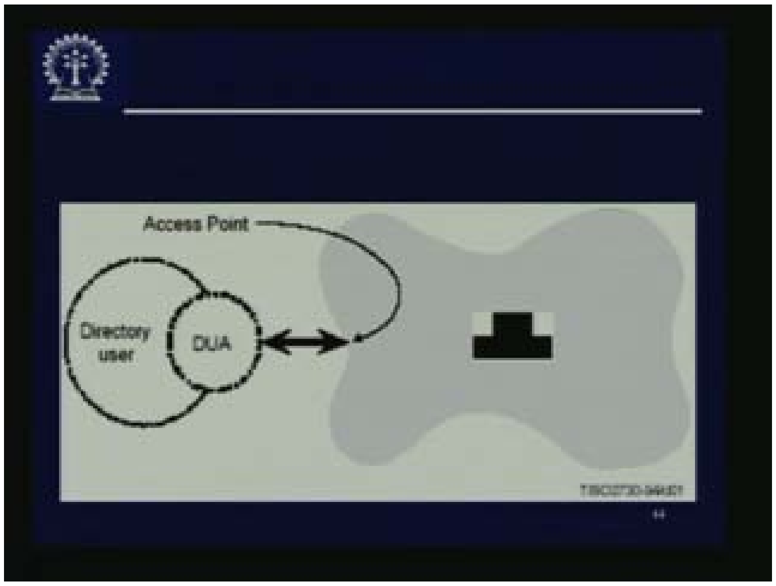
43

X.500 was originally how directory was envisaged by the telecom industry.

- The goal was to have global white pages
  - o Lookup anyone anywhere
  - o Developed by telecom industry
  - o ISO standard for OSI networks

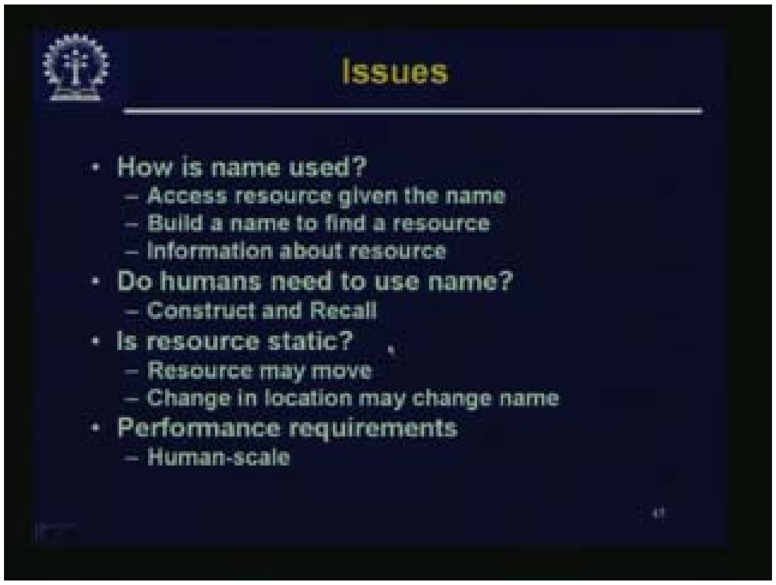
- Idea was distributed directory
  - o Application uses distributed directory structure
  - o Application uses directory user agent to access a directory access point.

(Refer Slide Time: 39:42-39:53)



The picture is something like this; you have a directory user here who uses a directory user agent which in turn connects an access point to the directory.

(Refer Slide Time: 39:54-40:39)



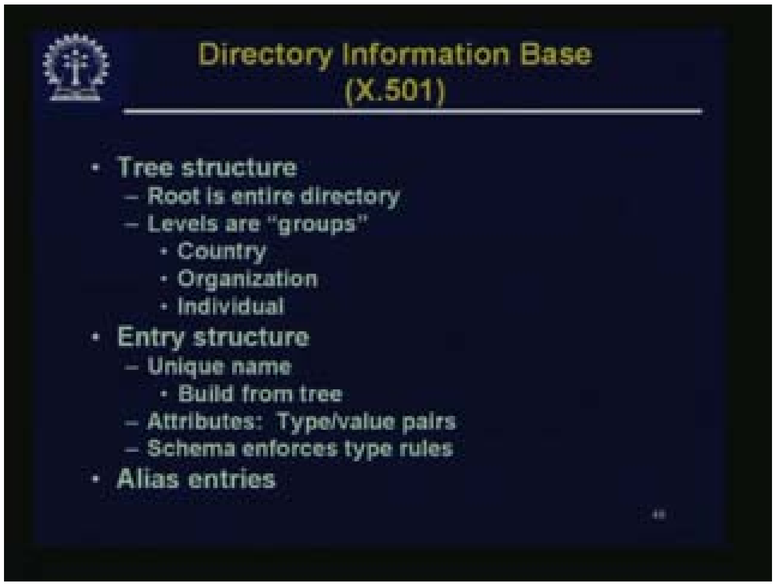
Issues

- How is name used?
  - Access resource given the name
  - Build a name to find a resource
  - Information about resource
- Do humans need to use name?
  - Construct and Recall
- Is resource static?
  - Resource may move
  - Change in location may change name
- Performance requirements
  - Human-scale

- How is the name used?
  - o Access resource given the name

o Build a name to find a resource
o Information about resource.
o These are the different uses of names
- Do only programs look at these names? Sometimes humans also need to use the names for constructing the names recalling names.
- Is resource static?
o Sometimes resource may move
o Change in location may change the name of a particular resource.
- Performance requirements
o Human scale

