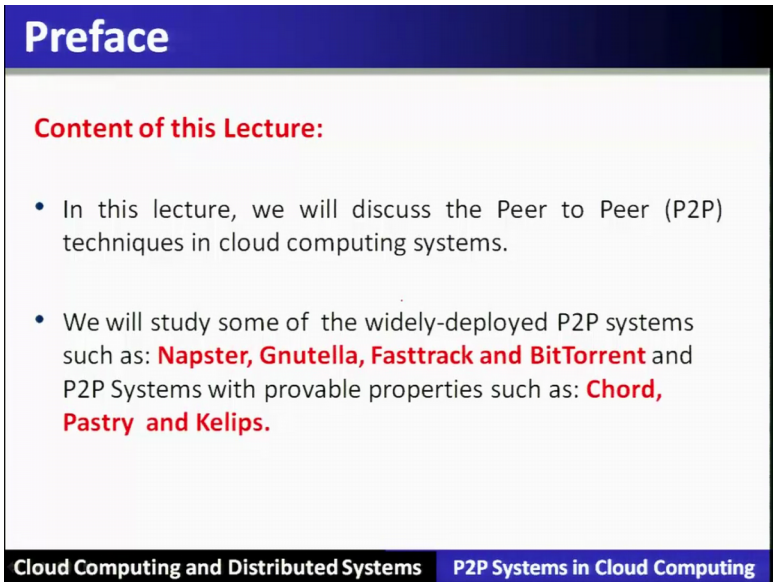
Cloud Computing and Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture – 18
P2P Systems in Cloud Computing

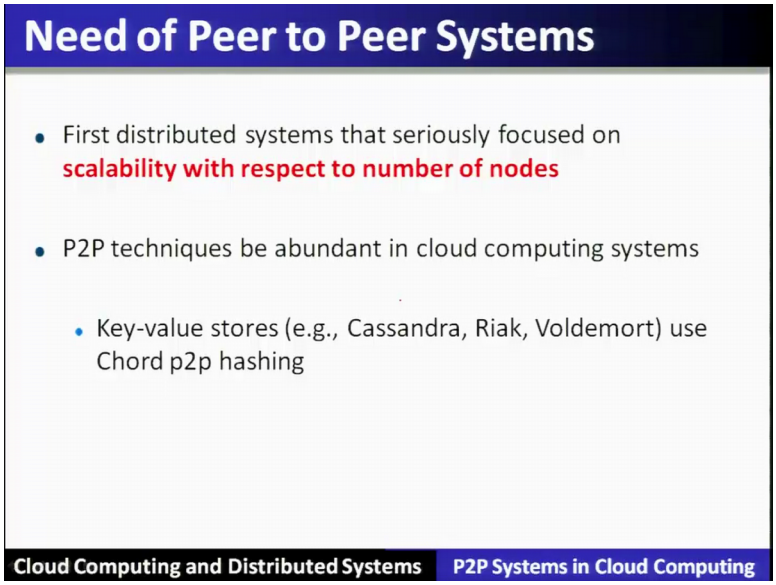
Peer to Peer Systems in Cloud Computing.

(Refer Slide Time: 00:18)



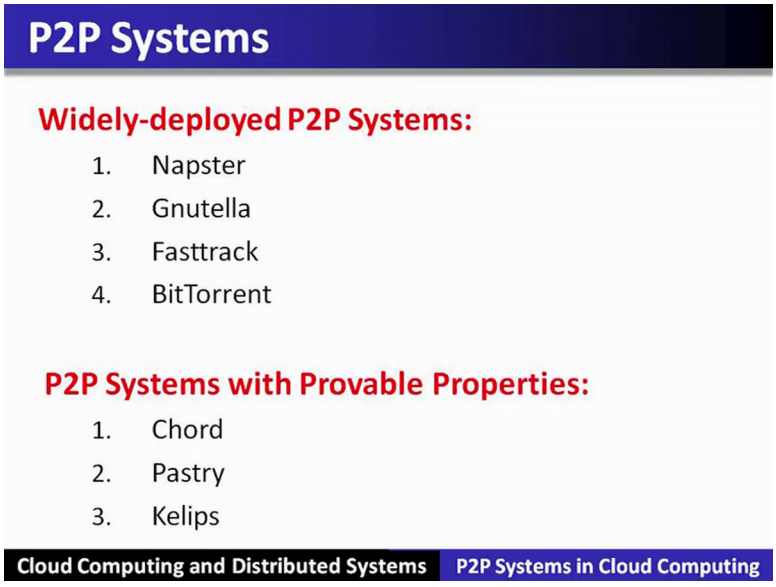
Preface content of this lecture we will discuss peer to peer systems and we will see the internals or under the hood of peer to peer systems as a techniques, which will be applied in various cloud computing systems. We will study some of the widely deployed peer to peer systems such as Napster, Gnutella, Fasttrack, BitTorrentand peer to peer systems with the provable properties that is from academia they are Chord, Pastry and Kelips.

(Refer Slide Time: 00:58)



So, let us see the need of peer to peer systems. So, distributed systems that focuses on scalability with the number of nodes is one of the basic issues which is required to support large number of clients. So, to support large number of clients or highly scalable large scale distributed system is required. And therefore, we will see one such system which is widely being accepted is called peer to peer system. The techniques of peer to peer system we will discuss which we will see that is currently being applied in various cloud computing systems such as key value stores that is Cassandra, Riak, Voldemort uses the chord peer to peer systems like consistent hashing and virtual ring.

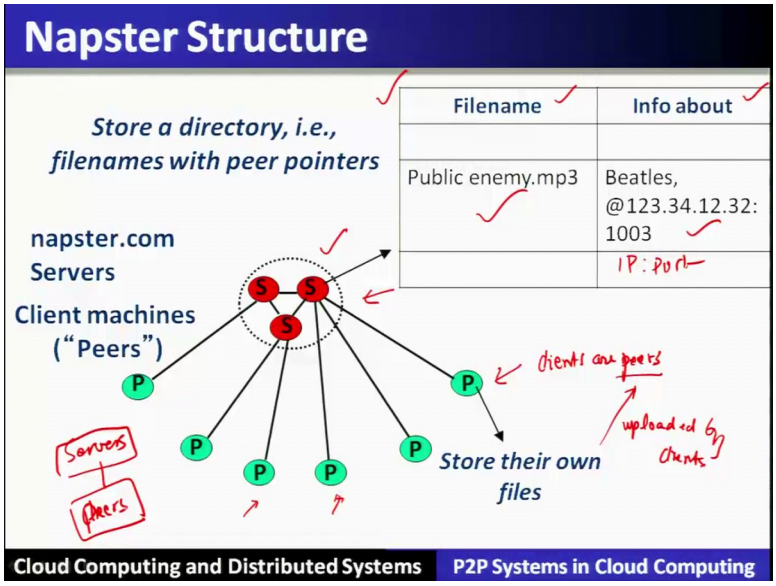
(Refer Slide Time: 02:00)



So, widely deployed peer to peer systems are such as Napster, Gnutella, Fasttrack, BitTorrent; we will discuss these particular widely deployed peer to peer systems their internals details that is under the (Refer Time: 02:18), the techniques which will focus on how these systems are evolving. Then we will

discuss peer to peer systems which are basically resultant from academia such as chord, pastry and kelips.

