**Linux Networking – Lecture 6**
**DFS**

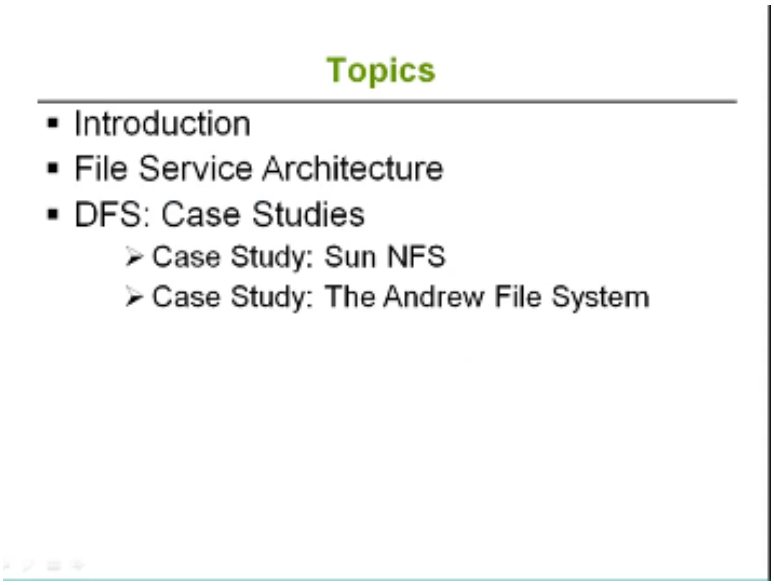
Hi everyone and welcome to the class the LPS class today we will be again continuing with the next networking lecture number six in the last lecture , we just introduced the DSS today we will be going on with the DFS the distributed file system in fact. I am going to actually will start the DFS discussion now today last time we just looked at some introduction but I am going to repeat that material again and before.

I begin I think again we had a good lecture on the DNS system we looked at the email blast and I am not going to recap that today if you have any doubts we can always take it up session of so let us begin the discussion on.

(Refer Slide Time: 01:08)

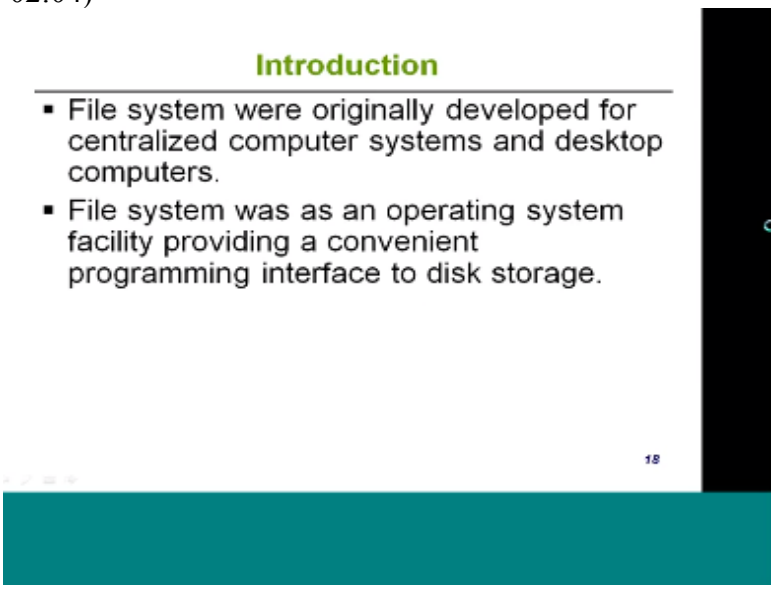

Go disputed file system again, I say I mentioned in the last class.

(Refer Slide Time: 01:11)



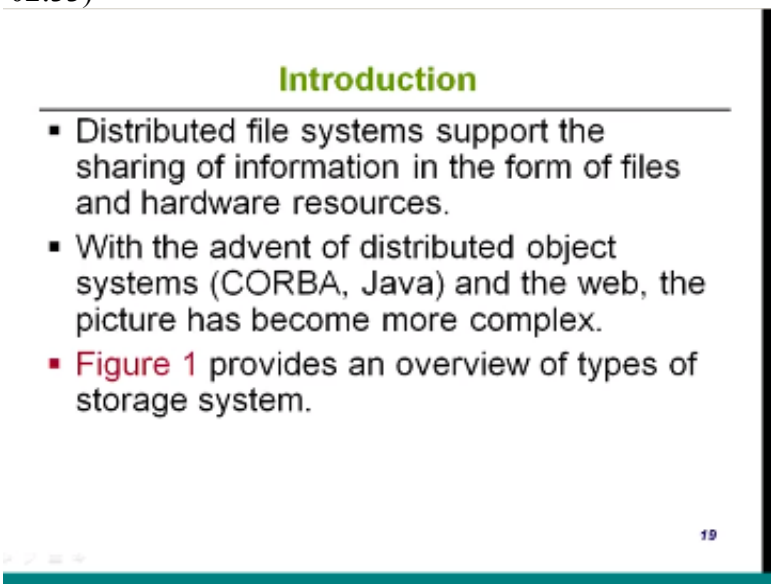

We will talk about the distributed file system what it entails why system big deal we will go into the architecture again ,you need to know like why it is a bit big again what are the issues with the file system what are the issues with the file system and also we will take up like two case studies one is on the Sun NSS system and then the other one is Andrew file system again now let us see like I mean how far we can get to but these are the topics that.

(Refer Slide Time: 02:04)



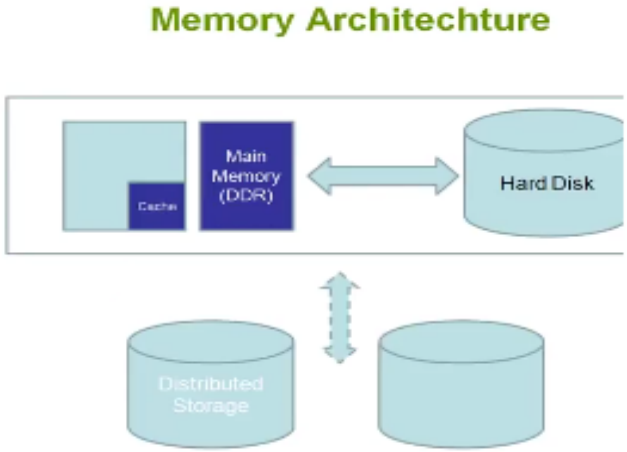

I want to call so last time we discussed the file system again the file system basically just it was done it was developed for a centralized computer system and the desktop computers, and file system is an operational operating system facility providing a convenient programming Interface to the disk storage so we will see like what are the gradations of file system or we have to understand the memory architecture before we go the file system.

(Refer Slide Time: 02:33)



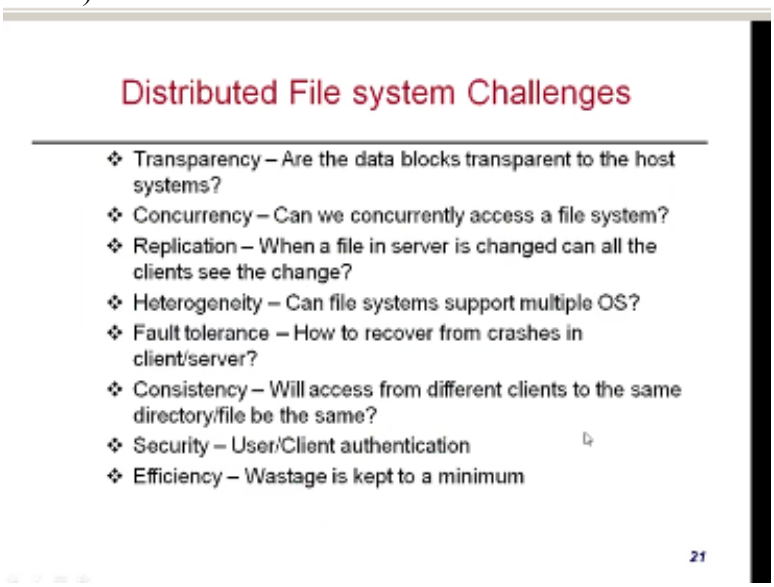

And a distributed file system specifically its support sharing of information in form of files and hardware resources across the network so that is the key thing and then once we have this build optic system like cobra or java in the web this entire thing basically like how much we are sharing and how the sharing happens has become like extremely complex so before we go into the storage system let us look at just the memory architecture.

(Refer Slide Time: 03:05)




I think, I mean this is something that you are familiar with so we have say like I mean this box represents the of the CPU and then there is a cache memory which is part of the CPU it is very localized and in fact you would not even know the existence unless you look at the on the

computer itself or basically like what it has in the control panel or any other place, basically typically these come in various sizes.
So right now we have one gate to gate we are measuring in terms of gate exactly then we also have the main memory this is an example will be like the DDR area 3 is very famous nowadays you get into two gate or four gate videos sometimes like even the caches like, I mean it used to be like in case now probably like in Meg's and gigs are also available and the memory of course like mainly in a heavy duty server it can go all the way.
I have to light one terabyte also then you have like hard disk which is essentially the storage there things get written and DCT it resides programs that the heart is before they get executed a distributed storage is nothing but so if you look at this particular box the box around it is a system or a computer and then basically like a distributed system is centralized for storage and that is shared between multiple computers or there are multiple disks that can be accessed by a single computer all through network.
So now I mean if you look at the hard disk the most this would be the storage finally they are all the similar stuff so what is a big deal in like using a distributed system, so let us look at some of the challenges form.
(Refer Slide Time: 05:19)




So when we talk about the distributed file systems there are many challenges that they need to overcome in order to make sure that we get we can reliably store and retrieve it one of the issues is the transparency, so as you know the data is written in various chunks basically so what kind of data in a hard disk is for transparent to the host systems is it all of them is it some portions of transparent to some of the systems and some portions are transferring to another system.

How is that divided and then when there is an overlap like what happens, so these are some of the things that we need to worry about because there are many systems accessing the same disk storage, so again that is one thing that we will talk about then the next issue will be the concurrency issue what happens is two servers or two clients access the same files and currently who gets the preference and why so these are the kind of questions that we asked about concurrency.

Then the other one is replication which is when the server the file in a server changes can all the can see the change immediately or do they have a how does that work so again that is another thing that we will talk about then the fourth challenge is the heterogeneity challenge again in this one is the weather the file system support multiple OS. I am coming, I am connecting with Windows wires and somebody else is connecting with Linux OS somebody else may be connecting with the IOS OS.

How does the system behave that sense and does it support all the races or supports some of the operating system how does it work then the other challenge is the fault tolerance challenge which is then the file system crashes, how does the client know about that or if the client crashes how does the file system react say like I mean you are writing to a file area and then the ten crashes now what happens, do that that is that does that area gets recovered or is it just all corrupted how does the how what happens.

Then the other one is consistency comes from the different clients then they access basically the if the a is that the what each client looks at from its perspective are they all in consistent view, so that is another thing basically that we will talk about and then the security is another control how do you authenticate a client or a user to a file system, and so that I mean he can only he can access.

