Digital Switching
Prof. Y. N. Singh
Department of Electrical Engineering
Indian institute of Technology, Kanpur

Lecture - 36

So, we were actually looking at the sip reply actually last time and we have looked at that. So, request and response these two things, actually technically was actually was covered, but when you actually the abstraction the way it is being defined in the sip RFC we do not define everything in one single go we define it in form of layers.

(Refer Slide Time: 00:45)

So, the lowest layer which is being specified is how basically the syntax in encoding of all the request in responses that is the first and foremost thing. So, this is nothing but, syntax and encoding, of all kind of possible requests and all kind of possible responses. So, there is a generic definition and this is can be done through a grammar and this grammar is known as BNF something very similar to ASNF syntax notation, this is a thing let me write down the proper spelling.

So, this is the formal way of a specifying the syntax, I will give an example of this we will specify in our own way. And most likely that is BNF and once you have this is specification at least, you know how to form a request and how to form a response, for every request; there can be multiple requests there can be multiple responses which are permitted. So, once this abstraction is clear. So, at least you know the message will be

from. This is like, how to write a letter, there will be dear Sir, there will be address, there will be date.

Then you will write, how are you, whatever it is then you will write whatever you want to say after all preamble, then you will say thank you, good bye; your name, it is a format it is a syntax. You will be using English language or you will be using Urdu or Hindi that is kind of encoding how it will be depicted in text. You might write in roman for example, write in actually roman is script, but write in Hindi language that is possible. So, we do that actually once in a while. So, all that it is specification is given by this.

So, once the letter is being written close sealed what has to be done with that. So, next layer is talks about how it will be transported. So, next layer will be transport. So, this is the lowest1. So, next on top of it. So, layers are like this. So, whatever functionality abstraction or whatever it is provided by it is need not be known this guy. This guy just passes on the information and this takes care of it. Actually this is the way it is going to happen is, whatever you are going to generate with this BNF grammar specification the message, this will passed using a transport.

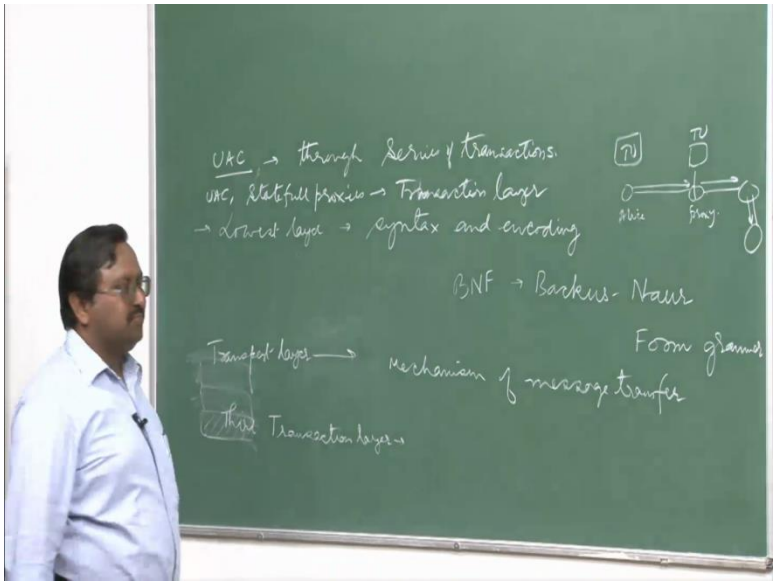
So, transport layer can be UDP, can be TCP, can be STTP or can be anything is basically message being encapsulated and then being transported over to the other person. So, it defines that particular transport and similarly how the responses will be send back. So, this can be any 1 of those, that this is basically mechanism how the message will be transported.

So, this how the message is found and this how the message is transferred from 1 entity to another entity that is the transport layer. I think, I should not call it a lower layer because it is not running on top of it actually. Because whatever you create that is being transported. So, that becomes an entity for this, these 2 are separate, but that is a way it has been written lowest in transport layer in RFC; then there is a third layer they do not try it bottom most, upper layer kind of thing is structure is not used the way it is not used in and networking. Third is what is known as Transaction Layer, what is it mean, for any kind of signaling, I will any signaling will consist of multiple transactions. May be, I will send you an invite the responsible will come for the first transaction over based on that I will decide on some other transaction which need to been.

So, there define every time there is 1 transaction which will happen fully and it is possible, I do a transaction with you and you give responses of that is a transaction running, is now on my behalf of is transacting with somebody else. So, I client will be transacting with sipporxy, sipporxy will be transacting with another sipproxy. And each transaction is separate each transaction is independent, but they are initiated by the earlier transaction. What is the different transaction in message?