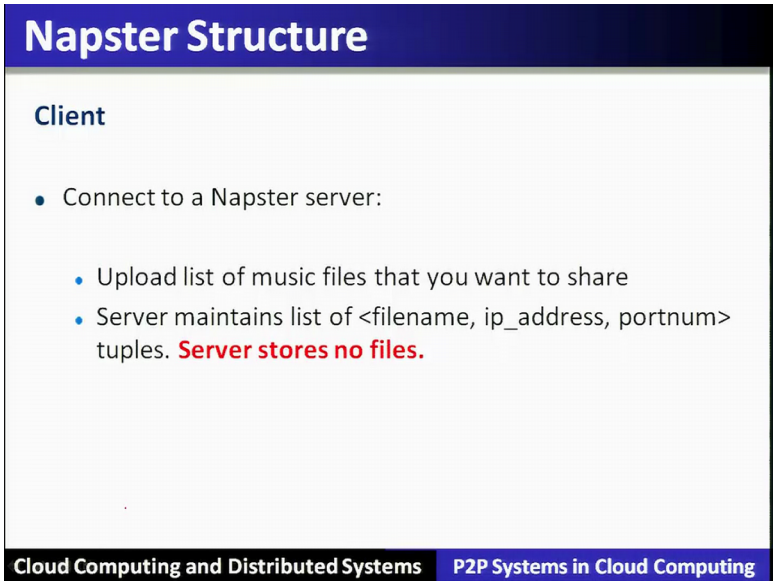
(Refer Slide Time: 02:38)



Let us see the Napster peer to peer system design. So, Napster is designed in the form of the peers; that means, the clients are the peers which are shown here as P. So, the clients are peers in this particular system which are shown as this particular Ps. Now these clients or these peers used to store the files which are being uploaded by the clients. Besides this there is a set of Napster servers which will store the dictionary or at the directory of filenames with peer pointers of where they are stored.

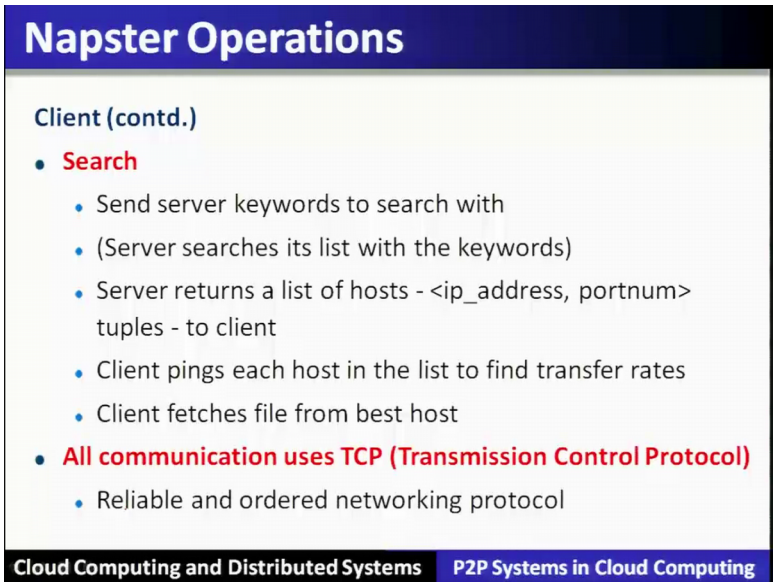
For example, the structure of the directory which the napster dot com servers used to store will be of this format that is filename and the peer pointers about the file name. For example, public enemy dot mp 3 is the name of the file and this information about this particular file on which machine that is IP address this particular file and the port address where this file is stored. So, this particular structure is maintained by the Napster servers. So, it has the servers and the clients they are called peers.

(Refer Slide Time: 04:43)



So, the client used to connect to the Napster server it can upload the list of files that you want to share. The server does not stores any file, but it maintains the list of file names its IP address and the port number this particular tuple in their directory.

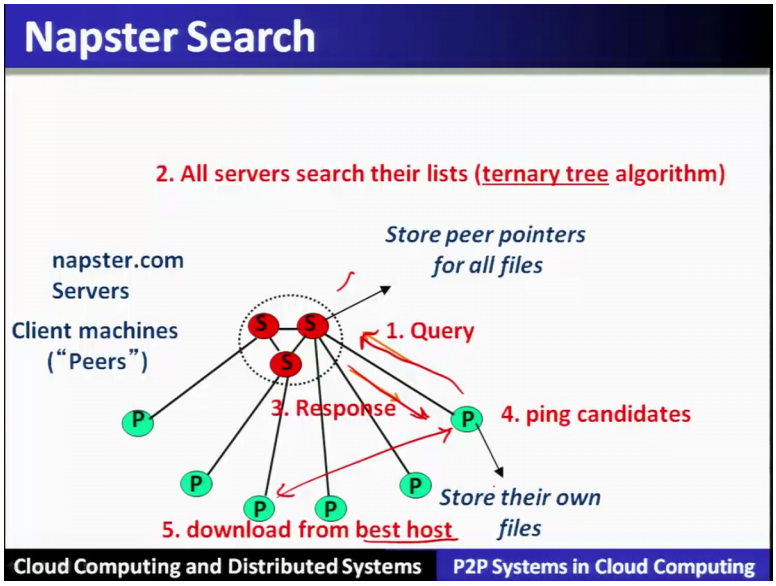
(Refer Slide Time: 05:05)



Search operation when a client want to search for a particular file with the keyword, then it will send this keyword to the server. Server searches its directory with the keyboard and forms a list of successful results and returns this list of hosts that is nothing, but IP address and port number where this files are available and these information that is the list will be sent to the client. Client pings directly to these hosts which are provided in the form of list to find out the transfer rate which is being provided by the different host; transfer rate by mean that the bandwidth which is available to download the file.

So, the client fetches the file from the best possible host which basically will give the fast download; here there will, here the communications are in the form of TCP that is the reliable communication.

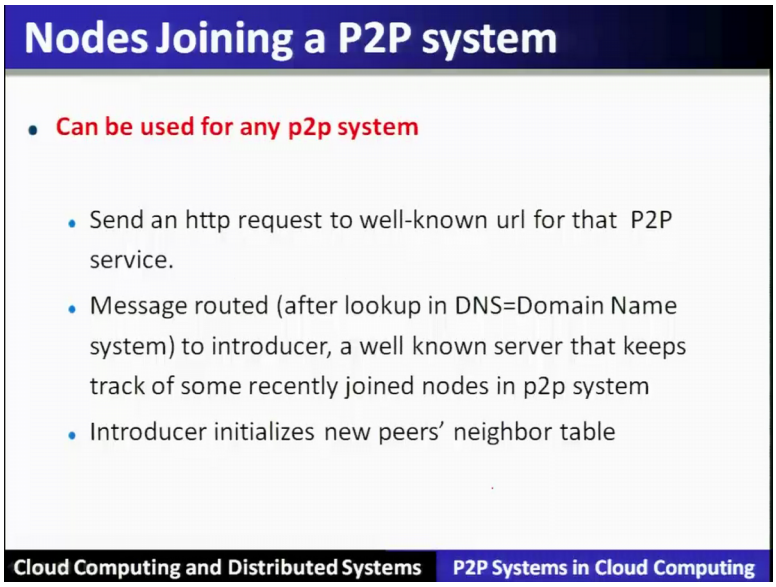
(Refer Slide Time: 06:23)



Now, let us see the servers, the Napster dot com servers are basically maintained in the form of a ternary tree algorithm; that means, each node of a tree is having 3 different child, in contrast to the binary tree having 2 childs.

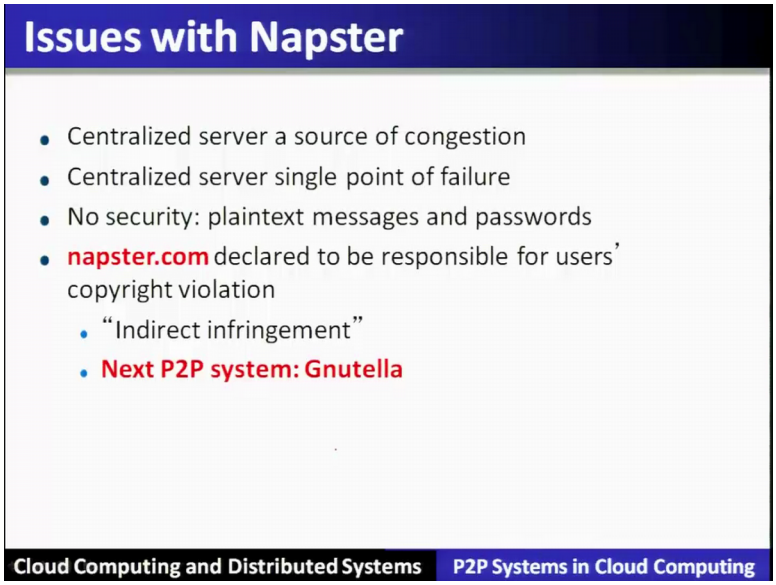
Now, this particular servers will store the peer pointers for all the files and whenever the client will query this particular file with the servers, this will be checked here in its directory and send back the response that is a list of IP addresses and the port number, which has those files. Then this particular peer we directly contact to those peers which is having the file and find out the best one; best host to download from.