(Refer Slide Time: 40:40-41:12)

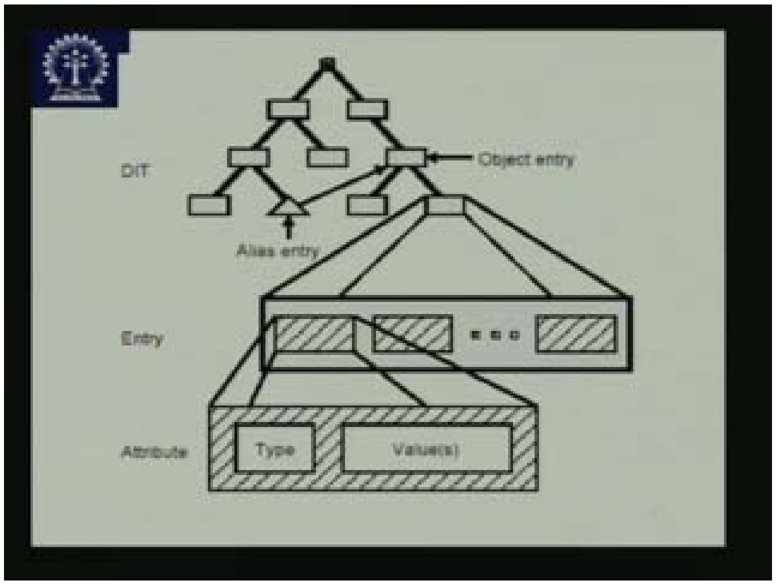


Directory Information Base (X.501)

- Tree structure
  - Root is entire directory
  - Levels are "groups"
    - Country
    - Organization
    - Individual
- Entry structure
  - Unique name
    - Build from tree
  - Attributes: Type/value pairs
  - Schema enforces type rules
- Alias entries

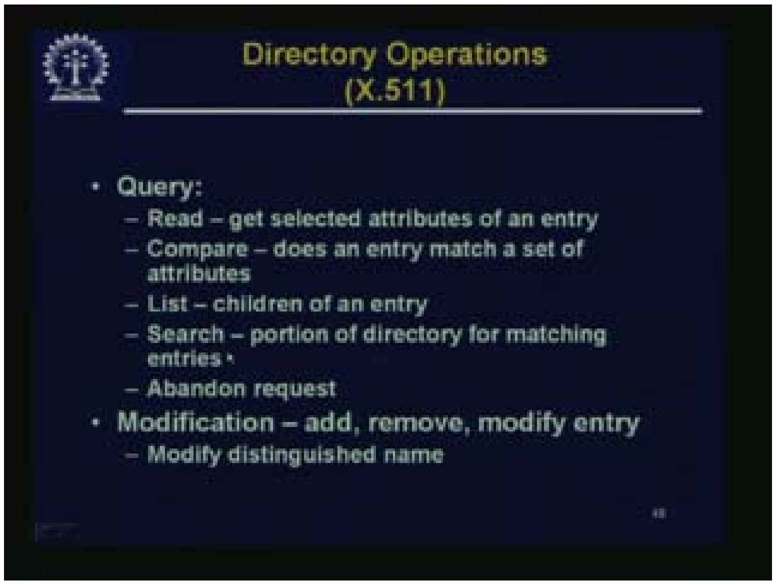
Directory information base which is defined in X.501 is given as a tree structure.
o Root is the entire directory
o Levels are groups. For ex: country, organization, individual.
- Entry structure
o Unique name build from tree
o Attributes: Type/Value pairs
o Schema enforces type rules
- There may be alias entries also.

(Refer Slide Time: 41:13-41:53)



Directory structure may look something like this. You have these different levels, which starting from the top may represent some organizations and then some sub organizations and finally you have the objects. Now, in an object entry you will have some names and each of these names should have a series of type value pairs. There may be more than one particular name; it may have a number of attribute and each attribute will have a type/value pair. This is the general structure of a X.501 tree structure.

(Refer Slide Time: 41:54-42:28)



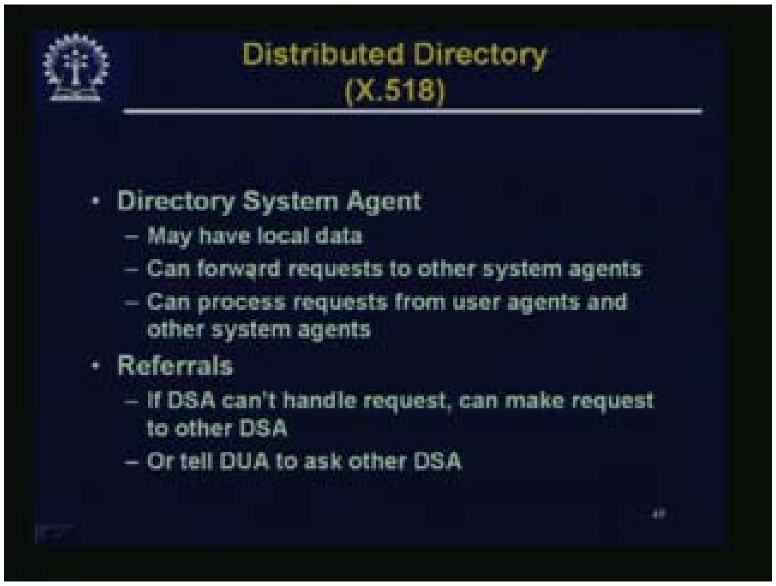
Directory Operations (X.511)