We saw like some of the high-level issues regarding security in when we studied about the Linux basically next form OS there we talked about the read write access read/write/execute axis also does that word so here we will be that that challenge has been multiplied pretty heavy so we will look at that and then finally efficiently how do we keep wastage of the space to the minimum. So are these techniques that we can employ to do this what we do so again as I mentioned the classical example that is on manifest form or Andrew file system so we will be learning about all these things so before we go into that let us look at the main memory architecture.

(Refer Slide Time: 09:42)

## Introduction

| | Sharing | Persis-tence | Distributed cache/replicas | Consistency maintenance | Example |
|---|---|---|---|---|---|
| Main memory | × | × | × | 1 | RAM |
| File system | × | ✓ | × | 1 | UNIX file system |
| Distributed file system | ✓ | ✓ | ✓ | ✓ | Sun NFS |
| Web | ✓ | ✓ | ✓ | × | Web server |
| Distributed shared memory | ✓ | × | ✓ | ✓ | Ivy (Ch. 18) |
| Remote objects (RMI/ORB) | ✓ | × | × | 1 | CORBA |
| Persistent object store | ✓ | ✓ | × | 1 | CORBA Persistent Object Service |
| Peer-to-peer storage system | ✓ | ✓ | ✓ | ✓ | OceanStore(Ch. 10) |

Figure 1. Storage systems and their properties

Types of consistency between copies: 1 - strict one-copy consistency
√ - approximate consistency
X - no automatic consistency

22

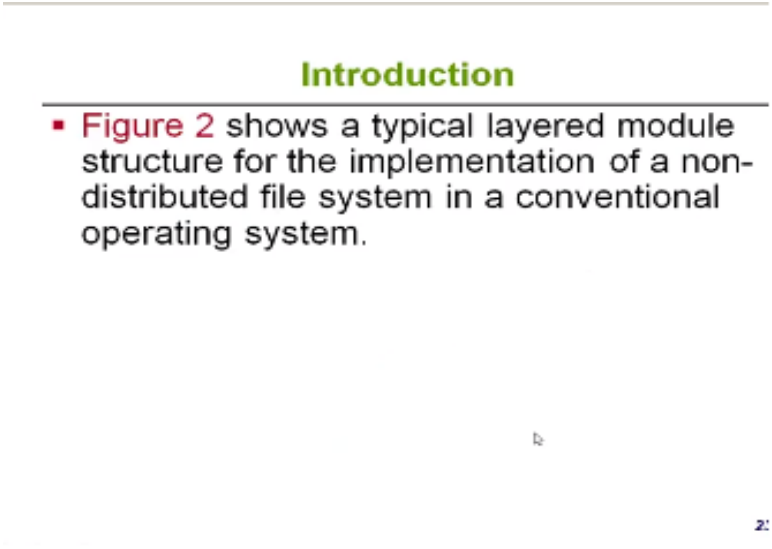
Which is essentially these are this in the previous slide, so let us look at it from the main memory onwards or we want looking at the caches and inside it whatever is inside the chip so main memory it is not the share it is not a shared memory and mostly like it is local to that particular machine or that particular CPU and the persistence ,it is not persistent, so persistence is another property essentially or the data whether then the process die whether the data still stays there or not.

So usually what happens whatever the program that is running in and process writes out the data into the hard disk so that the main memory is completely cleared of any of the remnants of that particular process , so what that means is basically the main memory more persistent method so we do not keep anything after the process gets executed then by nature main memory is also similarly is a very local localized memory.

So we do not keep any distributed cache or with us for that for more than the consistency maintenance again it tries to maintain the consistency and the example, essentially but it says is still one copy consistency ,so that you cannot replicate that copying across multiple memory systems here we call it mainland resistance example is the RAM or the DDR is another one so those kind of things now let us look at file system file system again like you do not allow the file system to share.

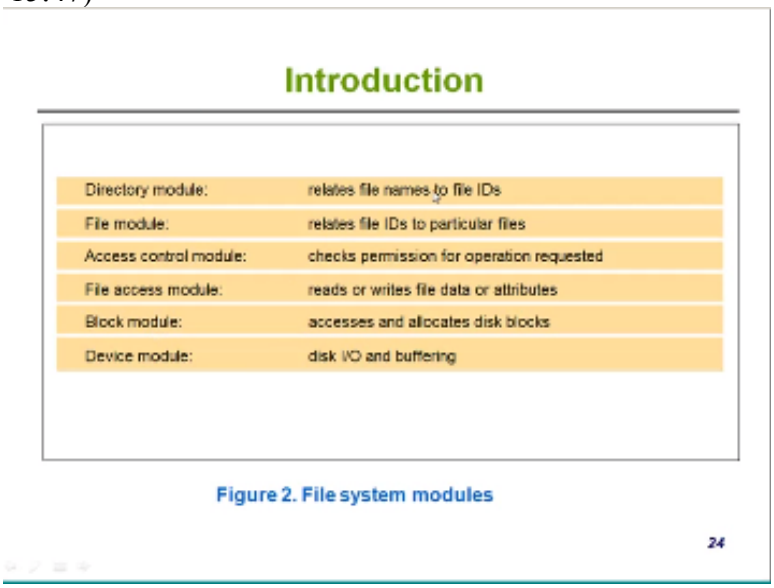
But it is persistent meaning the data gets written into that file system after the process gets executed again. I am distinguishing between program and process that this is the distinction that we saw very early in the course, which is essentially the process is a running instance of a program. I hope you guys remember again the file system also does not allow distributed or

cash-strapped because inherently like in with this file system we are talking about a file system within the computer.
So it is mainly for that particular machine and consistently maintenance is a strictly like one copy insistence a system now let us talk about the distributed file system this is this is where it gets interesting the yellow sharing in a distributed file system the data is persistent as we know and then it also allows the distributed cache essentially, so that you can create the because of data all over.
The place and finally the consistency maintenance is or will have multiple copies which essentially go through that consistency so we call it like approximate consistency so we know that the main consistency is how each of the plan will use at particulars files and here it is the it is almost there essentially some of the new information may not be burning both the file but pretty much like I mean the poppies are updated on a periodic basis.
So now let us look at a web server observer also allows you to share the data it is the data is persistent allow the distributed cache and the Pickers but there is no consistency essentially like if something gets updated everything is not looking at the same thing so one data may be old the other data may be new unless you take like refresh that they do not be the distributed shared memory is under one ,so here again yellow the share sharing automobile but the memory itself is not persistent.
The way that we allow them as being also through the distributed cache replicas and then finally the consistency maintenance also they are remote objects essentially this is something that the CORBA supports and again here sharing is done and then it is just a one copy consistent, so if you look at it I mean so the highest one is probably the distributed file system and then there is also the peer-to-peer storage system.
Which is also very similar but the others are kind of they are lower in the hierarchy as you can see the position object store and also the remote objects from the CORBA.
(Refer Slide Time: 15:25)




So let us look at the next picture this shows like a layered Module structure of the implementation of a non distributed file system.

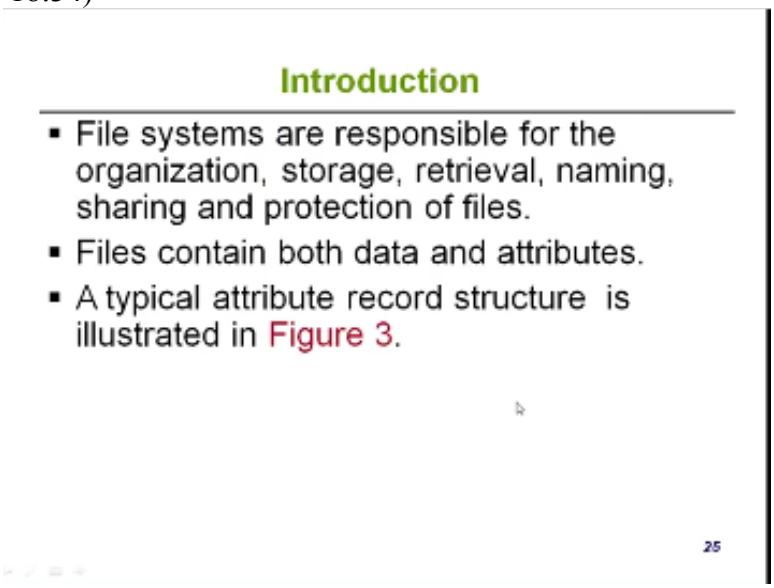
(Refer Slide Time: 15:47)



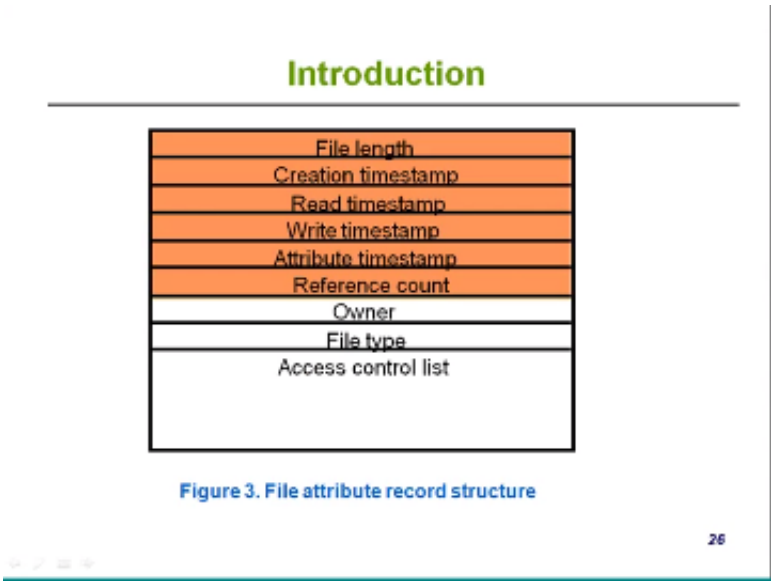

So here we have a directory module, and then a file module, then the access control, module file access module, a block module and the device module, so let us let us just look at what each the function of each other the directory modules is essentially like. I mean that it has all the file handles that are in that particular direction, so this is the each of the file IDs the file handles are also known as a file by ID.