Transaction there is finite dusted machines, the message once you transfer, see for example, how you will ensure reliability. I am transporting over UDP, UDP has no reliability, you send a message you create the lowest layer thing, the message through encoding and whatever it is, message format. You transcend the message over UDP using the transport, you have done the transaction now, you are waiting for the response message you lost what you should do. So, there has to be at some transaction machine, a finite stead machine which has to be implemented, which will keep track of the transaction till the transaction is complete or abundant. To maintain the transaction safe? Is state full proxies

(Refer Slide Time: 07:20)



If they are implemented between clients, I can have a transaction with this it is not transparently moved out. This transaction is only between these, this will result in a separate transaction with another proxy. Transaction is going to point to point messaging in n to n.

Not necessary not necessary. The transaction is point to point. Transaction is point to point, yes, it is point to point unless you have a state less proxy in between, because when you request which you will send through the transport will be transparently transmitted to the next guy. It does not maintain any states; if it is a transaction then this will maintain the state of that, within transaction whenever you are sending messages. You changing your state when you are getting back the message or response is for the transaction you are changing this state.

In this in this state full forward we have 2 transactions, server transaction with right condition for 1 side.

For a state full there is no client and server, whatever comes it simply just transports whatever is coming back, simply transferring here.

For the other side server main is forwarding the request the proxy.

In a state less it cannot maintain a state.

There use state full.

State full there is only 1 transaction with, it can result into another transaction, which is coming here.

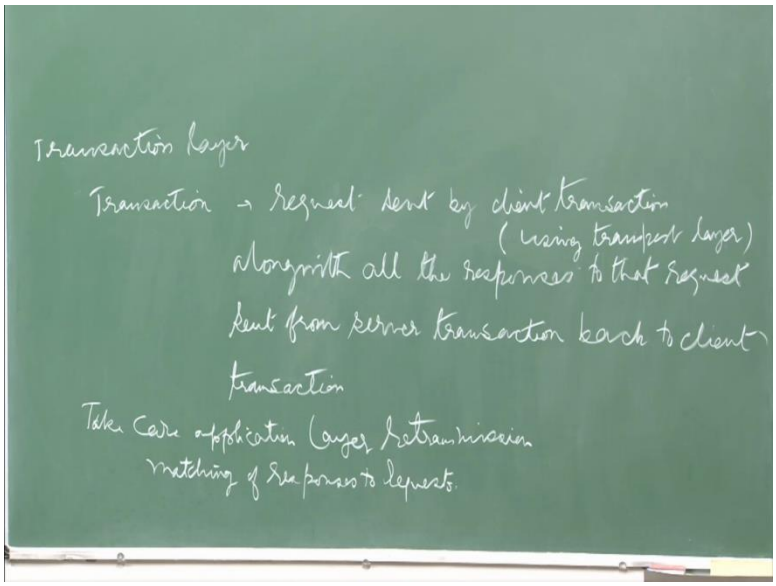
What we need for this side this is the server where a.

There is a server here, and there is a client here, there is a client here there is a server here.

Then what transaction we require client transaction as server transaction.

There is one client transaction other one is server transaction, client transaction, server transaction are actually finite state machines this is not 1 transaction, they with this actually with the term which is used here.

(Refer Slide Time: 09:18)



So, Transaction Layer is what the third layer is and formally transaction is defined as, a request which is sent by client transaction. Transaction is the instance which gets created in the client, is not a transaction client transaction is an entity running in the client. Similarly, there are server transactions. So, request is going from client transaction, which will cause a state to be changed. So, if loss happens it has to be retransmitting after certain time out; for example, this is going to be taken. So, reliability is the responsibility of this irrespective, what kind of transport is used. Transport can be anything remember, if it is TCP most likely it is reliable, UDP is not reliable, remember when I took the bob and that l is example, it was UDP ways transaction which was there in that triggers response message could have been lost in UDP, what you will do in that case.

So, client transaction will remember that the something was the response has not come within certain time. I have to again, retransmit that transaction request. So, is the request send by the client transaction, along with all the responses which you receive, not 1 response. You send a rate client transaction sends a request then the all responses which come will be included and this whole thing together will form a transaction actually. And this is sent using transport layer, is a different kind of instruction not exactly the same way. So, it takes of reliability, but any implement a client transaction, there is finite machine which is going to maintained for that.

So, they remember if for ultra link clear we had a finite stable machine and 2 sides, which can take care of the reliability of the message transfer. In case, a message lost you will time out and you will again re transmit message sequence in can be taken care of.

When it stands any server, whose takes get to process is that acting was client server is sending the request every time.

It will, it is going to have a transaction user setting inside.