- Query:
  - Read – get selected attributes of an entry
  - Compare – does an entry match a set of attributes
  - List – children of an entry
  - Search – portion of directory for matching entries
  - Abandon request
- Modification – add, remove, modify entry
  - Modify distinguished name

- Query is to this system defined in X.511
  - o Query is a read, get selected attributes of an entry

o Compare does an entry match a set of attributes?
o List children of an entry
o Search, Abandon request etc
o These are all kinds of queries are possible
- Modification you can modify these records – add, remove, modify entry
o Modify distinguished names etc

(Refer Slide Time: 42:29-43:14)



Distributed Directory (X.518)

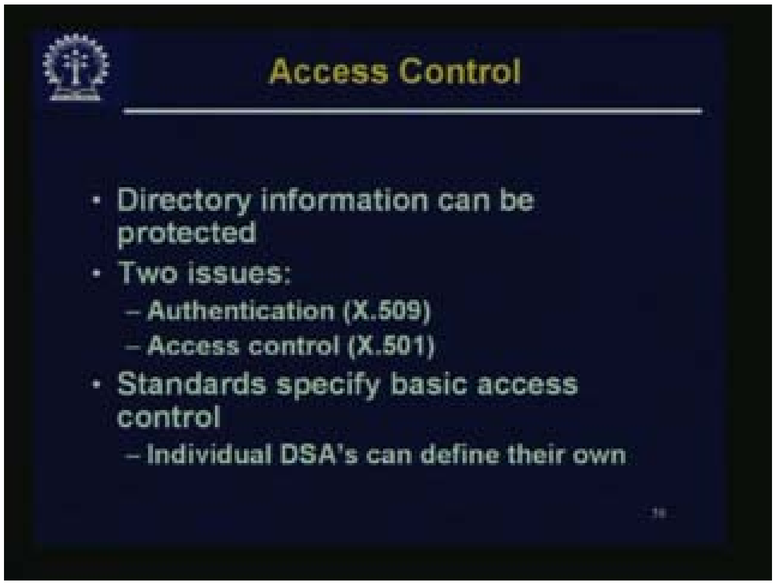
- Directory System Agent
  - May have local data
  - Can forward requests to other system agents
  - Can process requests from user agents and other system agents
- Referrals
  - If DSA can't handle request, can make request to other DSA
  - Or tell DUA to ask other DSA

- There is a directory system agent.
o It may have some local data
o Can forward request to other system agents
o Can process requests from user agents and other system agents

So, these are like the name server system with dissolvers and also this is the system using directory system agents which can do the query processing. It may get the data locally or it may forward the request to other systems.

- Referrals:
o If DSA cannot handle the request it can make request to other DSA just as you can make iterative and recursive queries in DNS.
o Or tell DUA to ask other DSA, this is the iterative process.

(Refer Slide Time: 43:15-44:14)

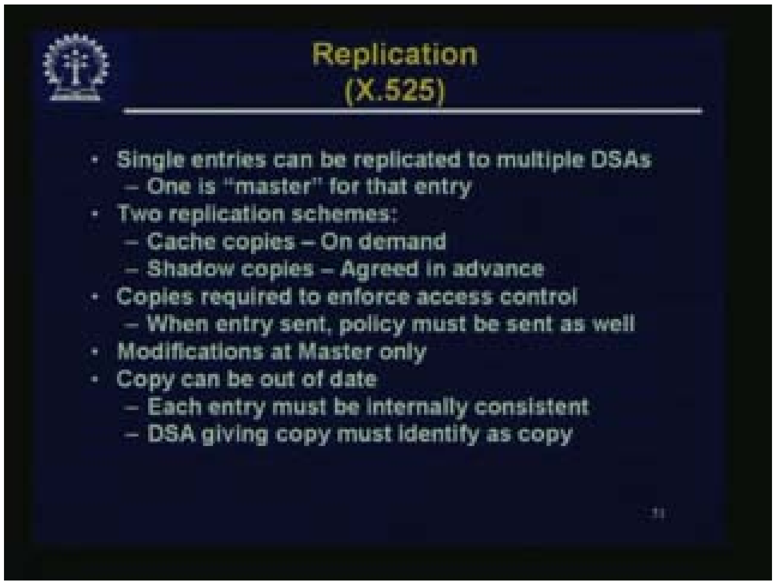


Access Control

- Directory information can be protected
- Two issues:
  - Authentication (X.509)
  - Access control (X.501)
- Standards specify basic access control
  - Individual DSA's can define their own

- Directory information can be protected. Actually they are usually protected.
- There are two issues:
  - Authentication defined in X.509
  - Access control defined in X.501
- This directory by itself does not give you security. So you have to have other components in order to ensure security. But a directory can be used for some authentication services, some security purposes, etc directory can very well be used. They are actually used that way.
- Standards specify basic access control and individual DSA's can define their own access control. They can specify to whom they are going to allow access to their local databases.