So essentially the file names are related to the file ID with using the direction body the file module keeps all the file IDs and correspondingly like body the file where is it located and all those details the access control module is the permission checker, we already know that there are

three types of permission the read write and execute, so the access control model is the one that checks for these permissions and whether certain operations are allowed or not allowed.
You the file access module that essentially is used to read and write pile data all the attributes and the block module essentially like now accesses be the file system or essentially the disk and act that goes and allocates the blocks that are needed for this writing go file and then so basically like. I mean the plots are usually chunks of some number, so that veins on the file ,file type it allocates some blocks for the particular file.
In the membrane and then the device module is essentially like an email that is pretty much it is the module that accesses the disk delta use and also like that for buffering ,so whenever you are reading the file it takes the file ID is actually like an active form that particular disk and it gets the files for the through the buffers it takes to the buffers again.
(Refer Slide Time: 18:34)



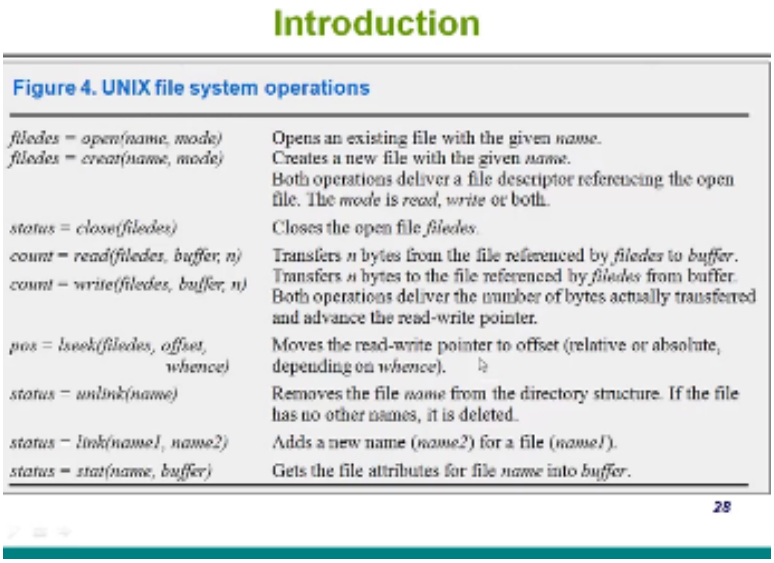

The file systems essentially like. I mean so that provides the organization storage, retrieval naming, sharing, and protection of all the files so a file contains the data and the attributes so now let us look at how the file records are retrospective or our file data structure reform.
(Refer Slide Time: 19:03)




So the key things that we want to write or the file length and then it was created a creation time stamp if it is read multiple times or the last and it where it is read retain time stamp then after creation when if it is written again then the right time stamp and then there are some attribute time stamps and then the reference count and then there are some additional ones which is the owner the file type and then the access control list.

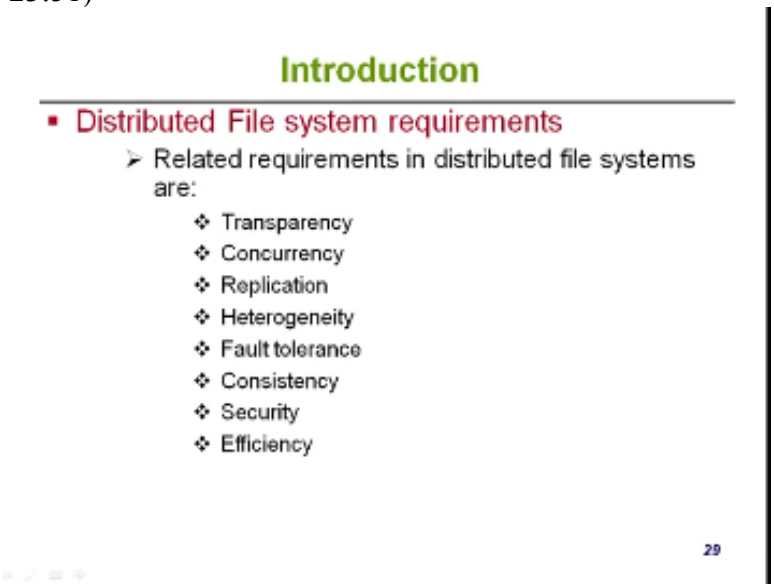
So this is typically the record that is kept as part of the file and we can actually access various these various form parameters to get to where what the values, so these are done through the operations actually.

(Refer Slide Time: 20:06)

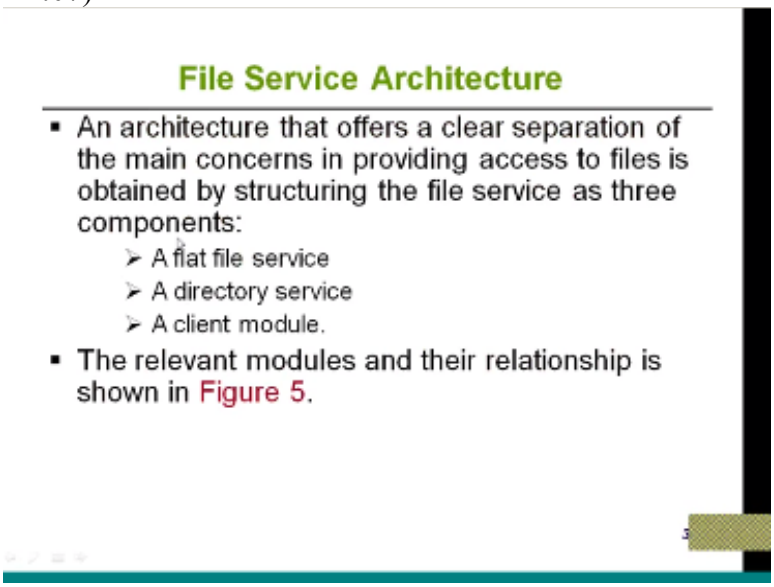



So let us look at some of the operations so here the operation is open is the name and then the mode, so it basically opens an existing file with a given name and then assigns the file ID to file

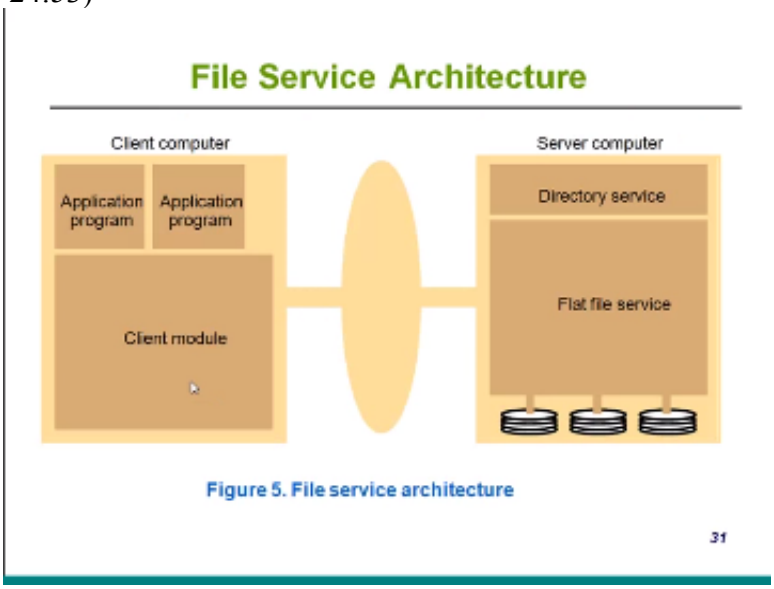
destination create his another operation with the name and the mode and this one creates the file with the name and that that ID is assigned to the File ID.
So the mode can be like read/write or read/ write and then close the file destination on the file handle closes that particular file in any of the open files, and then if it can close then it assigns the status as 0 and if it cannot hold then the status becomes one, so based on the return value we can decide whether the files not close or not in an account essentially like I mean this is so actually like.
I mean the operation is the read operation read file this buffer n it essentially transfers n bytes from the file to the buffer and essentially like we can also like measure that with the thumbs of the right again file destination buffer n will transfer the file actually like. I mean the N number of bytes are transferred from buffer into the file , so these operations essentially like be we deliver the files and they also advance the read/write pointer.
So the pointer is essentially like it is in the file, so that you know like I mean where exactly and then L C is the you give an offset and another pattern all events using this parameter it goes and moves the point of the read/write pointer to that offset and then unlink essentially like remove the file name from the directory structure if it has no other names it is just deleted otherwise it is basically look at this remove that name alone.
The other names are kept and link name one name to again this adds a new name 1 name 2 the file name one and then the other one is the stat command this actually gets all the file attributes we saw that here essentially this one the attribute count and then the attribute time step so in all the attributes are it is taken and then put it in the buffer.
(Refer Slide Time: 23:51)



So we already saw like, I mean to the requirement with itself like I mean we know that these are the challenges and that is what we so let us look at the file service architecture.
(Refer Slide Time: 24:07)

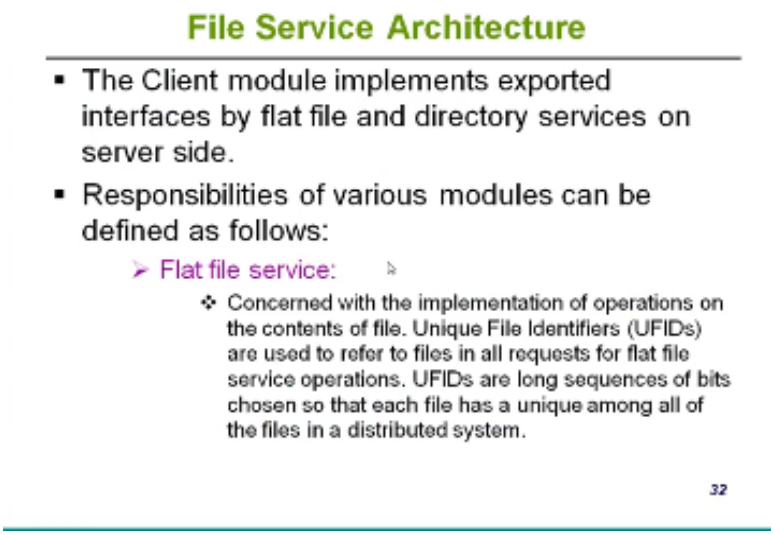



This actually offers a clear separation of the main concerns in providing access to the files and it is obtained by structuring the file service with three components, so there is a flat file service a directory service and the clan module service, so again this the five service architecture is a way to organize the services that are associated with the files so that things are provided properly so now let us look at the modules.
(Refer Slide Time: 24:53)

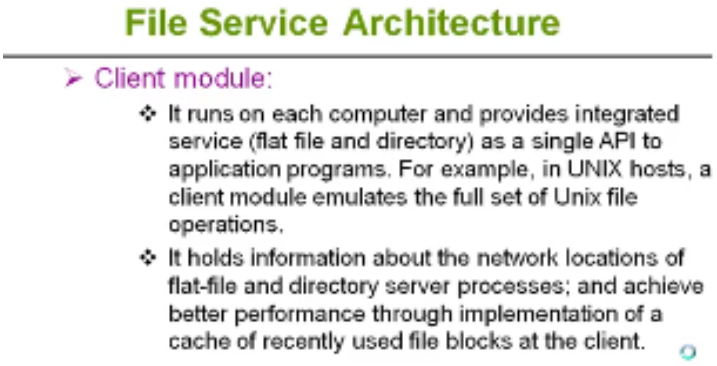



In the file service architecture so imagine the server computer sitting somewhere far away and then it is connected the client using this link is essentially like some kind of the network and there may be like more clients sitting, there which are accessing the same data and essentially

Like I mean so when the client tries to access the flyer flats a file service essentially immunity disassociated this and directly like a minute it collects the data and sends it to the client And it is almost like I mean you can think of your the can computer is accessing it as a one-on-one with interacting with the file services so there is a directory service and a fine service and basically like the flat file service collects that file information insert to the client.
(Refer Slide Time: 25:56)