No, when is the retirement there is that you will receive the there is response, how match it is a saying response of its particular transaction when I send the request forwarding to the proxies forwarded will receive a response.

There is a dialog and there is a corresponding a remember sequence number for the command. So, dialog will give the basil complete dialog, which is happening for that corresponding that call. Setup and then command sequence tell this transaction corresponds to which command, for each command there will be 1 transaction which will be created. Transaction user can issue multiple commands. So, there the layer on top of it is transaction user. So, this will handle all application retransmissions matching of responses to request application layer time outs. That is what, I was telling this will take care of any this is important thing because otherwise there is no reliability built in were application can actually use UDP.

The retransmission is possibility.

Is by, this transaction layer.

Transaction layer is proxy or in.

Everywhere, transaction is between 2 end points. see, it is not between Alice and Bob, is between Alice and proxy and whatever was at end Atlanta proxy. There is a transaction which is going to run between them, similarly between this proxy, another proxy there is entire transaction between this and bob there is another transaction which is running.

Who will arrange they have to read output it will I have to reconstruct.

Alice will realize for re retransmission layer, this will realize for retransmission message an this realize for this section.

My question is this p whether this we.

What is happening is, in between this there is something called transaction user tu. So, when, this guy actually sends a message, this will create a transaction dip for corresponding to, see these are transaction user is the layer which on top of transaction layer expect state full proxy, is state less proxy all other entities will have a transaction user. So, this transaction user has talk to this transaction user. So, for that it will create a transaction layer a transaction layer will be created. So, transaction distance will get created for invite.

Excuse me when is.

And this will go here and there is going to be a what we got this is a client transaction a server transaction will get created, which will then send a for example, I must be trying under end. And then this will be giving a message to transaction user, now this server transaction will remain as it is. This will not, it is will be dead its transaction is not over transaction user will now, know that I have I am a proxy; I have to now, forward it to appropriately. So, transaction user now will try to find out which is other proxy to who I need to send the message for that.

My question is when the proxy there will be retransmission is.

By Alice

Um

Either this retransmission message will be further by proxy.

No.

Retransmission only for the liability between these points; See message can be lost here UDP is lost for example; this guy never received anything, what you will do. This will this will keep on waiting I have send an invite I will get a response. That can happen, it is a UDP transmission. If I do a very simple thing on UDP I am transferring transaction user get got created. You want to make a call, it creates a client transaction it sends invite and there is no client transaction technically. You are saying, I am directly going to use transporting, this layer is not there.

You form a invite you send it you went through UDP it was lost, this guy even is not appear that there was a invite being send to me. Then what you will do? UDP packets are in constructed on IP packets; IP packets can be lost, because of congestion. There she will keep on waiting for all the time for getting even 100 kind of response. So, there estimate out, you have to again try it, but should it be done by transaction user is intention is I want to make a call, that is it invite is sent. So, let this be responsibility of the client transaction for retransmission.

No, in this cannot she does not maintain a timer from use proxy does.

Why proxy will maintain this will maintain a timer, client transaction will maintain a timer. There is a timer; I said there is a finite desisted machine with each client transaction and each server transaction. So, when this guy wants to send an invite, it will create a client transaction, which will have a finite desisted machine along with all timers in everything, which in turn will send a invite request, through a transport mechanism encoded as per this lowest layer. And it will maintain a state now to fit for some time. Event timer expire it will assume, it does not been do it again send the thing, remember command sequence will tell, if this has been correctly here. Time out is happened; even if a duplicate goes it knows that it is a duplicate. I have already received this command sequence.

This is a serial number being assigned and the reasons why serial numbers has been assigned and this is a traditional number it keeps on incrementing every time. You put next command, for every command you are going to create a separate client transaction. It is possible that, you can do even a cancel. For example; cancel is a special command when you give a cancel it will create another client transaction, not the same client transaction, which is supposed to be canceled. And this new client transaction will send the cancel command and it will refer to the earlier transaction, earlier command, which in turn will go higher. And then this there will be a server transaction which will now, clear all the state for corresponding to this, will send the appropriate response if it require. So, all the cancel will send a response back; in turn it will tell the transaction user, which in turn will clear up the client transaction. So, cancel is also tricky in these cases not a straight forward. It is not control all dells. In fact, no where it is control all dells, itself is a complicated process. If you put it in the shell or a control C, if you do net (( )) net transaction a very similar things actually almost will happen everywhere.

Sometimes you can abort TCP connections, server side times out if you do not get a packet with or message within certain time. See most of the protocols, by design use something called softest state, there do not have hardest state. So, they remain alive, there active. If they keep on getting something what you called heartbeat, message is called other in. The movement that heartbeat message does not come for certain time, it will assume that, the other guy is dead or else forgotten or something has happened.