(Refer Slide Time: 07:37)



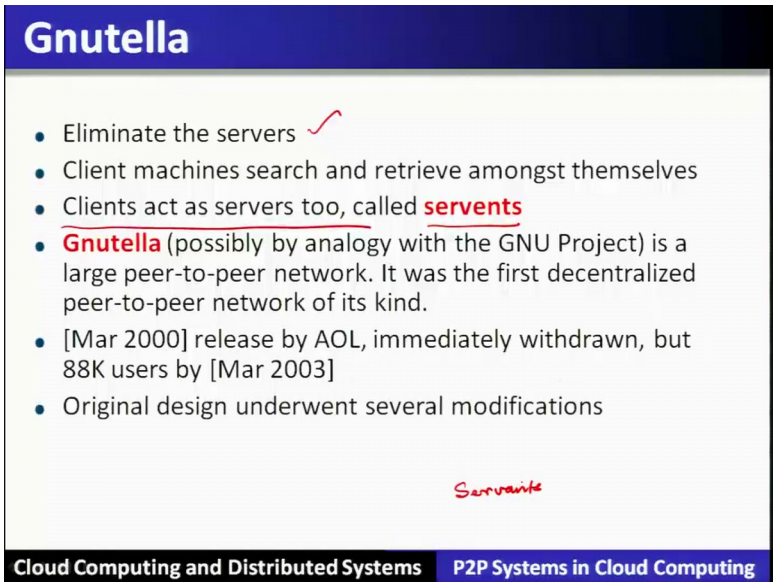
Now, coming to the nodes joining to this peer to peer system. So, it sends a http request to a well known url for the peer to peer service. For example, napster dot com; so, the message routed after the lookup from the DNS system to that introducer which is nothing, but a well known server that keeps track of recently joined nodes in the peer to peer system. So, introducer initializes the new peers neighbour table.

(Refer Slide Time: 08:15)



Now, let us see the issues which are basically affecting the Napster. So, the servers are primarily the centralized servers and hence it will be a source of congestion; also due to the centralized server there is a possibility of single point of failure. And here also there is no provision of the security and the napster dot com declared to be responsible for the copyright violation that is called indirect infringement.

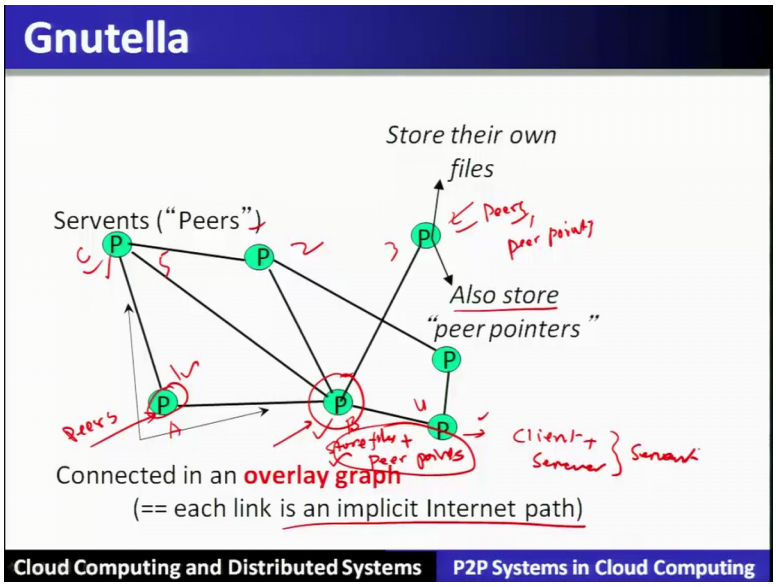
(Refer Slide Time: 09:00)



So, we will see the next peer to peer system which will eliminate these problems; to eliminate these problems Gnutella peer to peer system has in his in its design eliminated the use of servers.

Therefore the client machine search and retrieve among themselves therefore, the client will act as a server too and therefore, they are called as the servants. So, Gnutella is a large peer to peer network and it was first decentralized peer to peer network of its kind.

(Refer Slide Time: 09:48)



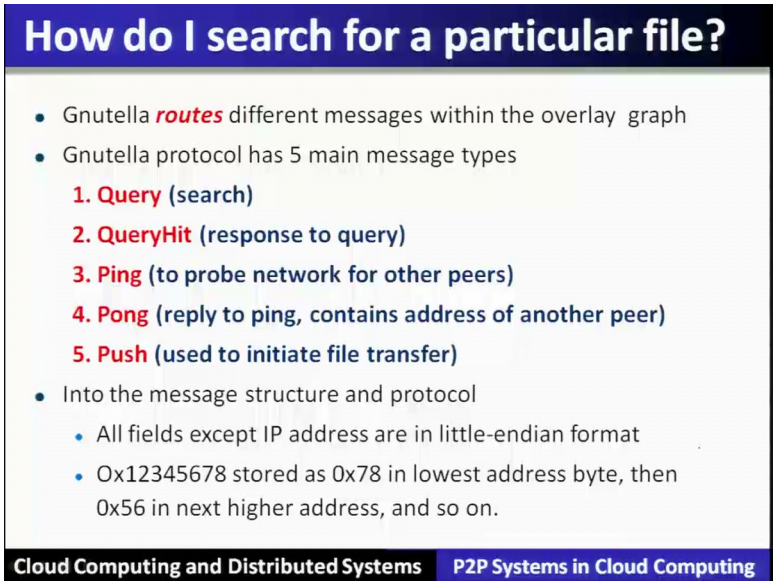
So, let us see the design of Gnutella. So, Gnutella design is organized in the form of the overlay graph. So, overlays are constructed over the underlying internet path that is the servants which are shown or which are the peers in Gnutella. They are forming an overlay graph with comprising of other peers who are connected through a path that is through internet path.

So; that means, if this particular peer is having the links or the connections with the neighbouring peers; that means, this particular peer A has a internet path with B and C: similarly all other peers will have the path and form and overlay a graph. Now this particular peers will store their own files and also store the peer pointers of the neighbouring peers that is the neighbouring peers which are storing the files.

For example, this particular peer has 1 2 3 4 and 5; 5 different peers. So, this particular peer will store his own files plus all the peer pointers that is the information about the files which are stored in these neighbours in the 5 neighbours are maintained over here and that is why it is the client plus the server that is called servant.

So, Gnutella has given the concept of an overlay graph.