So essentially in a flat file service all the files are assigned a unique file identifier or UF ID and so since the files themselves are uniquely named it is easy to actually obtain what the pilots and the essentially ligament so one system that is accessing one of the files the other systems may not be able to see every file has a unique identifier and the times need to know that unique files identifier in order to access that form So the clan there is a client module that exports the interfaces by the flat file and the direct restore services of the server side.
(Refer Slide Time: 27:00)

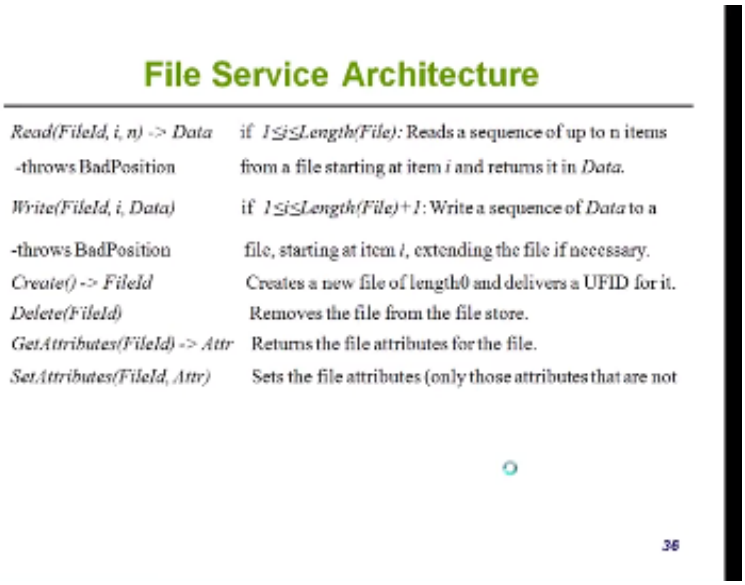



So the directory service itself it is provided it provides the mapping between the text names of the files and their UFID's so the client may obtain the UFID's ideas of file by coding its text name to the directory services and these are all like one-to-one mapping, so that that is the uniqueness of the flat file it is not very interesting and as you can see actually the UF ID can be in long sequence of bits because if we need to map all the files in the file system with unique identifiers it is kind of an odd just asking.

So the client module that is another one piece of the services that we saw in this picture this runs on each computer and provides an integrated service or a flat file service because this service is pretty much constant or same across all the clients , so the client is in fact in the unit side actually emulates the full service full set of UNIX five operations these are the operations that we saw like earlier like open closed and all those things all of the services.

Are provided by the client module it also holds the information about the network locations of the flat file and then the directory server processes so that it can actually go to those directory servers and access those five it actually better performance through implementation of a cache of frequently used file block at a client so we can improve the performance of client module by having a cache and local cache within that particular client in the system that we store the recently used by block again the issue will be how do we maintain the consistency and so we will talk about that.

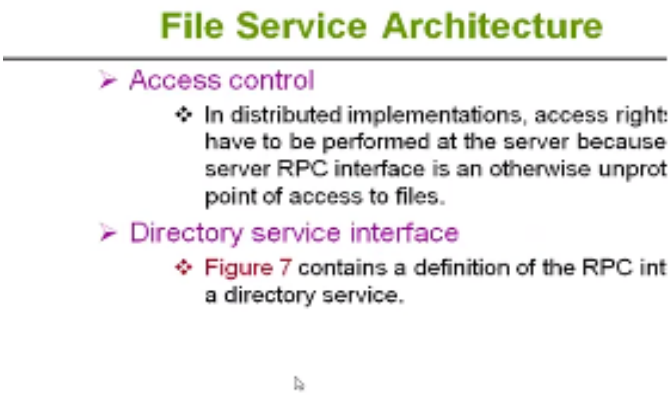
(Refer Slide Time: 29:17)




So here some of the again commands that we can use the read field. I for I, n and then that goes into data essentially again for - and arrow data here essentially like, I mean we read a sequence of 1/2 length of file and the variable is I which is essentially like the increment that and then basically like I mean and then essentially ligament that particular items are stored in data the right field.

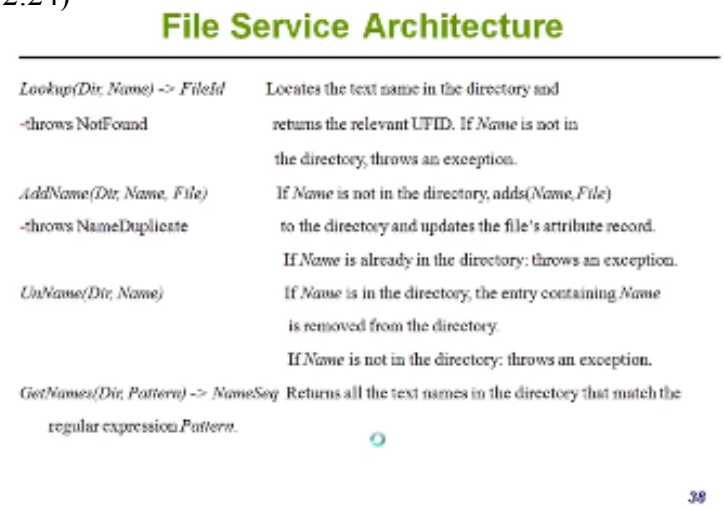
I data essentially it writes the sequence of data into a file starting at item, I and extending the file if miss and create is another command that creates a new file with link zero and delivers the UFID to the field variable and delete field this remove that file from the file store get attributes field and then store we can store it into another variable called attribute and then set attributes essentially like them and that is essentially Which are actually that we specify set to that particular field which is essentially file ID the unit ID that we have find out.

(Refer Slide Time: 31:24)



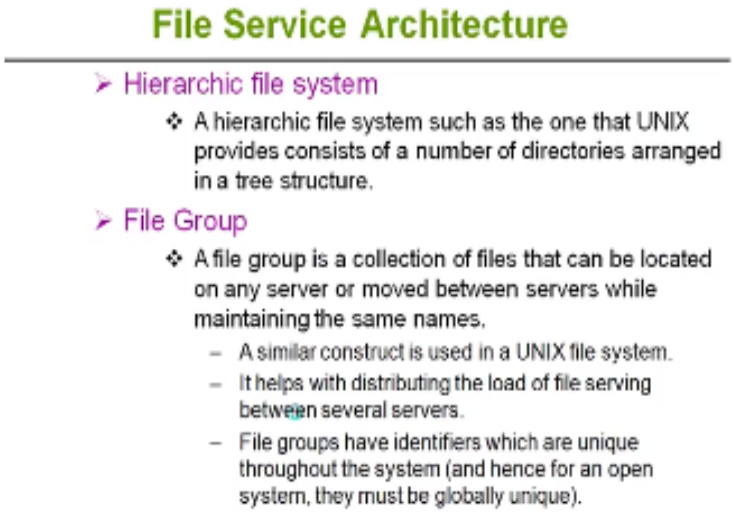

So again the access control , so in the distributed in implementation access right text has to be performed at the server um so it is not a client that provides the access control because the remote process control interface is unprotected point of access to these files essentially so until you reach this server basically like it is all unprotected and then they see be able so the access control itself is provided at the server side. Directly service interface eventually like that we will look into it now.

(Refer Slide Time: 32:24)

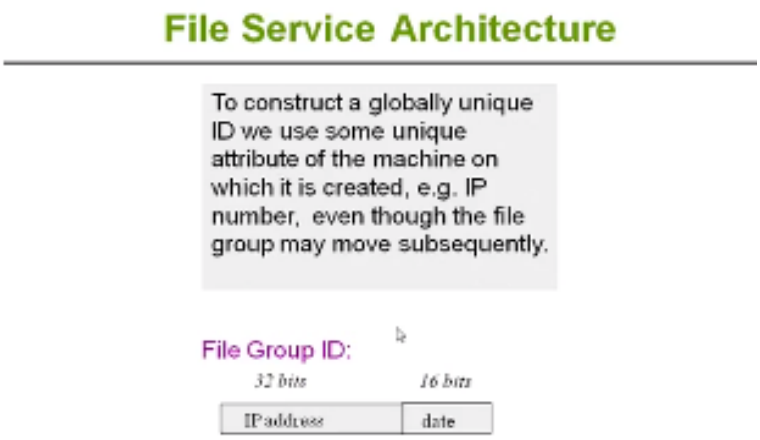



So here there is a lookup or particular directory with the name and that file ID and if the name itself is not there in the in that particular directory then it throws an exception, so that is the add name essentially like add the name and call the file into the particular direction and if there is a Already that may make this then it throws an exception , you unnamed directory name is

essentially like an aim is in the directory then it is removed from the directory if it is not there in the directory then expose an exception and then finally.
The get names will actually with pattern get the text names in the directory with a match of the regular expression pattern, we will learn about regular expression future modules in one of the programming modules and then that is assigned to the consumption so now we go into the next file service architecture.
(Refer Slide Time: 33:58)

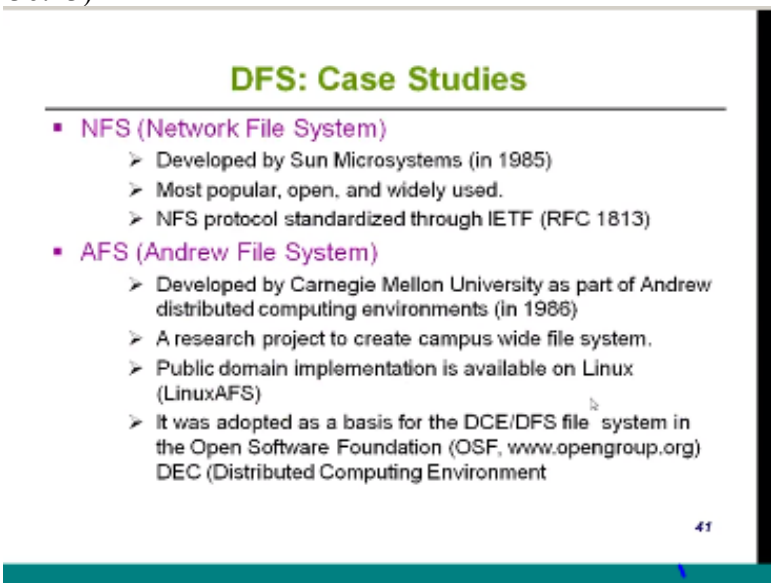



Which is hierarchical file system right now we look at this a flat file which is very easy to implement very easy to access but at the same time you cannot do a lot of functionality go and also you carry along unique file identifiers, which just causes issues by growing your directories and the growing the client modules now let us go into the next one which is the hierarchical file system this essentially like.
I mean is the file system where the directories are now arranged in a tree essentially so essentially you need to go into like various directories in order to figure out what is going wrong and then also the file group is another concept there it is a collection of file that can be located on any server or move between the servers while maintaining the same names.
So we saw some of these examples in me originally like started Linux a similar construct is used in more in a UNIX file system it helps with distributing the load of file serving between the service so essentially like, I mean you can move the file system or the file group to be Closer to where the clamp is are asking for so we can move to another service also the file groups have identifiers Which are unique to out the system so that is that is the way that it can be moved from one system to the other.
(Refer Slide Time: 36:02)




So here is one method to construct a globally unique ID so we did use some attribute of the machine on which it is created so basically like the idea so we know that idea this is the 32-bit or a for ruptured binary we just add a date to it like so the IP address followed by date we call it as a unique identifier the date could be a date and time all the way to the second then the file gets printed.