You simply reset every all data structure are in the server side, client side you will you will keep on will time out. If the response has not come again try it will do it multiple time if does not happen server must have been dead, it aborts and tell the higher layer something has happened same procedure will be followed here, but it is done in different way the naming is different, that is only thing and it.

So, on and application layer, because your transport is directly UDP this is only the encoding of the message in the application. So, this message is technically being routed over this transport, but this message is generated by transactions and client transactions and server transactions are created by transaction users. Only in his state less proxies this is not the transaction user is not there. So, correspondly there is no server transaction and there is no correspondingly client transaction.

So, whatever comes you simply change the header what as per the policy and transfer it. you do not even worry about what is happen, what is not happen. it is the end point which we worry, if it is lost or if I have done a wrong translation, reverse whatever come you again do the transaction pass it back. So, they actually take handle what we call application. Take care of application layer retransmission I think I will write it then matching of responses; each request will have a separate client transaction being created. So, when the response is come back they are transferred who appropriate client responses.

In ruler of word transaction there is a.

Transaction is the result.

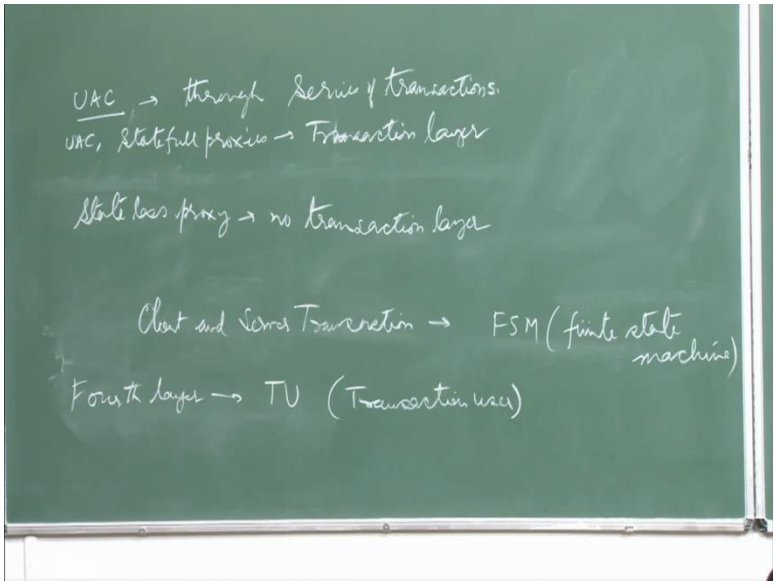
Where, in case if I am the client I want to set of production there will be 1 transaction user and that be 5 transaction layers which will be.

Yeah and of course, what we call application layer time outs. So any, what we call UAC was.

User aided client.

User aided client and whatever it will do it will through a series of transactions. UACs as well as. So, this is always will be working through series of transactions. So, now, it is important that, UAC and state full proxies have transaction layer. In fact, I should write transaction, because TLV used for transport layer most is state less proxy no transaction layer.

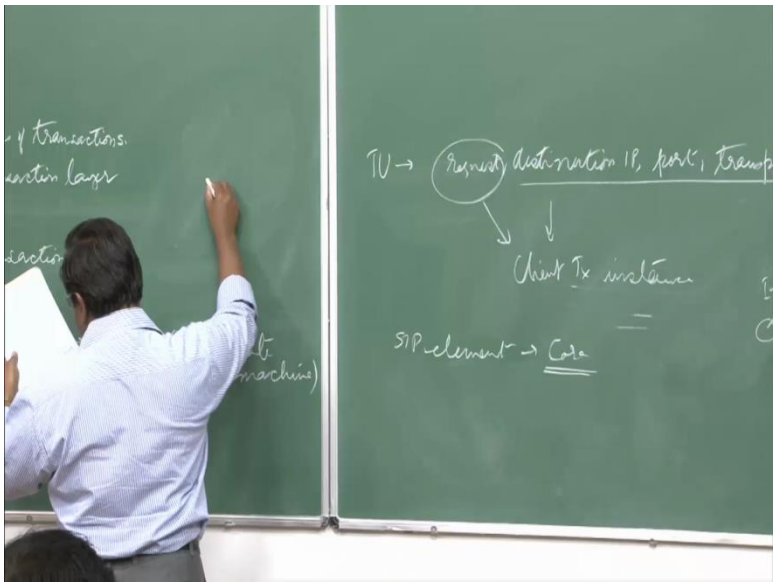
(Refer Slide Time: 23:57)



Of course, all the time this client transactions or server transaction, they are always represented by fine distinct machine finite distinct machine is, we are finite states are there and depending on whatever the input. The state will change and there will be corresponding output which will be coming out, FSM is that FSM stands for finite state machine and of course, the fourth as I mentioned is always going to be transaction user. This is known as TU transaction user is the user, which is technically a software running there which creates transaction machines, client transactions. And server transactions and expect all expect the state less proxy all sip entities are transaction users there is no separate all entities are itself are transaction users.