(Refer Slide Time: 44:15-44:59)



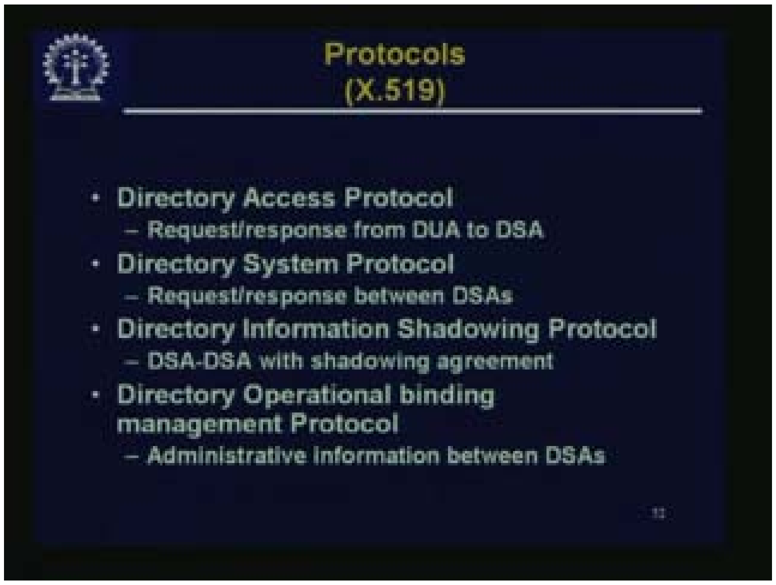
Replication (X.525)

- Single entries can be replicated to multiple DSAs
  - One is "master" for that entry
- Two replication schemes:
  - Cache copies – On demand
  - Shadow copies – Agreed in advance
- Copies required to enforce access control
  - When entry sent, policy must be sent as well
- Modifications at Master only
- Copy can be out of date
  - Each entry must be internally consistent
  - DSA giving copy must identify as copy

Replication: This is defined in X.525.

- Single entries can be replicated to multiple DSAs. Just like you have a primary name server and a number of secondary name servers, similarly you can have a directory in a primary or master and then you can have replication.
- Two replication schemes:
  - o Cache copies - On demand
  - o Shadow copies - Agreed in advance from time to time
- There is a transfer, copy required to enforce access control.
  - o When entry sent, policy must be sent as well
- Modification is done at the master only
- Copy can be out of date
  - o How to handle that is defined in X.525

(Refer Slide Time: 45:00-45:30)



Protocols (X.519)

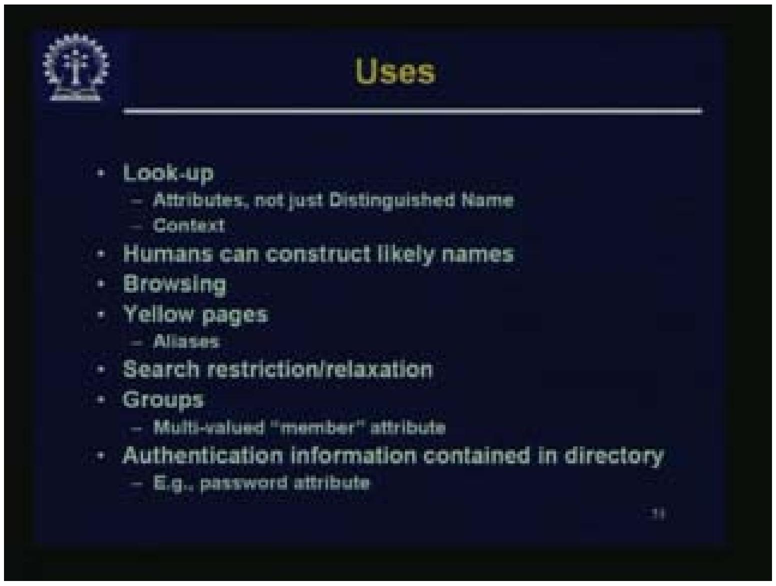
- Directory Access Protocol
  - Request/response from DUA to DSA
- Directory System Protocol
  - Request/response between DSAs
- Directory Information Shadowing Protocol
  - DSA-DSA with shadowing agreement
- Directory Operational binding management Protocol
  - Administrative information between DSAs

12

There are a number of protocols which are defined in X.519.

- Directory access protocol means the structure of the query and other things would be defined.
- Directory system protocol
  - o Request/response between DSAs
- Directory information shadowing protocol
  - o DSA-DSA with shadowing agreement
- Directory operational binding management protocol. There are a number of protocols.

(Refer Slide Time: 45:31-46:07)



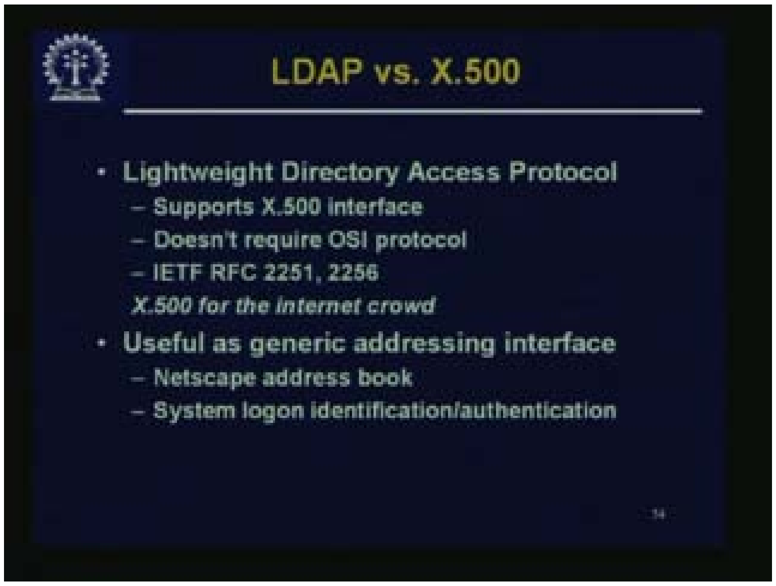
Uses

- Look-up
  - Attributes, not just Distinguished Name
  - Context
- Humans can construct likely names
- Browsing
- Yellow pages
  - Aliases
- Search restriction/relaxation
- Groups
  - Multi-valued "member" attribute
- Authentication information contained in directory
  - E.g., password attribute