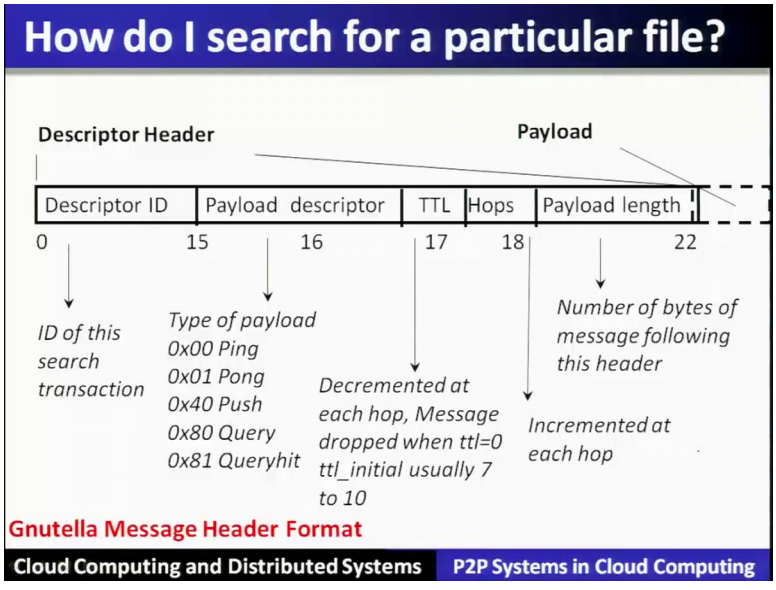
(Refer Slide Time: 12:09)



And then provide then provide the routes for different messages to flow on the overlay graph for searching purpose and other operations. Therefore, Gnutella protocol has supported 5 different messages types the first one is the query; that means, whenever a particular client want to search a particular file that is done through the to the message type called query.

So, in the query the keyword which keyword of that is the name of a file will be given for searching. The response to that query will be collected through the QueryHit, then as per as to maintain the overlay graph ping and pong there are 2 different message types; they will probe the network for the other peers in the overlay graph and pong will be the reply to the ping. Finally, the fifth message type called push will be used to initiate the file transfer even if the responders are behind the firewall. The information which are encapsulated or packed into the message that is the message structure and the protocol follows the little endian format.

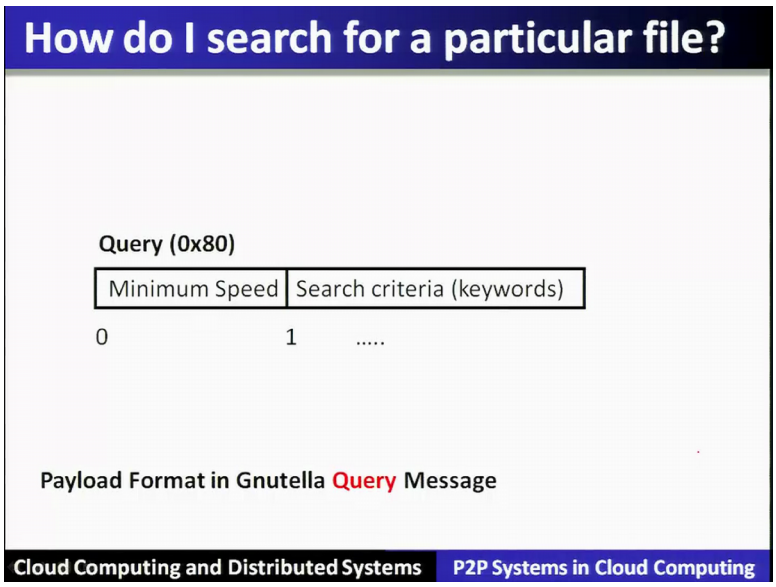
(Refer Slide Time: 13:41)



So, this is the structure of the message header it starts with the descriptor ID that is for each search transaction it is having a unique ID, then follows is the type of the message that is whether it is ping pong push query and QueryHit that we have already explained.

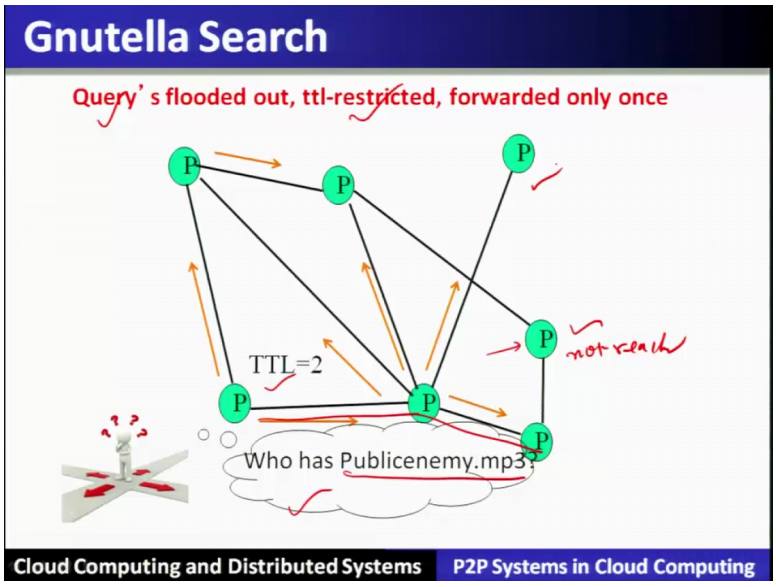
Then comes next field which is called TTL Time To Live; now this Time To Live will control the flooding of the messages on the overlay graph that is it will be decremented at each host and the message will be dropped when TTL becomes 0. Then comes the number of hops it will be incremented at each hop and then it will be having the payload.

(Refer Slide Time: 14:39)



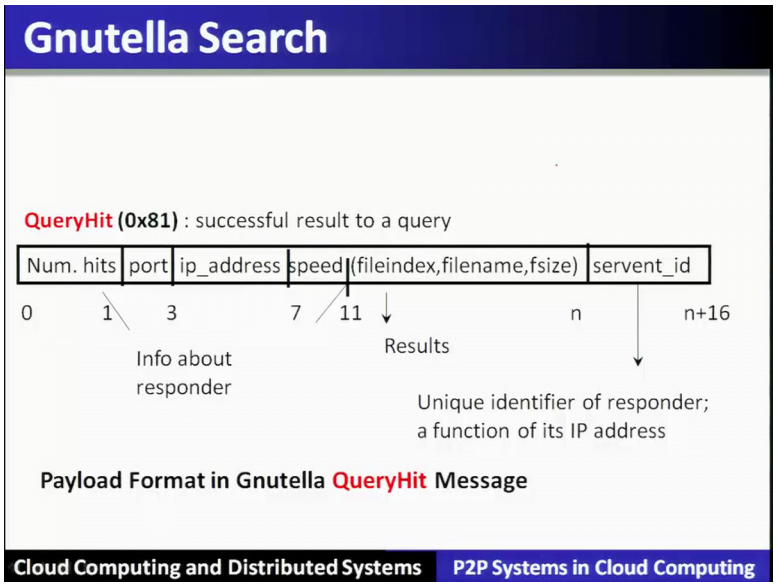
So, this is the structure of the Gnutella query message.

(Refer Slide Time: 14:47)



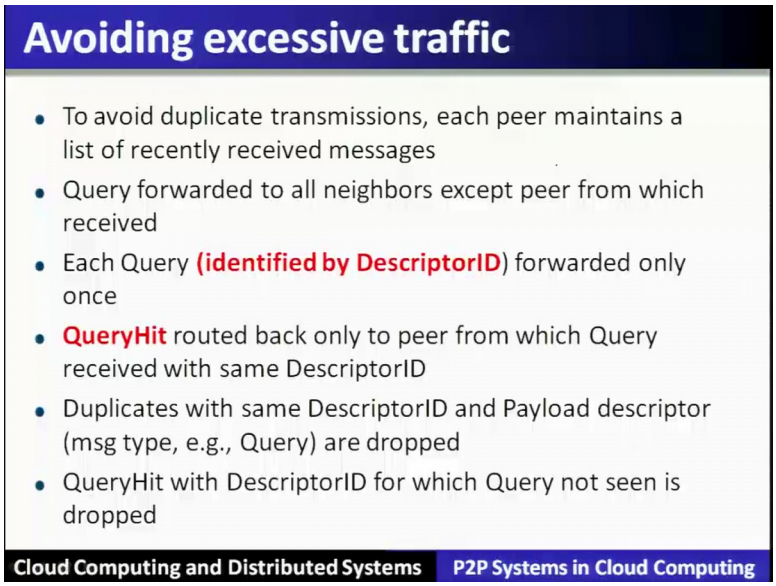
Now, let us see how the search operation is supported in the Gnutella. So, the query message will be containing the search keyword and let us say that it want to search this particular file that is publicenemy dot mp 3. Now this particular message will also have the TTL that is 2; that means, this message will be flooded, but it will be TTL restricted that is up to 2 hop within that 2 hop; it will search. So; that means, this particular message this is one hop and this is another hop since this particular peer is having at the third hop therefore, it will not reach to this point; so, also for this particular peer.