So, proxy is the transaction user it is not nothing, inside the proxy, itself is transaction user gives the reason client itself is a transaction user, because it creates transactions proxy also does the same thing, raise star also does the same thing. So, expect state less proxies everything else is every other entities is a transaction user.

(Refer Slide Time: 25:59)



The way it is going to be done is, what you will do is it will create a client transaction and for this it will create client transaction instance actually. There can be multiple client transaction instances as she has mentioned; as a case, you want to make multiple calls or multiple invites to be sent, you will create for every kind of thing is separate as this. I already gave an example you create an invite you create client transaction instance, you want to cancel it will create another client transaction instance in that case.

Which will then be referring back to this, because is the cancel it is a separate transaction, but going to refer to in earlier transaction? So, there will be server transaction correspondingly which will go and do all cancellation thing on the server side and once the confirmation come from that side, it will do the cancellation on the client side. So, here cancellation is slightly different, you create a transaction which is invited for example, on the server side, certainly you want to cancel it. You will do not cancel it directly you will start a separate transaction which goes creates a cancel server transaction it will refer reference will be given to this knows from the at reference this has to be cancelled.

This will be cleared off the response will come back; whether the response comes back you will clear this thing. You will tell TU and this will clear off this 1, by the 2 separate transaction instances it is gets created. And for creating a transaction instance, you require a request, you require a destination IP.

You require a port, you require a transport UDPTCT or STTP whatever it is, and once you give this thing, based on this is where the request has to be send. This always lead to creation of client distance, client transaction distance an transaction states will be of the type request and the message will be transported to and entity on this side. Where server will keep on waiting the moment I request come, it will check, it will find out there is no server instance for this it will create a server instance. It is like forking for example, whenever you are doing telnet to a server, oasis assist to a server, there is only 1 server running there. First time the request go it will figure out there is no instance running.

So, it will actually do a fork is creating a copy of yourself and then that will meet responsible to response to this. So, you will creating class server transaction instance, this will be done exactly the way, the way it is done in network you will create an instance on the server side also, cancel thing I have already told. And of course, this is basically, what are the layers which will be used for communication purpose. Now, most of this sip elements we define something called core, core is the functionality of course, this is only an abstraction, this does not tell that software has to be designed only in this way. It depends, whatever you feel, way you feel comfortable you can design your software, but there is an abstraction given in RFC.

So assume that, if somebody else design software whatever, but he is following this abstraction. You should be able to communicate with him, only that thing has to be ensured by, you do not know how the other person other server or other sip entity have been implemented. So, it have to ensure if you make me client, it should be able to work with anybody is proxy, which is sip compliant. If you are designing proxy it should be able to talk with any other proxy did not built by anybody else or any client raise star it should be able to talk with any kind of proxy.

But internally how implement it is your, it say I think implementer issue. So, that IETF's should not govern, because that innovation way, I think should I think we should live it to the event, I leave to the inventors. You can do lot of innovation I implementing the

software itself. So, in fact, every entity, every sip element we define cores actually in this case, core is the implementation core is technically the implementation and they are, they technically represents TU client transaction instance creation mechanism, client transaction instance everything put together in software that is core and this will be existing for every entity expect the state less proxy. So, that is core and core will depend on whether it is a proxy core, whether is a user client core or whether is a user server user UA's user agent server actually core or it can be raise star core, you can define or you can re directing server re direction server, which is there it can be core for that and remember for every transaction will get completed. When you will get a final response, it cannot be completed before you get a final response.

So, transaction client transaction, server transaction this transaction will get completed then they say entity server entity can go away if it requires. And this will happen only when the final transaction has happened; a final response has been received for a request. So, if you carefully note then the Alice actually has sent a request, there was invite which was sent. Final transaction happens; final response comes only when the final response is from all of the we have reached here, then you will get an and is kind of final response

So, in fact, all response codes which is starts with 2xx, 3xx, 4xx, 5xx and they are all final responses. 1xx is not a final response, it is a kind of a temporary response you call it provisional and it will not terminate the transaction. So, your client transaction has to be maintained if you are getting this provisional responses, you cannot terminate that. You cannot closer 1xx series will not close the client transaction. Only anything related to this will close. Now, this request response now, can actually can keep on moving from 1 person another person so on.