13

Uses are of course for Look-up
o Attributes, not just distinguished name
o Context
- Humans can construct likely names
- Browsing
- Yellow pages
o Aliases also may be given
- Search restriction/relaxation may be there
- Groups may be defined that means having a number of members who will be having they are own attributes
- Authentication information that may be contained in the directory and so on
o Directory may be used for various purposes

(Refer Slide Time: 46:08-47:20)



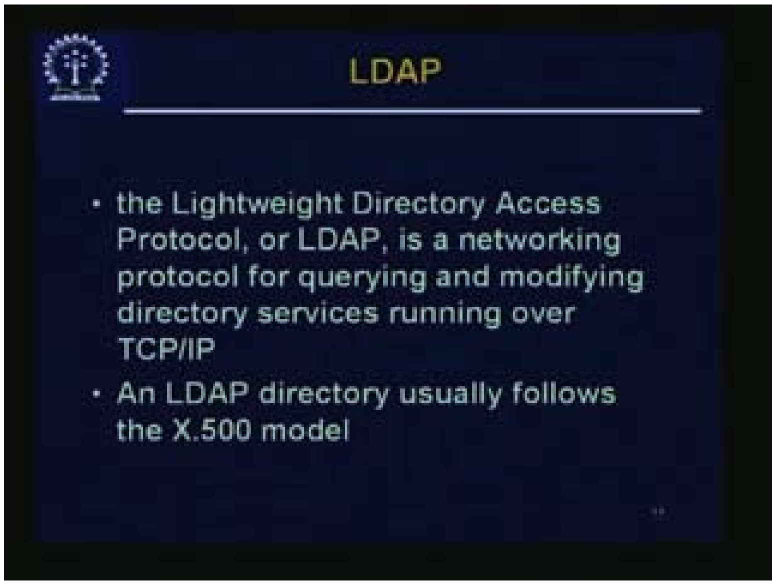
LDAP vs. X.500

- Lightweight Directory Access Protocol
  - Supports X.500 interface
  - Doesn't require OSI protocol
  - IETF RFC 2251, 2256

  X.500 for the internet crowd
- Useful as generic addressing interface
  - Netscape address book
  - System logon identification/authentication

We will look at LDAP, which is:
- The lightweight directory access protocol designed by the same people who designed OSI. X.500. It actually tends to be a little complex, it is heavy and intuits the OSI all the seven layers. Now for the internet purpose which actually uses the TCP/IP stack rather than the seven layer OSI stack there was a lightweight directory access protocol which can interoperate at least on one side, that LDAP can use that X.500 directory service. but this is much simpler than X.500 and LDAP is used in many places.
- This is a lightweight directory access protocol.
o Supports X.500 interface
o Doesn't require the OSI protocol. This uses the TCP/IP protocol
o So this is X.500 for the internet crowd
- Useful as generic addressing interface
- Like Netscape address book, etc

(Refer Slide Time: 47:21-47:35)

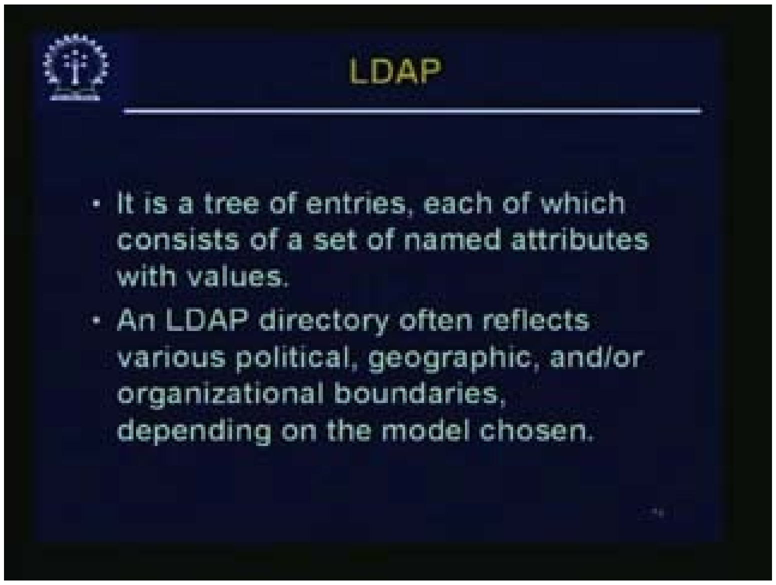


LDAP

- the Lightweight Directory Access Protocol, or LDAP, is a networking protocol for querying and modifying directory services running over TCP/IP
- An LDAP directory usually follows the X.500 model

- The LDAP or lightweight directory access protocol is a networking protocol for querying and modifying directory services running over TCP/IP.
- An LDAP directory usually follows the X.500 model

(Refer Slide Time: 47:36-48:11)