(Refer Slide Time: 15:47)



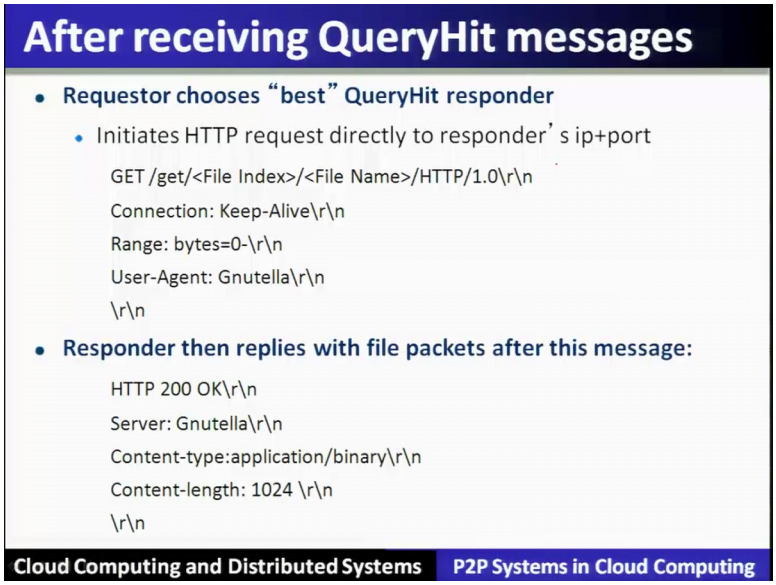
So, these messages will be forwarded only once. So, the QueryHit means the peers when they receive the query message they will search and give the response in the form of QueryHit and it will this particular result in the form of a QueryHit will be routed on the reverse path.

(Refer Slide Time: 16:09)



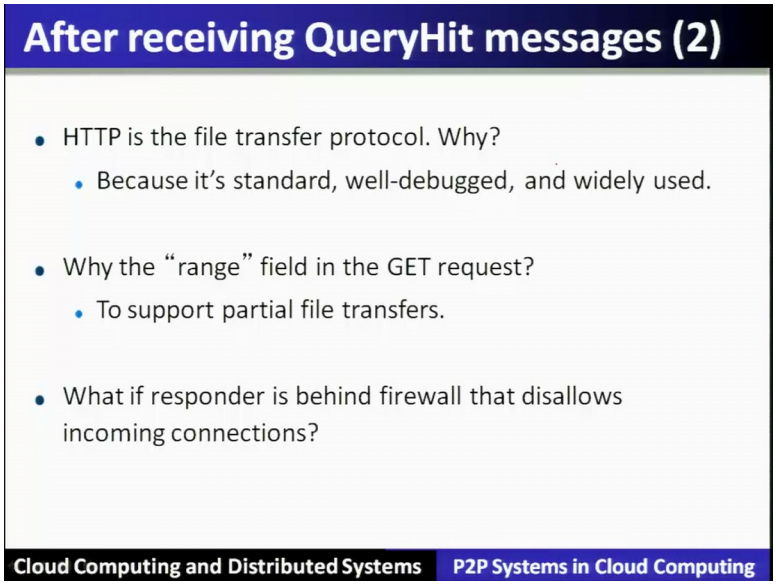
To avoid the excessive flooding the Gnutella has made various provisions; to avoid the duplicate transmissions and the query is to forwarded to all the neighbors except the peer from which it is received. And it will be forwarded only once QueryHit will be routed back only to the peer from which the query is received with the same descriptor ID duplicates with the same descriptor ID are dropped QueryHit with descriptor ID for which query not seen is also dropped.

(Refer Slide Time: 16:46)



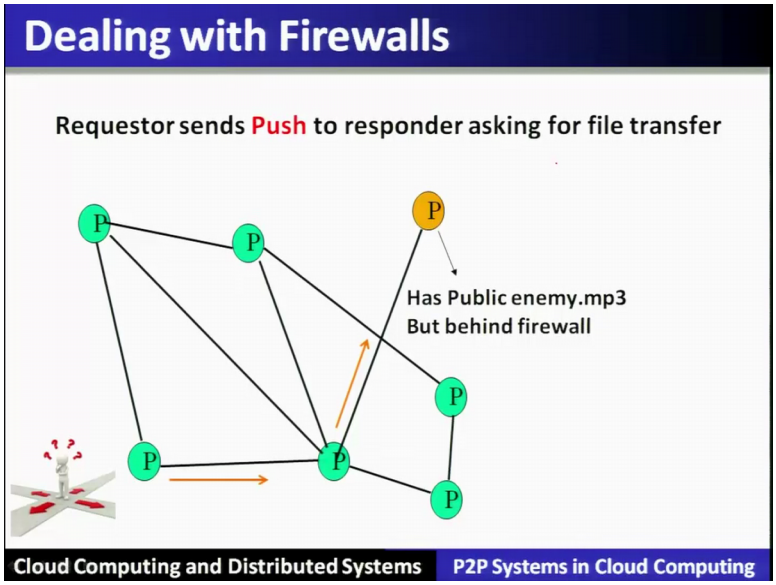
So, after receiving the QueryHit message that is a response the requester chooses the best response and then the responder will reply the file packets after this particular message.

(Refer Slide Time: 17:07)



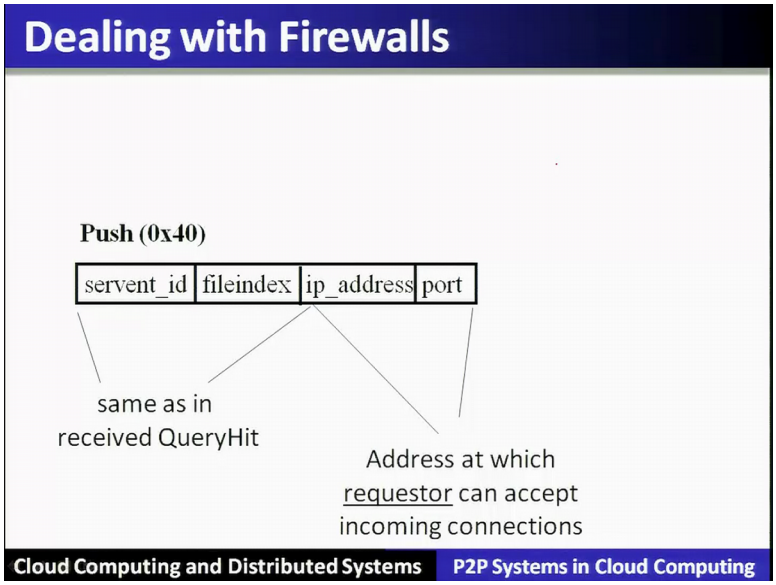
This Gnutella uses HTTP as the file transfer protocol; now if the responder is behind the firewall that disallows the incoming connections.