(Refer Slide Time: 36:43)




So now we go into the distributed file system again the three hierarchies that we talked about are the three different types of file system that we talked about flat files a hierarchical files and now we go into a distributed time system, so here we will talk about two main file systems one is the NSS or the Sun and this is the this was developed by Sun Microsystems in 1985 it is the most popular and it is open file system and it is used pretty much universally today.

You also the NFF protocol itself is standardized through this particular standard is known as the RFC 1813 so the main idea is the here is basically the serve the server or the file system itself is a stateless system so essentially like. I mean the server itself is not required to remember anything so it does not have anything in the memory essentially the things like which clients are connected.

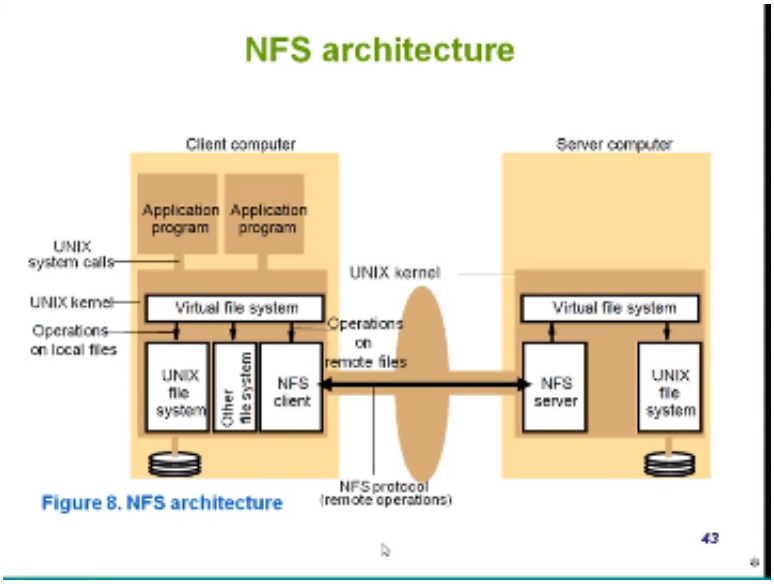
Which files are open exactly so essentially let me just the file system itself is sitting in one place the signs basically are send some requests essentially with all the information that is needed and then basically the server just fulfills the request and then this forgets the minute its root is the request and then perform, so the onus is on the clients to provide all the information to complete it or accessing a file or essentially to provide it to get that service.

So the whole so the advantage of this kind of a system is that the server state does not even does not grow it more number of things, so there is no change to the server eater which you need to change the number of servers you and then the other key idea of NFS is when you perform an operation and if you repeating the operation you get the same result there are no side effects so essentially like.

I mean if you say like A=D + 1 and then essentially like every time you do the a basic A it is always it you get only like A= B + 1 or you do not get A=B+1 norm there now the state of A is different so that the next time you when you query it is it is different the actual the second system that we study is for the af-s of the annual file system this is developed by the Carnegie Mellon University as a part of the Andrew distributed computing environments in 1986 so you can see that actually they are fairly close.

The this research project was to create a campus-wide file system and basically a public domain implementation is available on Unix it is called Linux a so this was again also adopted as a basis for the DSS system DSS file system in the open software foundation and the distributed computing environment essentially.

(Refer Slide Time: 40:59)



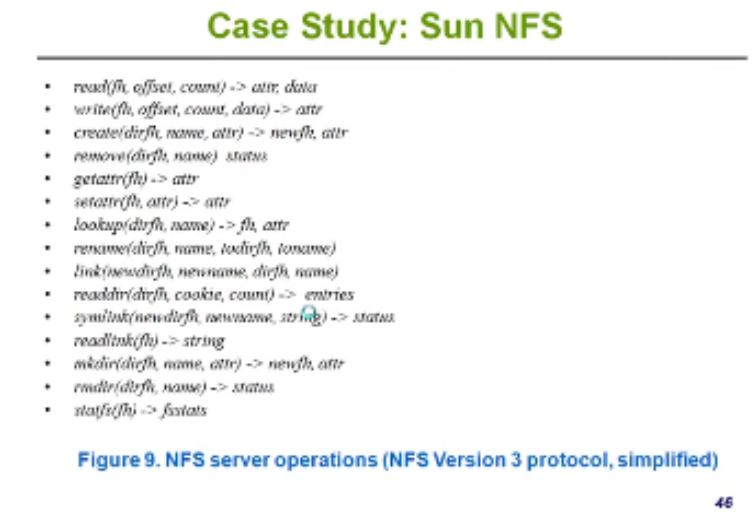

So let us look at the NFS architecture the NFS architecture consists of again the client computer but, I mentioned the client has a lot of onus on providing the data so let us look at first the server computer server computer has nothing but a virtual file system which has the rituals the unique the UNIX file system and then it has an NFS server which is the another program that serves the various clients and then in the client computer again like. I mean you have the same kind of things there is a virtual file system which has the UNIX file.

System and then whatever the extra ones here basically the blank ones are various devices slash deaths and then there is also an NFS client the NSS client talks to the NFS server place again provide the information as what the files that it responds retrieve or to write the arm write the content into the file they all these programs actually longer all decide on the Linux all the fun you so the communication between the client and the server are using RPC.

What or what is known as the remote procedure calls essentially so the client itself basically it is a it is a transparent access to the NSS file system so then you go through this NFS climb to the end of the server since it is stateless basically link sizes like this is now connected together and there is more NFS server and then a plan and basically exists you can transfer only access whatever is inside.

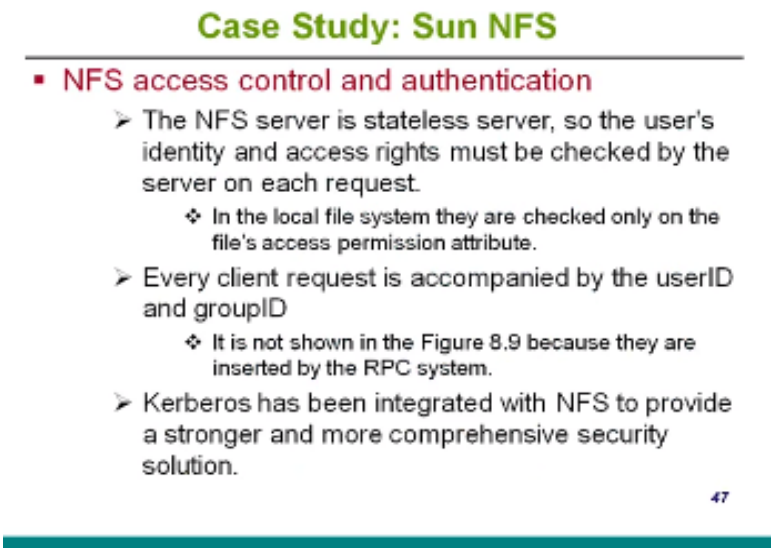
The UNIX file system server side as well you so the clients job is essentially like going to basically provide this file system and also the transfer and access to the NFS it also provides some virtual nodes essentially, this is essentially like procedures for procedures on an Individual file or it is an interface or procedures on an individual and then basically to translate that the individual file procedures into the various NSS.

Remote procedure calls so that rail again mate can access the NFS server and retrieve that one so here essentially like the application program provides the unit system calls through the UNIX kernel and essentially like. I am in the operation zone, so they all the virtual file system just keeps all the operations as a local operation so we would not feel that actually well there are some remote eyes also.
So from the application side you are just calling a local program but the virtual file system identifies which is the local and this is the remote files and then the remote is the channel through them client and then it provides the this RPC into the server to retrieve that file.
So the finite in fact we already introduced this concept of a file handle the file handle is the file identifiers about views in the NFS itself, so here in particular file handle denoted as apex could be the file system identifier the node number and then the anode generation so all the all of them are combined together to get to the I handle.
(Refer Slide Time: 45:44)



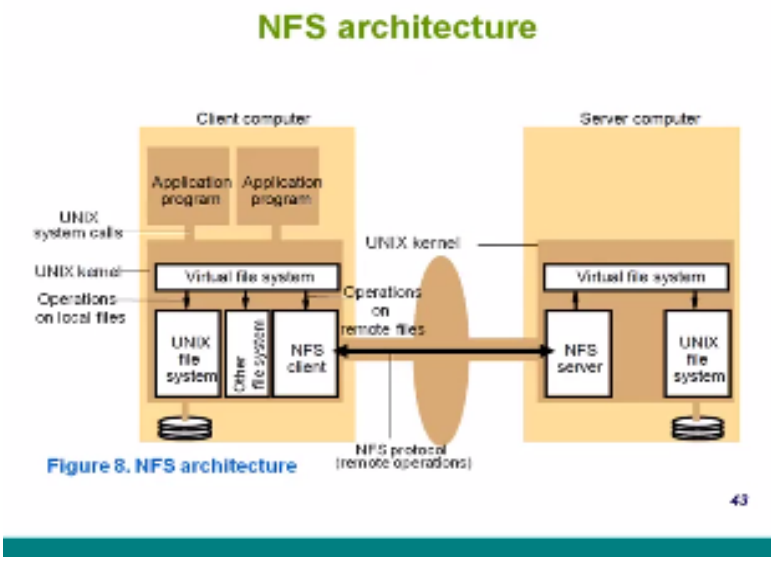

So the various commands are used here so in this one there is a read operation we again like very similar to the flat file axis or give the file handle or the file item the offset in the count and then that retrieves the data and put it in the data variable, the right for writing VCT we do again the file handle the offset the count and then the data and then basically like a minute is written into the file system and then basically like this one actually passing system.
The create command essentially where the now we specify a directory file handle the name and then the attribute and this command returns a new file humble and also some attribute on the pipe and then the remove actually is removed just return the status until a level blogger mode or

not again we give the just the directory file handle and then the name pending the gate attribute file handle gets the attribute related with that particular file.
Then the set will set that then there is a lookup command essentially which should return to the file handle and then attribute rename command it is basically the original name of the from directory pile handle and the name to the to directory and then to name link provides a link to the existing file then the reader essentially read the particular directory and then returns all the values available in trees.
Stream link is a symbolic link G or a soft link which ties a file to another end you name and then that returns a status the read link file handle essentially like that returns a string essentially so it gives this bag is the expanded link pack make the is it creates a new directory and all that it gives you the new file handle on the box and then our under base tables a directory and then staffed file system file handle gives the file system status.
So these are some of the commands that are used in the thumb no.5 which is one of the very popular systems.
(Refer Slide Time: 49:21)