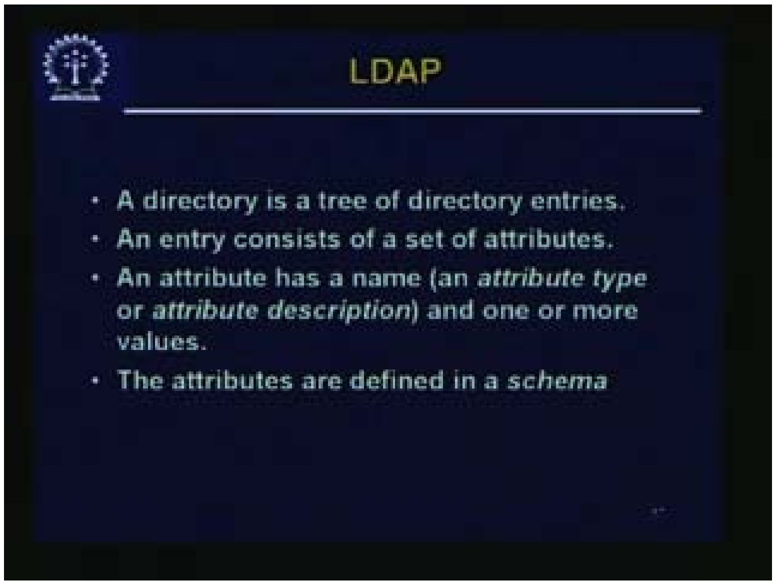


LDAP

- It is a tree of entries, each of which consists of a set of named attributes with values.
- An LDAP directory often reflects various political, geographic, and/or organizational boundaries, depending on the model chosen.

- It is a tree of entries, each of which consists of a set named attributes with values
- An LDAP directory often reflects various political, geographic, and/or organizational boundaries depending on the model chosen. When you do that you can also define your security policies based on this directory and based on this boundary, especially authentication service.

(Refer Slide Time: 48:12-48:21)

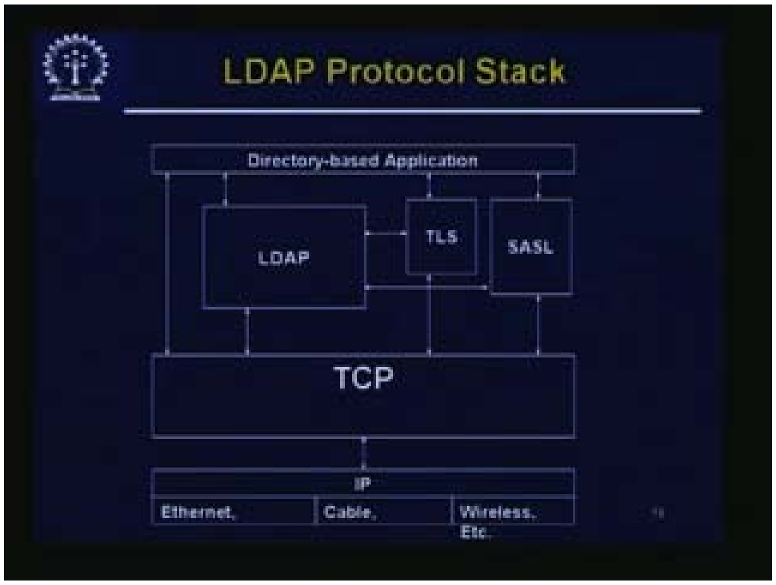


LDAP

- A directory is a tree of directory entries.
- An entry consists of a set of attributes.
- An attribute has a name (an *attribute type* or *attribute description*) and one or more values.
- The attributes are defined in a *schema*

- A directory is a tree of directory entries
- An entry consists of a set of attributes
- An attribute values pair
- The attributes are defined in the schema.

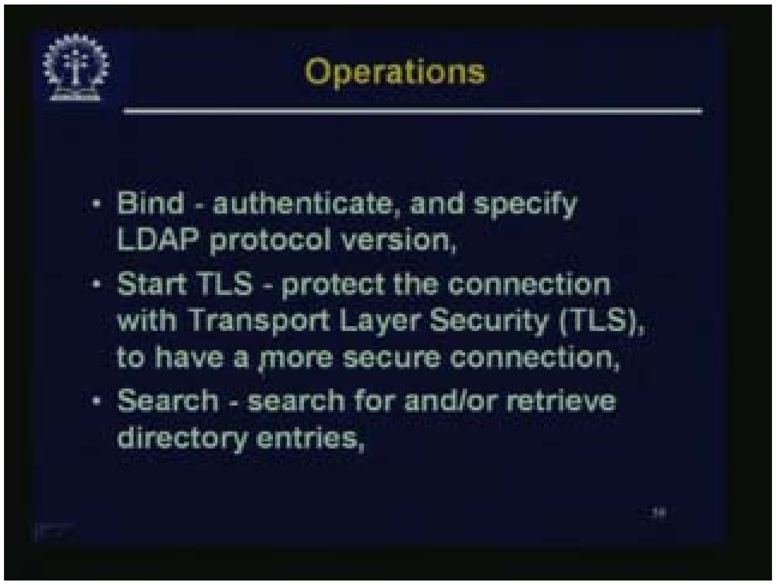
(Refer Slide Time: 48:22-49:58)



LDAP Protocol Stack

This would be the protocol stack for LDAP. You have a directory based application, some authorization service or access to some information which may be there for the organization which uses LDAP. LDAP may use TLS (Transport Level Security). Actually you could use SSL also. In future we will give one lecture to security because it has become so important. Now, in an organizational context, a directory may be an important component of the entire security arrangement. Security is the complex issue. But for LDAP we require that we communicate securely in many cases. And many LDAP implementations support this TLS Transport Level Security. We can also use SSL or SASL and these uses TCP. TCP sits on the IP which sits on the other layer etc. So it comes in between the directory based application and the TCP layer. This is where it stands in the protocol stack.