(Refer Slide Time: 17:19)



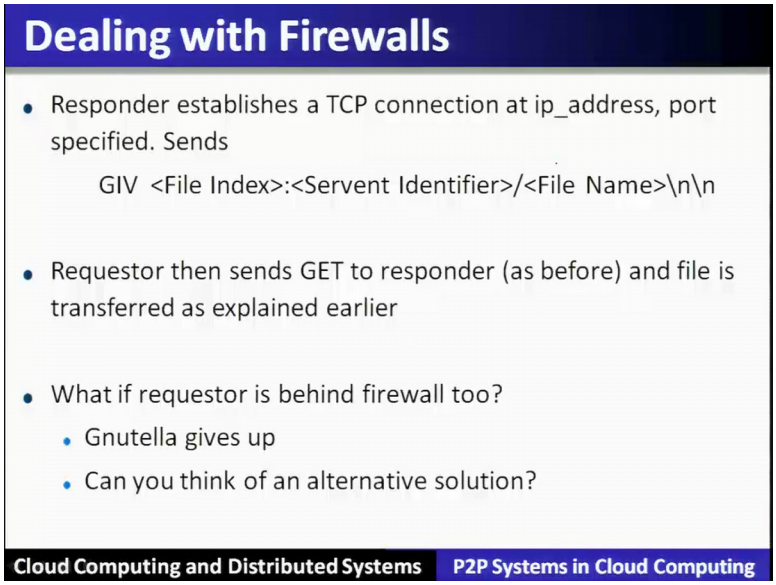
Then a push message is being used. So, the requester send the push to the responder asking for the file transfer.

(Refer Slide Time: 17:31)



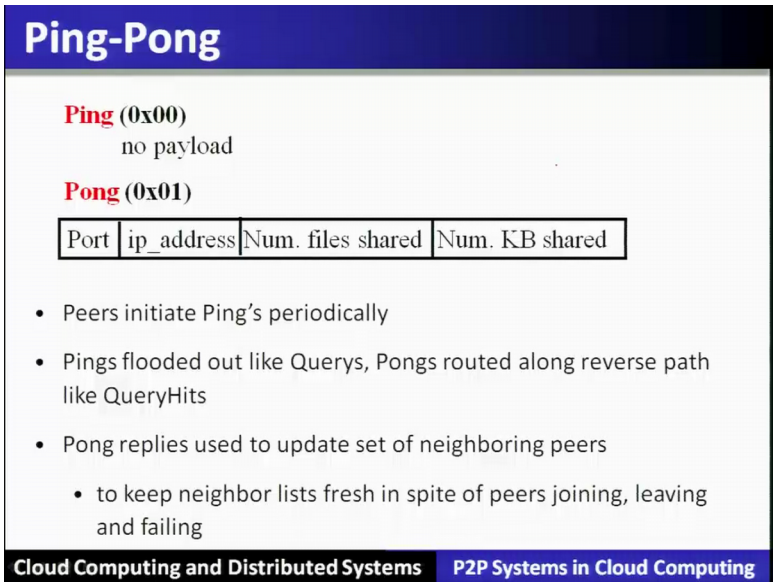
And it will allow this particular access in this particular scenario.

(Refer Slide Time: 17:35)



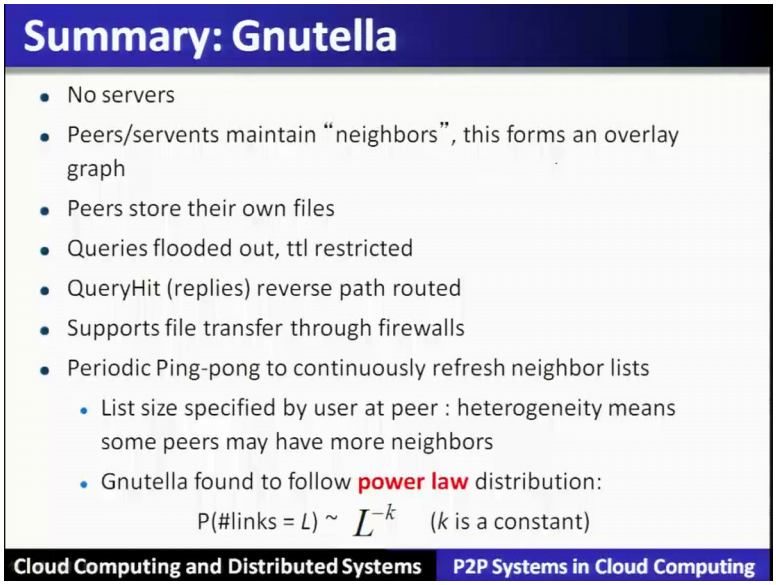
If the requester is also behind firewall then the Gnutella gives up.

(Refer Slide Time: 17:45)



Now, ping and pong there are 2 different type of messages used in Gnutella for maintenance of the overlay graph.

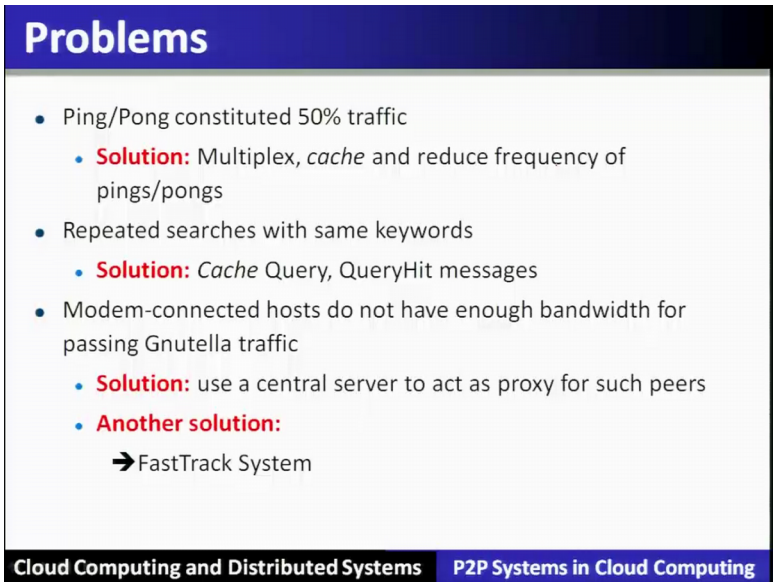
(Refer Slide Time: 17:54)



Now, to summarize Gnutella has no servers; that is the client also does the operation of server hence they are called servants. This avoids the single point failure of that particular server and also ensures the scalability; large scale peer to peer system possible using Gnutella to support large number of clients. This peers of the servants are maintained in the form of a overlay graph and peers store their own files the queries are flooded out, but they are TTL restricted.

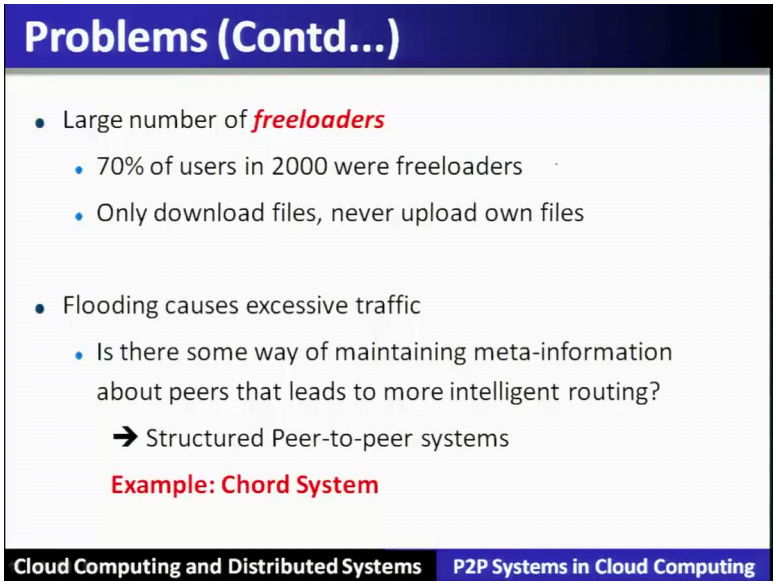
Here the periodic ping pong will refresh the neighbor list; that means, it will ensure the maintenance of the overlay graph. Now as far as Gnutella is concerned it follows the power law distribution that is the probability of number of links of a particular peer has will be of the order that is reciprocal of L raised to power minus k which is a constant.