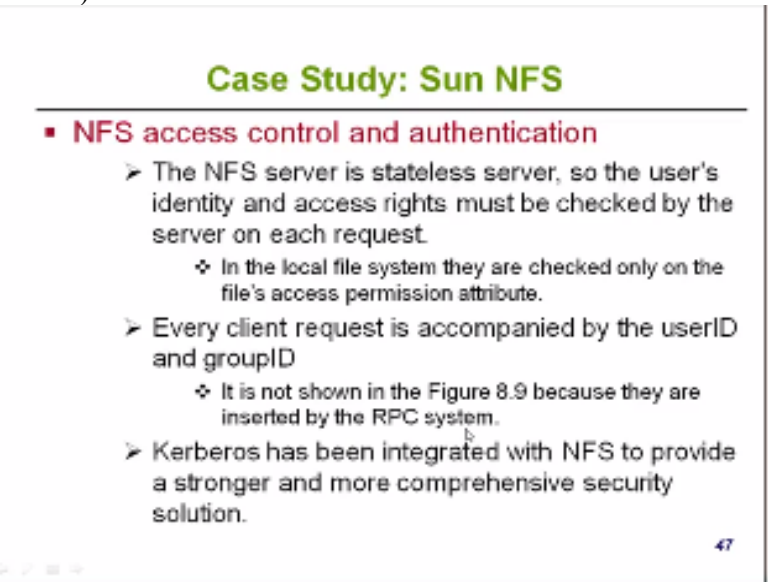



So now let us look at some of the other features of Sun NFS in fact we learned about this one is the security or how do we make sure that there is an access control and open easy so as I mentioned earlier the NFS server itself people state level which means that the users identity access right all them are controlled by the server only on the requests and essentially the client is so the local file system they are checked only on the files access permission attribute.
So essentially like I mean this is oh we have to check every time whenever we access that so it does not remember like I mean a particular user is authenticated or not so every time you need to check again and again also the client request is accompanied by a user ID and the group ID these

are inserted by the RPC for the remote procedure call we talked about this also like earlier like with some example.
(Refer Slide Time: 50:43)

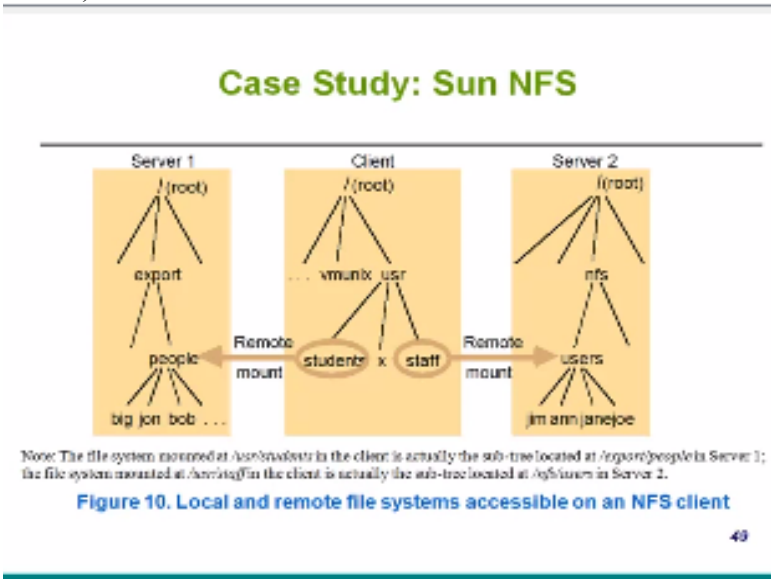



Where the virtual file system converts or basically like it forwards the remote access to the NFS client which converts the file handle into the continually remove resist ball it contains this additional data.
(Refer Slide Time: 51:01)

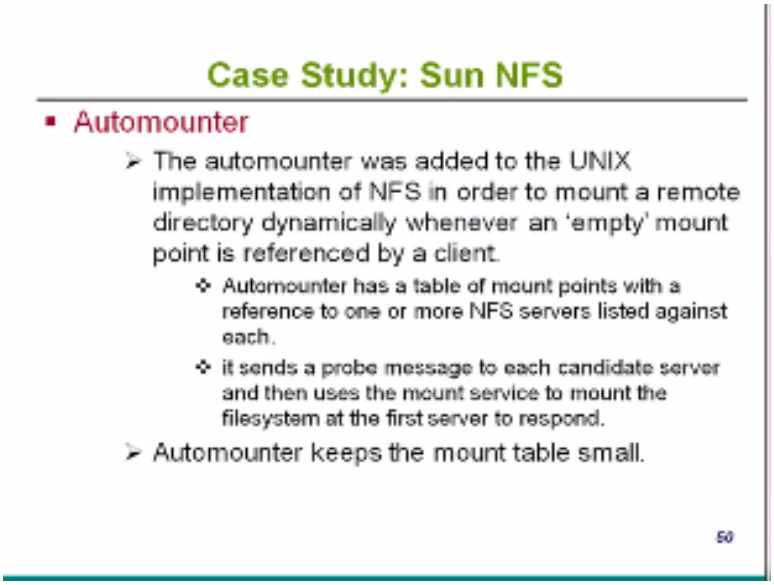



Which is the user ID in the group ID and then it sends it to the file system to gather the phone so the Kerberos is also it has been introduced visually integrated with the NSS and that provides a stronger and much more comprehensive security you but now let us look at the mount service mount service is used to mount a particular disk into the file system so if it is a new disk we use the amount function to do this to move on the remote files so the operation is mount remote goes

remote directory in the local between so it basically like bounce that file system into the local directory you.
So in this example the server maintains the table of clients which ever once mounted the file system separates all each client maintains a table of mounted file systems holding the IP address the port number and then the 500 the remote file systems may be hard Mobile mounted or soft mounted in a plan completed very similar to the soft link of pardon so here another example it shows the two remote mountain piles you.
(Refer Slide Time: 53:04)

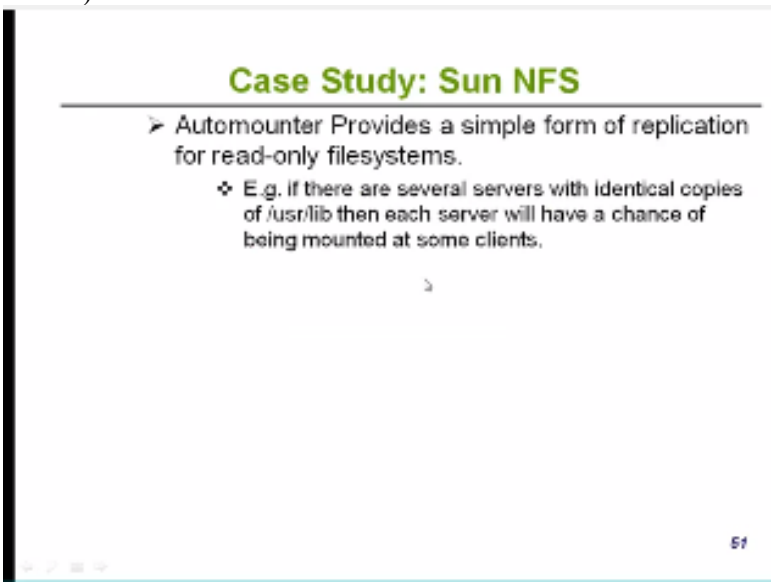



So in the server there is a directory called people which contains like Big John and Bob in the time side there is a directory called user it has students which are remotely mounted on people and then there is also another practical tab which is a remote human form starting actually in users and in so to you.
(Refer Slide Time: 53:56)



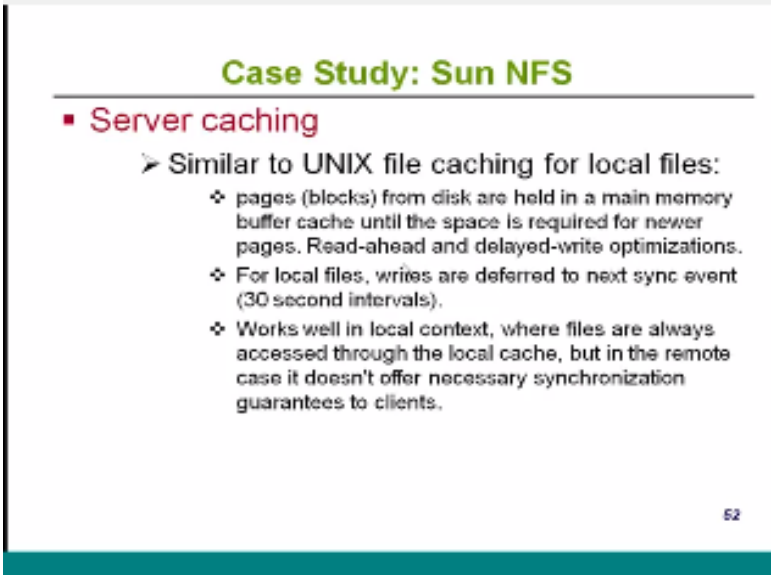

Then there is also a concept of Auto monitor the Auto monitor was added to the unit's implementation of NFS in order to mount a remote directory dynamically whenever an empty mount point is reference by a client so this is kind of the plant ask for a particular node which Is empty by default then the auto mounted automatically moves for the particular remote Directory that you so again the auto mounted has a table of mount points with the reference to one or more NSS service listen against each so it sends first a probe message the each candidate server to see whether the particular one point is still available then it uses the one point to mount the file systems. And then serve that apart and usually the auto mounted keeps the bound table to be very small but it also provides.

(Refer Slide Time: 55:12)




The form of replication or read-only file you so let us look at that caching essentially.

(Refer Slide Time: 55:25)

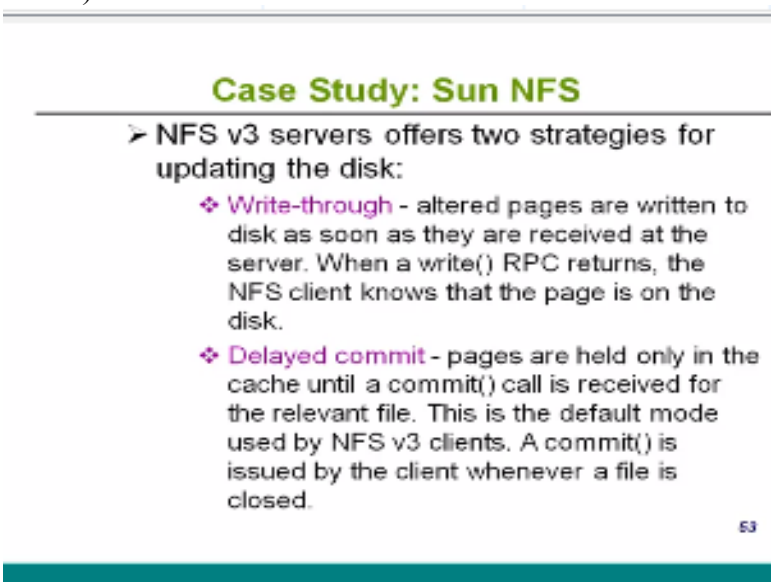



Begin to caching convene of two forms one is a server-side caching and then also all the other one is client-side caching before we go into the server-side caching let us look at the client-side caching the caching always is it provides improved performance because certain blocks if you cache and basically goes up we continue log and the client requests that every time the cache can provide very quick access to that video blog so that the system that is requesting that is not starving or data so for reads essentially ligament or the always the protocol from the client side is checks with the local cache before goes into the server.