(Refer Slide Time: 49:59-50:50)

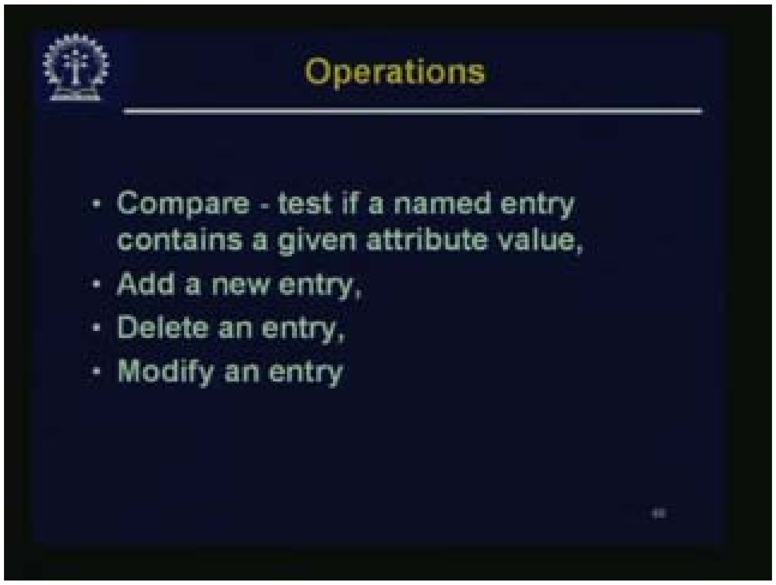


Operations

- Bind - authenticate, and specify LDAP protocol version,
- Start TLS - protect the connection with Transport Layer Security (TLS), to have a more secure connection,
- Search - search for and/or retrieve directory entries,

This mentions some of the operations which are in LDAP:
- Bind - authenticate and specify LDAP protocol version. This actually starts the process.
- Start TLS - protect the connection with Transport Layer Security to have a more secure connection. Since you are giving some access to information, you need to put some security feature in that. So that is the start TLS.
- Search - Search for and/or retrieve directory entries. If you go through the access controls then you can search for some records.

(Refer Slide Time: 50:51-51:08)

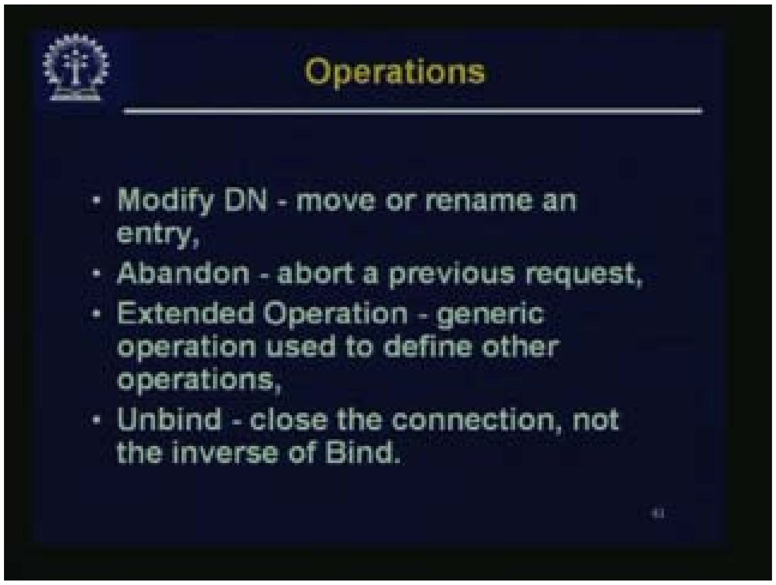


Operations

- Compare - test if a named entry contains a given attribute value,
- Add a new entry,
- Delete an entry,
- Modify an entry

- Compare - test if a named entry contains a given attribute value
- You can add a new entry
- You can delete an entry
- You can modify an entry. These are various LDAP operations.

(Refer Slide Time: 51:09-51:37)



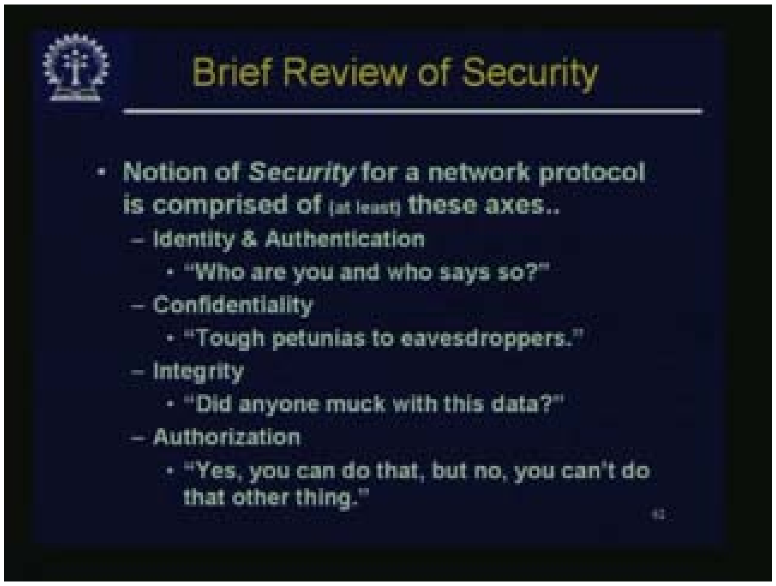
Operations

- Modify DN - move or rename an entry,
- Abandon - abort a previous request,
- Extended Operation - generic operation used to define other operations,
- Unbind - close the connection, not the inverse of Bind.

- Modify DN - move or rename an entry
- Abandon - abort a previous request
- Extended operation - generic operation used to define other operations

- Unbind - Close the connection, not the inverse of bind. But anyway this is to close the connection, so these are roughly some operations in lightweight directory access protocol.