(Refer Slide Time: 33:48)

So, there in terms which we check things which can happen which we define. So, 1 thing which can happen is known as loop. So, is this is a ph1 talk to a proxy, figures out still transaction not over. this request is sent to another proxy this proxy analyze the invite whatever request which was sent and this request comes back this guy and this request again goes back in this direction that is very important, then it is known as a loop which is going to happen. So, this is loop something also another interesting thing which can happen is, you send a request a proxy this can send to another proxy, this instance stands it back, but this new request which is come back through Some whatever way.

There can be other proxies also is not going to go in the same way, it goes to somebody else. So, there is different than loop actually, this can happen because of redirection, this can happen, because of redirection because you are probably raised out to this mean, but then you have moved some other place you told this guy the you tell this person where I am. So, it tells this person that I am here, it actually passes on the request to this guy, this guy then looks because now, the remember that sipproxy when I do redirection. So, redirection is the source the original destination is move to converted map to another new destination.

So, bob was that block c.com and then he has move to say some xyz at at lanta.com. So, that is a redirection. So, at lanta dot com is back here. So, request comes here for xyz or

may be some other guy whom this guy cannot go directly it comes back here, which in terms sends the response request here this kind of thing is known as a spiral in this case.

Sir, this is the legitimate entity

This is the legitimate entity; generally, it is a legitimate is legitimate.

Loop will be endless or I mean it will come out.

It depends, loop usually is not desirable, but yes it is possible, a loop is there and loop is legitimate yes it is possible, but loops needs to be resolved in that case. Probably 1 example is bob at block.c.com. His job actually got moved. So, he is still return that account he did a redirection he joined somewhere else say, Johnny at Atlanta.com. So, the call was redirected here and then by the time he also actually has moved this entrance still return. And now is sum JoJo at block c.com. So, it is being again redirected here. So, loop is still a valid thing, what does signaling is happening in a loop. So, ideally speaking you should have actually the error here is that you should of now told this guy that, whatever is bob at the block c.com is same as JoJo at block c.com.

So, redirection can happen here itself directly. So, earlier this loop is nothing but, what we call a data which is not required a configuration which is bad actually of course, is not going to empite the call can will be still be rooted, but is not desirable. Mostly it will happen because of direction because for whatever reasons, this redirection is probably you are moving and then you are even returning your earlier account and redirecting to somebody else, like dot forward in email, but dot forward in email.

For example, from ITK you say it goes to whatever is xyz at gmail.com. there you do a dot forward to go to somebody else here, it come to IIT gate what is you have to be mean, but different name, but further you still the same, but name is already changed. And that dot forward again send to gmail.com with the different name id different mail id and of course, if you create a loops same mail here it will keep on looping indefinitely now that is in validity thing. It will terminate in valid loops are not allowed and how that valid loops will be terminated.

Yet him.

Yes, there is a maximum forward limit and that will be terminated it should not happen keep on happening indefinitely. So, loops usually which are invalid loops are taken care of by maximum forward apps, which have there and then of course, this is.

Only way, you have to handle this way you cannot to forward every.

Whether you.

You do not handle it simple. So, if you see, it will see, I am telling here it is a look only way if this can be said is if as of now because there is nothing has been received from the endpoint. The call id and that the tag of the sender these 2 pairs actually if you combine and those are registered here you just keep check if there is any other requests which is coming a while it is this particular transition is pending, which corresponds to the same particular call Id and same particular this thing and same particular what we call sequence, command sequence.

If that thing comes that is probably is the loop that time you have to decide what has happened, then you check this is this then you actually can do the these can do the pairs can do the check. And then, user need not even offer you can directly say bob dot block see com should be now directly forward to this loop is removed. That intelligence can be built in this system, now that is left to the programmer RFC does not talk about it.

So, if you are a smart enough actually you can implement this and then make your proxy better or you can at least inform the user we found a loop for you would you like to correct it. And this way to correct it at least is parents also can exist, but I think is parents are also not desirable if this could I been directly from here it would have been better. Now, this guy can do this job and short circuit directly from here. So, you will actual reduce a lot of load on these machines fine over loops and spirals.

And this will be done with in l gave some we have align at roming can there to say v e I say well act to do.

They have to do it, but these machines are entity they can figure out a collogue, what not collogue that is dialogue, but dialogue is not there unless you get the response back.

Because if this will be of different network productive there is no.

So, only the invite will go in this way, but the response will come pretty fast, they see they will always say, it do not they will not put what we call that thing, via fields.