(Refer Slide Time: 18:59)

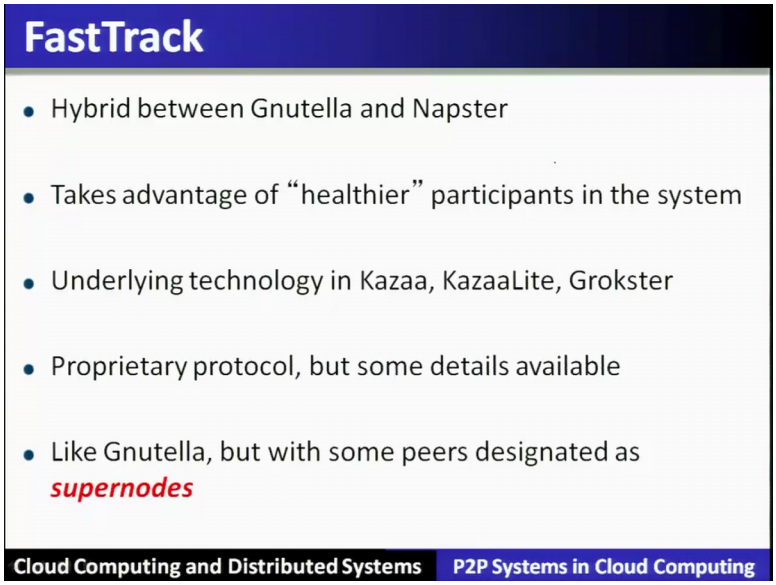


Here the problems in the Gnutella is that ping pong constitutes a 50 percent of the traffic which can be solved with the help of multiplexing. Similarly, the repeated searches with the same keyword also can be reduced using a caching method and also there is a many host which do not have the enough bandwidth for passing the Gnutella traffic; so, we will see another solution which is called a FastTrack system.

(Refer Slide Time: 19:36)

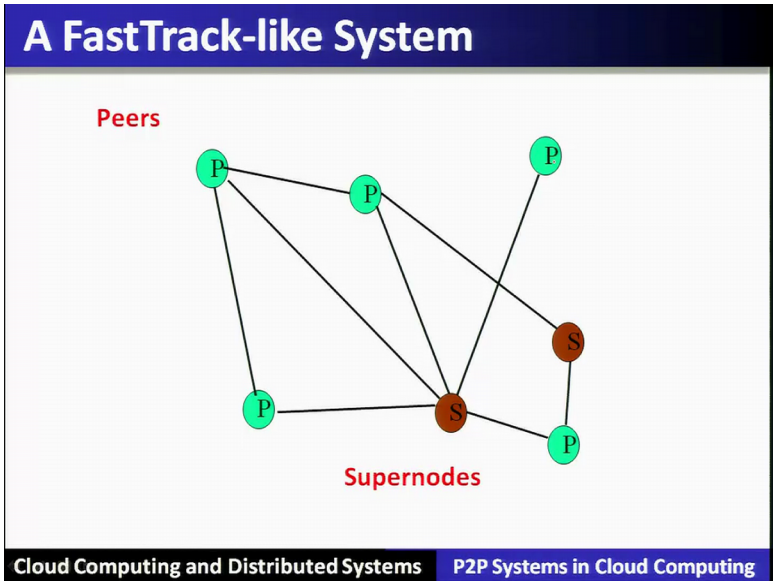


(Refer Slide Time: 19:41)



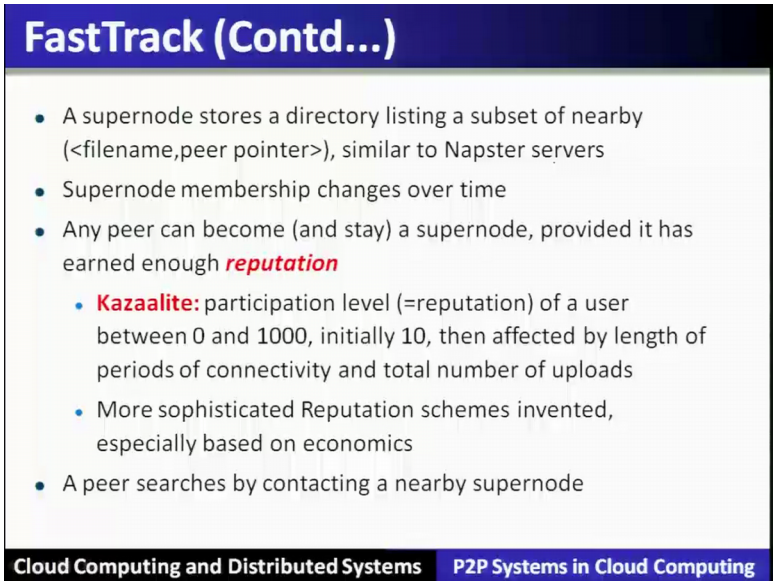
So, with the fasttrack system this is an hybrid of Gnutella and Napster to take the advantage of the servants which are basically the healthier in the participation; that means, whose reputations are high as far as uploading is concerned.

(Refer Slide Time: 20:04)



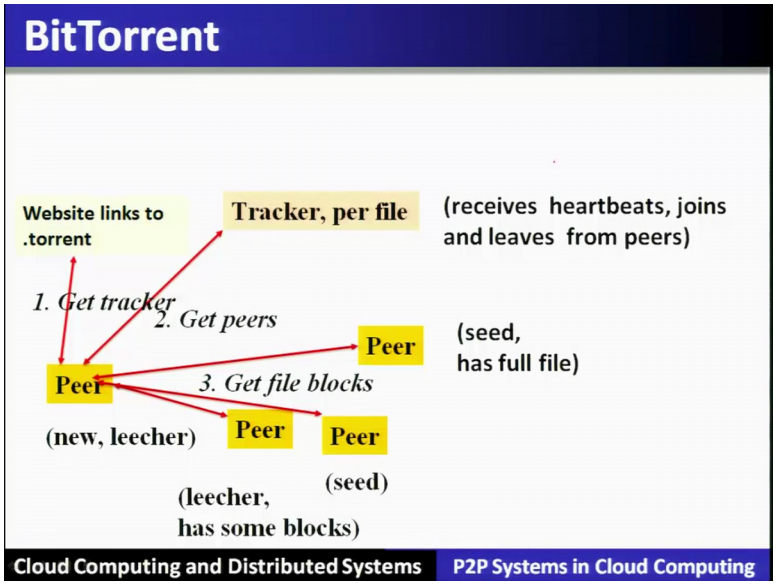
So, fasttrack is like Gnutella overlay structure, but here there will be a heterogeneous kind of nodes; that means, some nodes which are called supernodes will have more responsibility than the normal peers.

(Refer Slide Time: 20:17)



So, supernodes stores the directory listing a subset of the nearby host similar to the Napster and supernode membership changes over the time; it is not any peer can become a supernode, but the peer which is having an enough reputation can become the supernode. So, peer searches by contacting a nearby supernode here in this case.

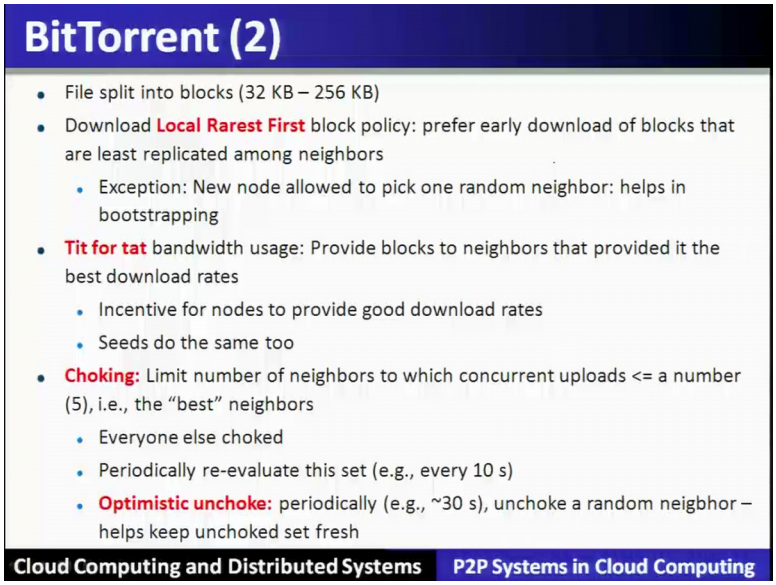
(Refer Slide Time: 20:48)



Now, comes the next peer to peer system which is called a BitTorrent. So, BitTorrent has used various reputation mechanisms to ensure that more number of peer should involve in this particular system. Therefore, the peers are given different responsibilities that is the peer which is having the full file at that node stored is called a seed and the peer which is having some blocks of a file, but not full file is called leecher.