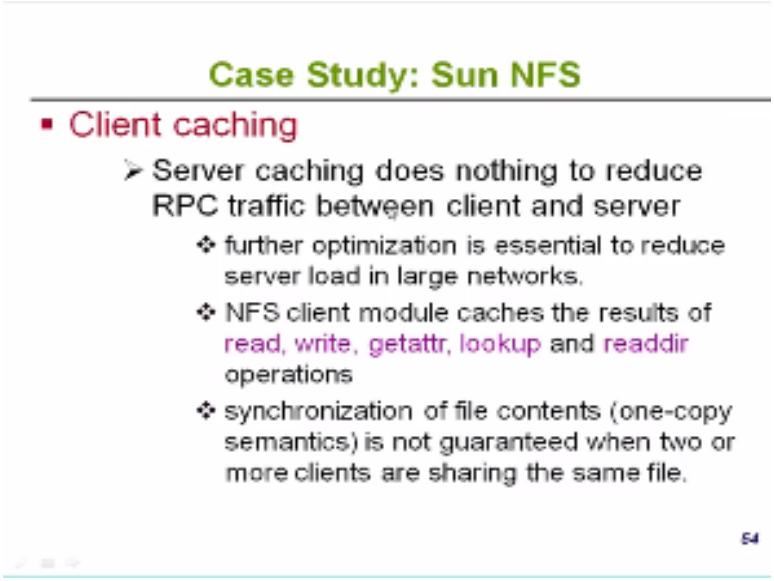
Because whenever we go into service it can take a longer time but as the local cache will be easier so only there is a cache miss it goes to perform but at the same time right basically you can the most or the Sun and express itself provide the periodic right back and not an immediate right back to the server mainly the reason is we do not want to contact server that often because that can slow down the communication you so the client cashes two types of data the one is the data block itself and then the other one is the attributes now let us look at the server caching basically so the server caching is very similar to the UNIX file caching in the local form.

So the various blocks from the disk they are held in the main memory buffer cache until the space is required for new things the read ahead and the delayed write optimizations are also possible so what this means is basically so if you know that actually you are grading from one particular file instead of getting the portions that are in the read operation you get more thinking that maybe like I mean you can read ahead of time so and then delay write is essentially like I mean so we delay the write with either the various cache consistency protocols.

And so that we can further optimize overall caching for the local files the rights are deferred to the next thing even so this is again the delayed right principle so 30-second intervals then they think even happens if the way I mean the Sun NFS works very well with the local context where I shall always access to the location but in a remote catch a case the synchronization legate does not guarantee the necessary implementation to the clients.
(Refer Slide Time: 59:08)

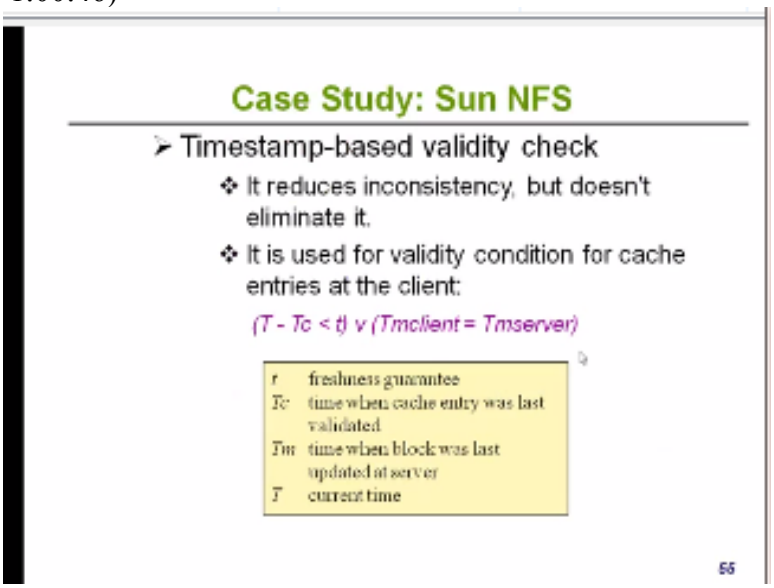



So the NFS version three service offers two strategies for updating the if one is the right group and then the other one is the delayed coming I think that you already know about this when we had studied the cache coherency protocol or to maintain prevalence in a cache. Whereas the right to is essentially making writing it both in the cache as well as in the main file in this case essentially like I am in for the altered pages are returning to the disk as soon as they are received at the server. So then the other one is the delayed commit where is the cages are held in a cache until it emits signal arrives and then basically at that point the file system itself is written with the new beta. You so we look at from this we designed the caching.
(Refer Slide Time: 1:00:19)



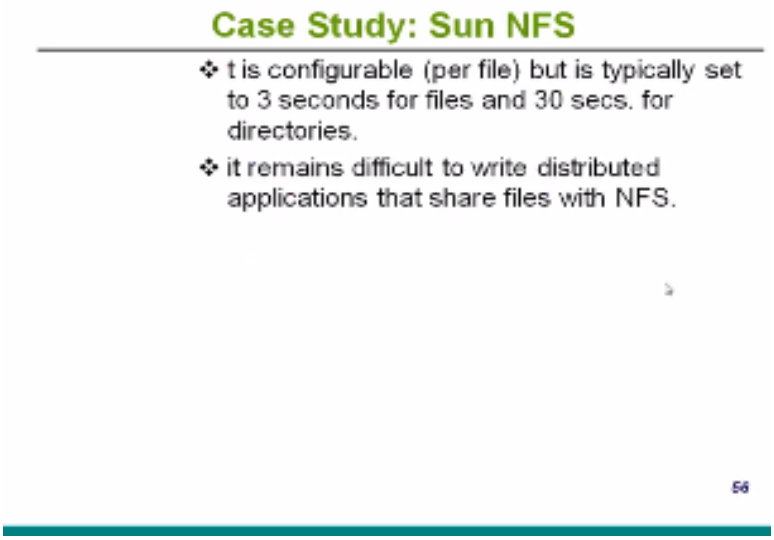

So again the client cache of the results from read and write get attribute lookup and then the reader operations the synchronization is not guaranteed when two or more clans are sharing this.

(Refer Slide Time: 1:00:46)

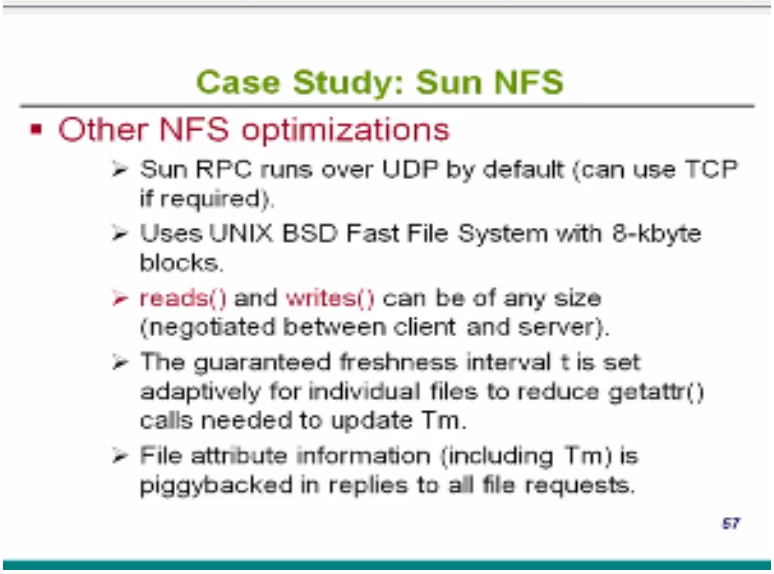



So that is something that we paid apparently for then the validity itself is checked to time stamp based check essentially like it reduces the inconsistency but it does not eliminate it so it is used for the validity condition or cash increase at the Klan. So the formula is like T which is the current time - TC is the time when the cache entry was last valuated must be ≤ T which is the freshness guarantee so we do not want excessive time pass between these trees and do anything and then essentially the PM client is the time and the block was last updated as a server and then TM server is essentially like them.

So actually TM client is the time in the block with lots of data that we client and then the same thing. When it was a physical server TDC is the time when the cache entry was last validated so basically like I mean we need to make sure what see the validity condition that is form satisfied or the tension face you so in the previous one the Tito system if the freshness guarantee that can be customized so you can decide what that value should be and then based on that we can construct the rules.
(Refer Slide Time: 1:02:40)

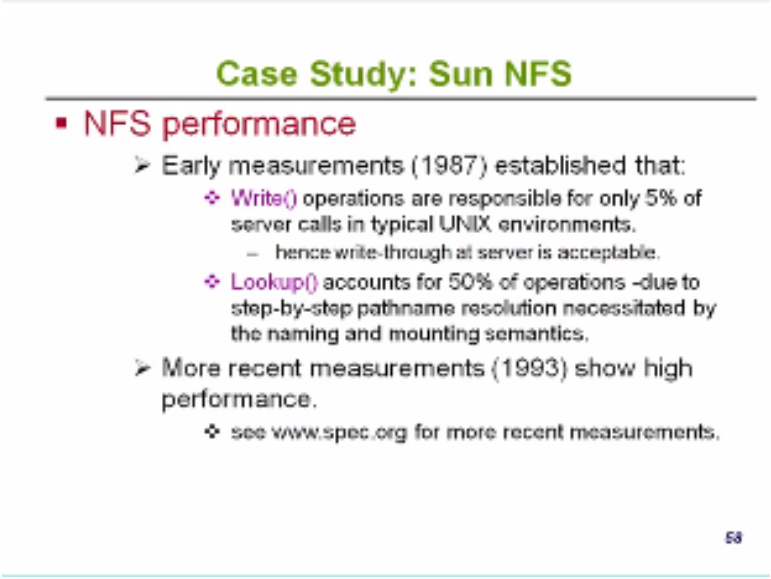



So the tea is set between like three seconds to 36 you also it remains that difficult to write distributed applications on the share files with the NFS.
(Refer Slide Time: 1:03:11)

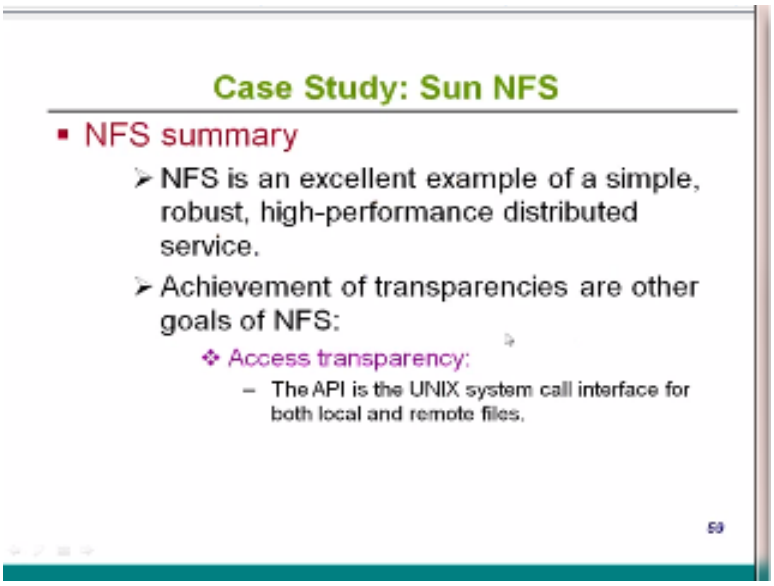



So one other question that may come up is how do we maintain the how do we updates or how does client update the flavor so for files essentially legman we can do the right back on the clan

cache to the server and then we can decide that interval of 30 seconds also like there are commands called flush on clothes which is essentially it takes the memory deep then writes and then push it into the file system automatically or the directory is essentially like media simply white to a server so as an example.
We can sail the tent X and Y they have a file name called A that is cached and then the file name A occupies the blocks one two and three. So now the clients X and Y both can open A and then X rights to the blocks one and two and then kind Y actually like now it breaks to the block and one block one and 30 seconds later what happens the kind Y reads octave and forty seconds either go pay my weeks block one so again like I mean this is one scenario making very that kind of whole system will behave you.
(Refer Slide Time: 1:05:05)