So, whenever you have doing redirection do not put a via header entry into that, but the problem is if this is going to send to another proxy then this will not know it is going to go back to this guy. And it puts a via header then it becomes a problem and those complications are there and we have to live with those and these kind of things I thing has to be identified by processing. See because this will not be cleared remember.

So, whenever a new call, new invite comes whatever is the current transactions which are pending which are not clear compare with all of them and based on that you decide if something is being duplicated. Where identify loops are something eminately stock with those proxy servers are at line dissolved the matter. So, that intelligence has to be part of the proxy design. So, it is not a triggal thing. So, this also not obvious stuff now, I talked about that back to back user agent, I talked about stateless proxy, is state full proxy I cannot diagrammatically actually represent them once I have come with this transaction stuff.

So, I talked about this thing B2B is the agent. So, this special kind of proxing thing, but actually it is not a proxy, is only for signaling back to back user agent also takes care of media. So, there is a proxy, there is a proxy this is a B2B user agent. So, media is going to be transacted on this signaling through this particular path. So, there is 1 user agent there is another user agent client, user agent server media in signaling both will be in transacted here. And this can make another sip trapojite that is a back to back user agent. If it would have been the proxy then media is always transacted directly between these, certain media gate ways can be built it can be of this kind.

See very in data's build a.

Signaling in data both for media we call it.

This will not be called a proxy?

This not a proxy, this a back to back user agent.

How can differentiate, you know if it is come.

Proxy has only for signaling part there is in this is no rule for media, media is transitive directly between PS, but this like a another PF. So, this guy cannot see this and this guy cannot see this. They both have to go through this for signaling as well as media this kind of a media gate way.

You know what I do thing will get?

You do 1 to tape the call, along with all signaling information you have this is the best approach. Back to back user client is the best approach in that case. So, do not allow the call to be setup between these 2 entities use B2B. So, 13 usually will require this, because I am now not using mega courier for controlling this particular entity, again it will I am using sip only. So, it is you will actually see your user line is this, but actually it is not this. So, similarly this guy will feel as if this guy is here, but actually he is not and this is back to back connect specifies together in software.

But when you are each giving on the add address of IP, I said back response will he not know.

No this guy will do the signaling here, this list will keep in pending. This B2B user agent will ask this client to it should not talk to this guy with all the information given by him and it will behave as if he is this guy.

You can precede the request will both this path.

Yeah.

But is called normally done.

It is not normally, it will not be done, normally it will not be done, but these kind of a specialize entities will also exist. So, I think all the state full proxy is in transaction a state full proxy is also need to be explained.

Sir, how should define 2 we also by using normally small as regarded this it will the path we had been.

We can in though also is used what we are.

For thing is, when the proxy on the receiver side gets a message it now signals to many entities not 1 entity which are registered for the same user.

This a parallel coding.

Is a parallel thing when the responses will come all of from all of them and it will intelligently will figure out as per the policy that only 1 of the message have to you pass that.

Sir, where we do the better and user.

Parallel forking for example, if you have setup as a feature with your registrar, you want that all 3 of my phone should ring when a call should come and can pickup from anyone of those. You can say, first of all try at my home phone give 4 rings, if I do not pickup then my office phone should ring 4 rings should be there, if again it is not picked up then my mobile should ring that is a serial.

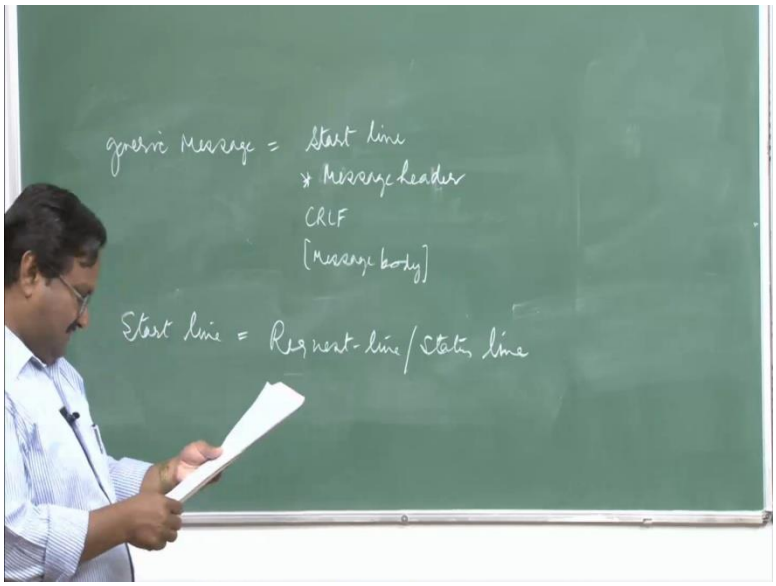
Would tell you all the mix approach?