(Refer Slide Time: 51:38-53:18)



Brief Review of Security

- Notion of *Security* for a network protocol is comprised of (at least) these axes..
  - Identity & Authentication
    - "Who are you and who says so?"
  - Confidentiality
    - "Tough petunias to eavesdroppers."
  - Integrity
    - "Did anyone muck with this data?"
  - Authorization
    - "Yes, you can do that, but no, you can't do that other thing."

42

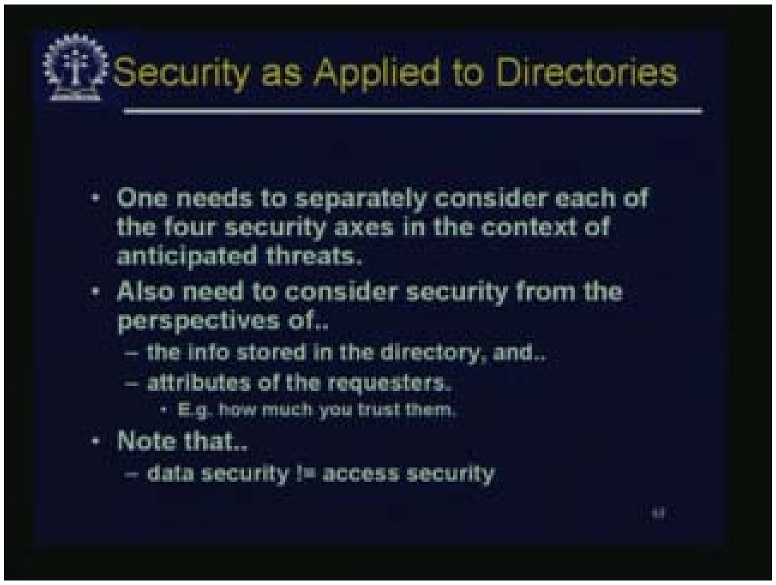
We will just touch on this security issue:

- Notion of security for a network protocol is comprised of at least these axes:
- Identity and Authentication:

o Who are you and who says so? So identify yourself and then it should have some protection against spoofing that somebody is claiming to identity which is false.

- Confidentiality: Whatever information is being passed around, other people should not be able to snoop into it so confidentiality is important. You might use some kind encryption for this confidentiality purpose.
- Integrity - Did anyone muck with this data? This means, did any one change the data? If there is a change which has done by some person who is authorized to do that you might want to keep a log or audit trail for such changes otherwise you want to be sure that on the way somebody has not changed this data. It may be necessary for some applications to maintain some signature kind of a thing to see that the data has not been changed.
- Authorization: Yes, you can do that, but no, you cannot do that other thing. This means there some organizations in access control user. These identity, confidentiality, nativity and authorization kind of accesses are definitely there.

(Refer Slide Time: 53:19-53:44)

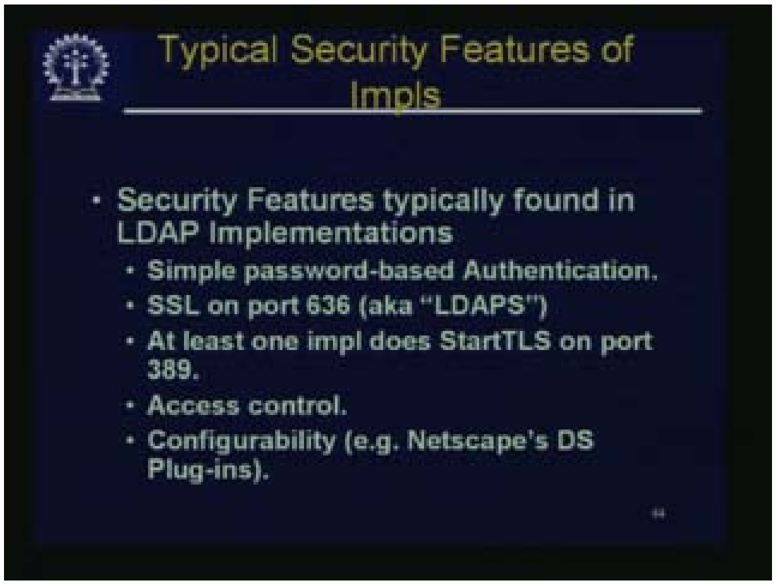


Security as Applied to Directories

- One needs to separately consider each of the four security axes in the context of anticipated threats.
- Also need to consider security from the perspectives of..
  - the info stored in the directory, and..
  - attributes of the requesters.
    - E.g. how much you trust them.
- Note that..
  - data security != access security

- One needs to separately consider each of the four security axes in the context of anticipated threats.
- Also need to consider security from the perspectives of
  - the information stored in the directory, and
  - Attributes of the requesters
- Data security is not equal to access security.

(Refer Slide Time: 53:45-55:06)



Typical Security Features of Impls

- Security Features typically found in LDAP Implementations
  - Simple password-based Authentication.
  - SSL on port 636 (aka "LDAPS")
  - At least one impl does StartTLS on port 389.
  - Access control.
  - Configurability (e.g. Netscape's DS Plug-ins).

- Some typical security features of LDAP implementations:
- Simple password based Authentication.

- SSL on a particular port 636.
- SSL is secure socket layer and TLS is transport layer security. This is on port 389
- There is some Access control
- There is some Configurability. There are other things that you can do with the directory especially in the context of an organization. Therefore because of spams and other issues the idea of actually generating a very global white pages for everything under sun, such a grand idea did not really work out but in the context of specific organization with its own security boundary and its own needs, LDAP is a well defined is a protocol which can be used.