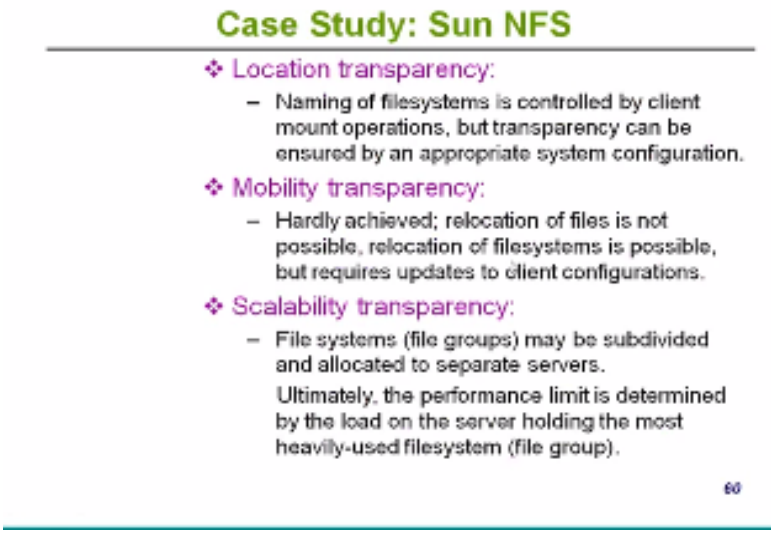



So the performance itself is angle on the parameter or another concern that we talked about so the right operations only like I mean responsible 5% of the server cause the typical unison moment the lookup accounts for 50% operations because step-by-step our name resolution necessitated by the gaming and the mountain semantics. So the recent measurements show a higher performance of an NFS and then a specimen was taken in.
(Refer Slide Time: 1:05:46)



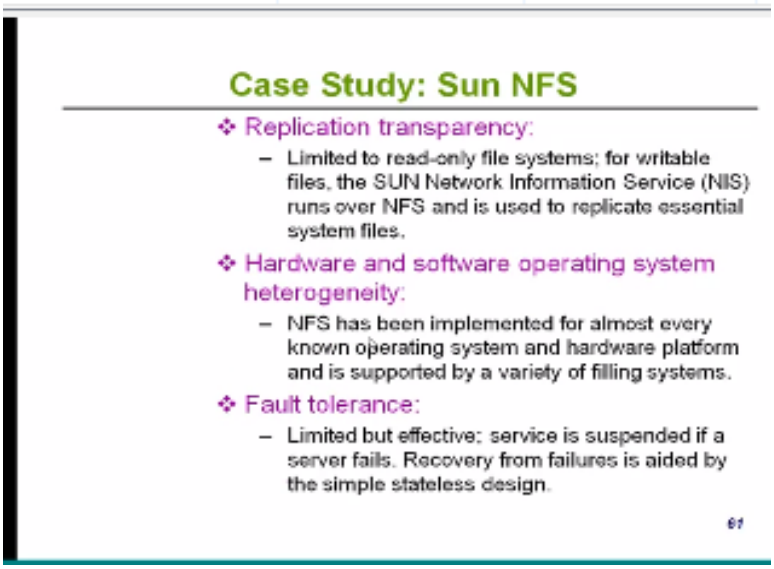

So in summary NFS is an excellent example of a very simple but a robust high-performance distributed service or dispute file system so the access transparency a scintillate woman for the same eunuchs all is other units colleges is same for both mobile and more ads so that is one of the we talked about transparency so let us look at those transparency in detail.

(Refer Slide Time: 1:06:23)

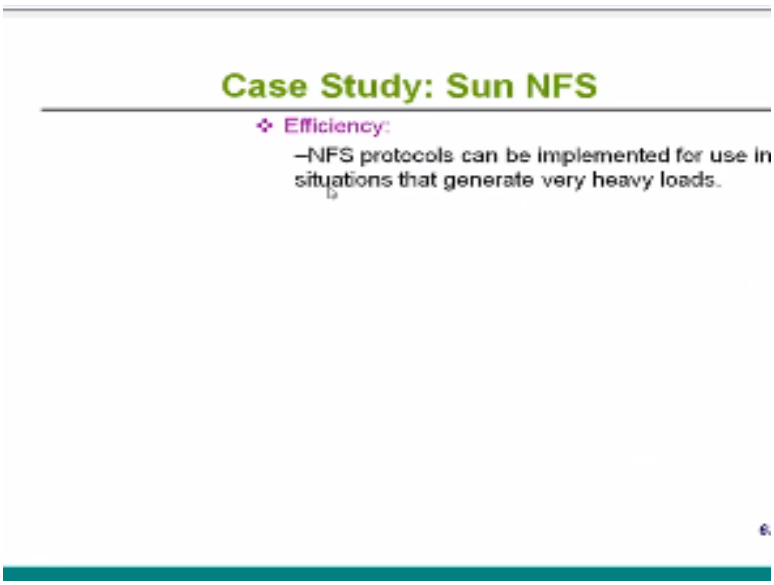



So the location transparency the name you know the file systems is controlled by the client mounting operations but the transparency can be ensured by the appropriate system configuration so even though like I am in the client actually controls the mount operations and how to name the files the transparency can be control if you actually configure the system if you have proprietary information or you need to nursing the bond points the mobility grants transparent Athens means that when the system is changed from one tool until the other how do.

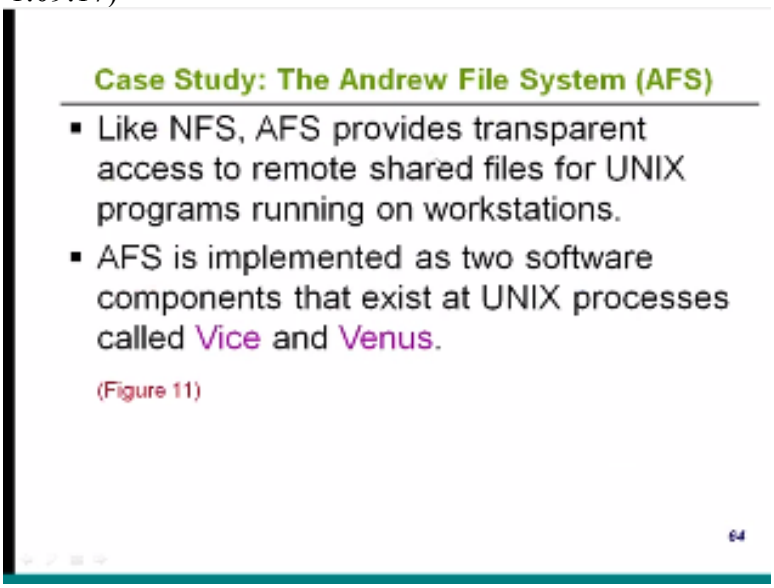
We ensure that thanks grace transparency this is not achieved at all and relocating the files is not possible and only the file systems are possible but that requires an update to the time configurations because clients are the ones deciding everything about the this particular file system the person we love and then the scalability transparency essentially will again improve again this one we can subdivide the file systems and allocate and we can allocate separate servers for each of the file system so that rail again we can scale the but it is also depends the Performance itself is determined by the load on the for all holding the most heavily used by replication is another one.
(Refer Slide Time: 1:08:04)



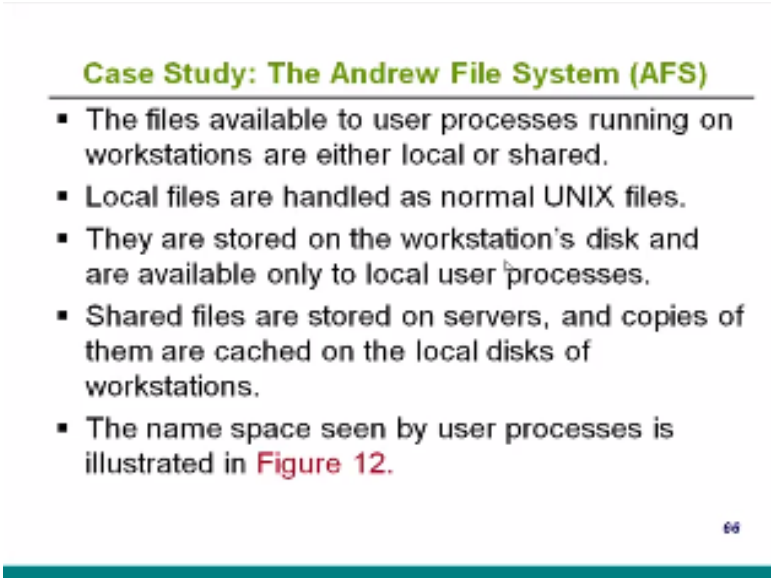

So for replication since we do a limited only we limit this file system to read-only assistance basically the replication transmittances so for writable files the Sun Network Information Service or the NIS one power method and that's used to replicate essential system the hardware software operating system heterogeneity again the NFS has been implemented or almost every known operating system and hardware platform and that is supported by a variety of home buying or billing systems fault-tolerance it is limited but effective so service is suspended the server fails victory from failures is aided by a simple state.
(Refer Slide Time: 1:09:06)




And efficiency basically can be implemented for use in situations that generate a very heavy so the next case study will be the Andrews file system.
(Refer Slide Time: 1:09:17)




Before we go into that I also want to talk about a little bit on the Sun NFS file system so the key takeaways that you want to take away our number one is it is a stateless server and then the operations themselves are what is known as the idempotent operations what I mean is here you Oh you so you.
(Refer Slide Time: 1:11:06)




Yeah and then the other takeaway is also the client. so you so yeah so the idempotent server operation is essentially like I mean when you repeat an operation it does not have any side effects so it also helps with the ways other things basically like the fault-tolerant the scalable performance the consistency. They are all addressed in the NFS system and then one thing to notice then particulars system crashes it tends to slow the other the server to the other clanks the tenses are oh there.

So this is kind of one of the drawbacks of the system people Inc whip it and then the client also like needs to cache the data or scalable performance not just the server alone needs to pass in fact in server caches sometimes it is not and since we put a cache in the landside the data consistency is extremely home because now we do not know like in which copy the latest form because there is some other copy that is still sitting in the hospital so that's all I have for today we will continue from this point next week then we are doing the next pass ion next class thank you very much.