Yes, you can, that is your thing you have to do the everything hence to be with registered with the registrar and this inform get everything from registrar and do as per that. So, policy of calling is again left to the implementer end users, RFC does not define that. In fact, there are many set of RFC's other RFC's is talk about this thing.

Is it a user choice.

If a users choice is a policy. So, next is the message format. So, this I have already mentioned, but I am just formally writing that they should not take that much time.

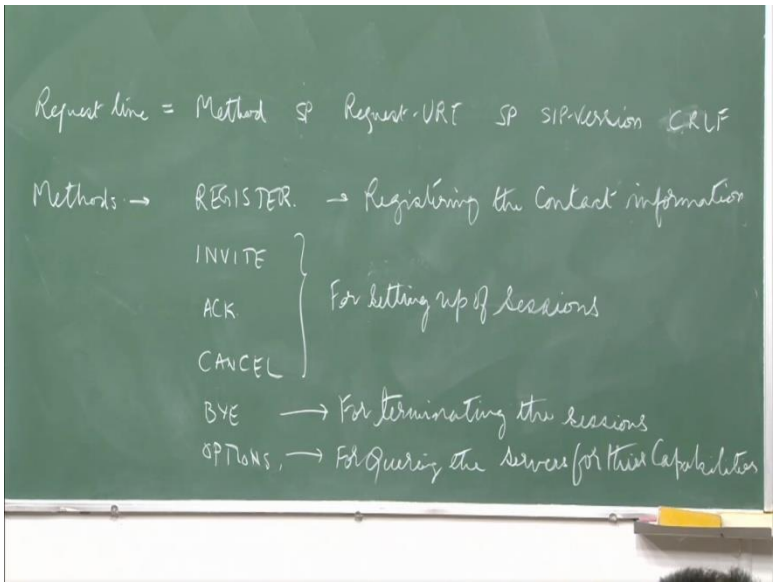
(Refer Slide Time: 47:40)



So, generic message will be always look like this I am now, writing the grammar of that actually, it will always consist of the start line of still header switch will be there CRLF category written line a gap. And then, you will have the message body which is optional, I think optional is with this and this is a compulsory, flexible and a start line can be either request line or a status line. And even when message body is not there this an optional thing, but CRLF is compulsory we start line is compulsory, this a variable field actually message item they have to be there message body is optional. So, because I know there will be 1 line.

This is a variable number of lines; the end of the header will be identified by CRLF which is again compulsory thing. You cannot identify the end of this number of header fields unless this is there that is why this always has to be present. This an optional thing that is a BNF grammar actually which I am writing now and request line will look now these are 2 option request line or response line.

(Refer Slide Time: 49:34)



So, request line will be a method that is a space I have already mentioned this thing earlier, but I am writing formally. Then there will be request URI, there will be a space single a space character and then of course, sip version line field. Sip version as per the CRLF we called and line field we will come back to the first thing. So, that is the request line which will there.

Then this length of the message and all that will not be there.

And the generating switch after message body the length of the message and so on.

Length of the message will be part of the message header it is part of the messager content length that will define how many bytes will be there in the message body. So, if it is 0 there is no message body. So, the methods which are permitted, so far I am have not talked about all methods. So, all the methods which are there are these which go in a any sip request. So, one is register invite we all know, what is invite you also know what is ack, it is not a response remember it is a method in the request. In the request line, the status line is where to will come.

So, ack is not an response, ack is a request then there is a cancel, bye, options. This is for raised in the contact information; this is use for setting up of your sessions. This is for terminating the sessions this for querying the servers for their capabilities. So, this is as per RFC 3261 for other RFC's also actually specify other methods, but these are the

one's which are required by every implementation. Other things are nothing but, sip extensions actually the other methods also.

Will street house will message that be the all were…

Variable length, variable number of address is not 1 line there can be multiple lines.

We since include or model i.

See that is why CRLF is there, see you identify, you know first line is this line you know this CRLF all lines in between them is message header. So, it can be 1 line, it can be 2, it can be 10, it can be 20; nobody stops, 1 of the field always will be content length which will define what is the message body size.

There is no CRC in this case.

There is no CRC, that is taken care of by UDP or IP packet or data link. So, usually if there is going to be noise done c will take care of that and will do the re transmission.

What is the difference request been take care of the…

This for the setting of this session.

What is the session management?

Session management if you re invite, in that is also again invite only, but you way the different parameters.

Session management will be transparent in detail.

Modification of parameters also and termination everything this is for management or you change the parameters of the media. So, I close here and then most more of it actually will discuss tomorrow morning.