And when a new peer joins which does not have these particular store files; they are called again the leecher and also for every file there is a tracker where the files are stored.

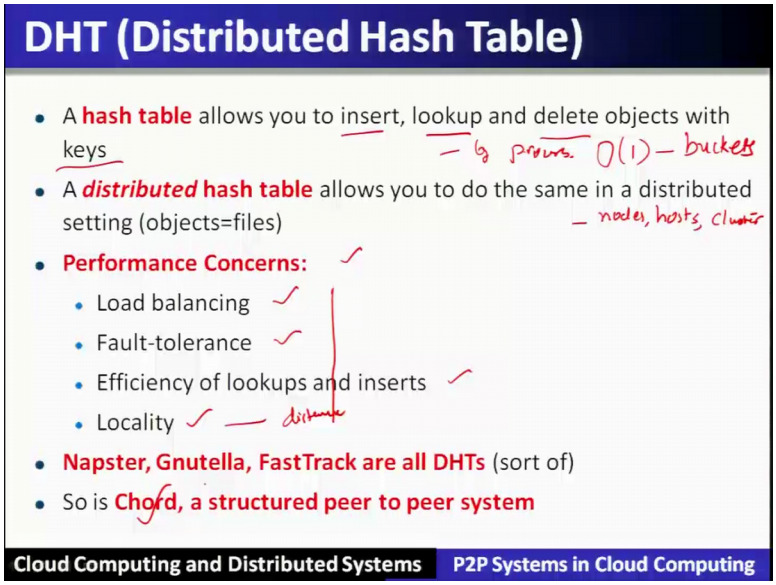
(Refer Slide Time: 21:48)



BitTorrent is split the file into the blocks of size 32 kilobyte to 256 kilobytes and it applies the algorithm which is called the local rarest first block policy; that means, it will first prefer the early downloads of a blocks that are least replicated among the neighbors.

Then it applies another algorithm which is called tit for tat for bandwidth usage; that means, it provides the blocks to the neighbors that provided at the best download rates. So, whosever is providing the best downloads in this case, it will also be supported in that manner that is the incentives mechanism. Third algorithm is called which is used here in BitTorrent is called choking that is it will limit the number of neighbours to which the concurrent uploads are handling up to 5 to avoid the choking.

(Refer Slide Time: 22:59)



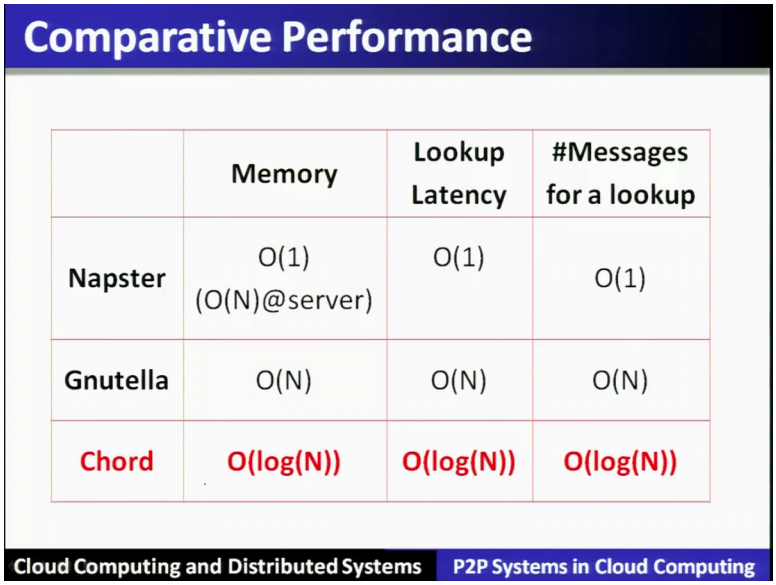
Now, we will discuss the distributed hash table. So, we have seen the hash table which is maintained in particular process and which supports the 3 different operations called insert lookup. And it will delete the objects with the key values that we have seen is maintained by a particular process.

The idea of hash table is to support these operations very efficiently let us say with the order 1 complexity. However, on the other hand the distributed hash table also allows the same set of operations insert lookup and delete, but in a distributed settings that is it is not confined in one process, but it is to be maintained over the distributed systems. Now, here in the hash table these keys are stored in the form of buckets, but here in the distributed hash table; these keys are stored on the nodes or you can also call them as a hosts or the cluster.

So, cluster can be very big; now as far as distributed hash table is concerned it has to deal with the performance concerned that is a perfect load balancing has to be ensured fault tolerance. In the sense if the nodes are failing and a new nodes are joining; then it should continue to give the services of insert, lookup and delete operations. Similarly, the efficiency of lookup and inserts are to be ensured and the property of locality; that means, whenever insert lookup and delete insert lookup operations are supported it has to be supported with the nearest cluster node in a close proximity that is with a smaller distance these insert and lookup they are being supported.

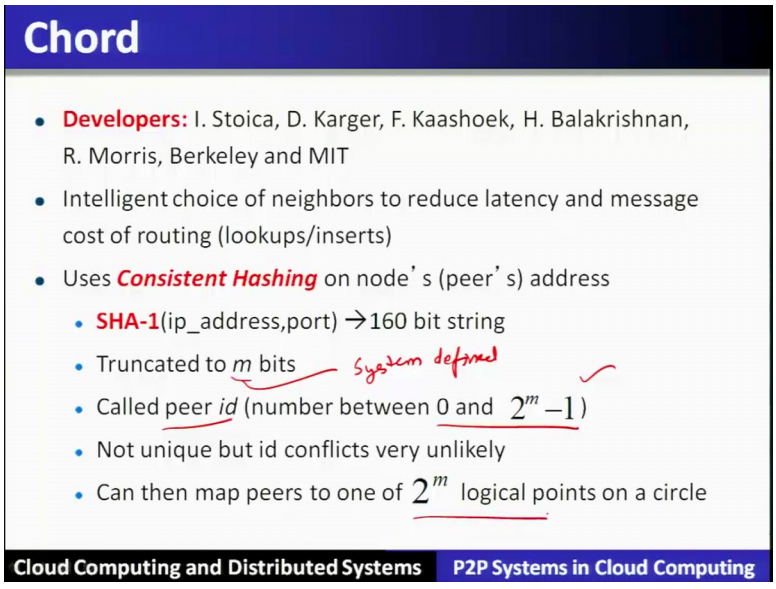
Now we have seen Napster Gnutella and fasttrack they are some sort of DHT, but not exactly DHT because here these performance were not a prime concern. Therefore, we will see the chord protocol peer to peer which is a structured peer to peer system which is a real sense is a distributed hash table why? Because it addresses all the performance concerns and on the other hand it supports an efficient insert, lookup and delete of these particular system.

(Refer Slide Time: 26:08)



Now, as far as comparative performance of Napster Gnutella and chord if we compare here the memory which is required here for the chord will be of the order log N the lookup will be of the order log N and the number of message for the lookup also is of the order log N which is quite improvement compare to the Gnutella system which is of the order N.

(Refer Slide Time: 26:42)

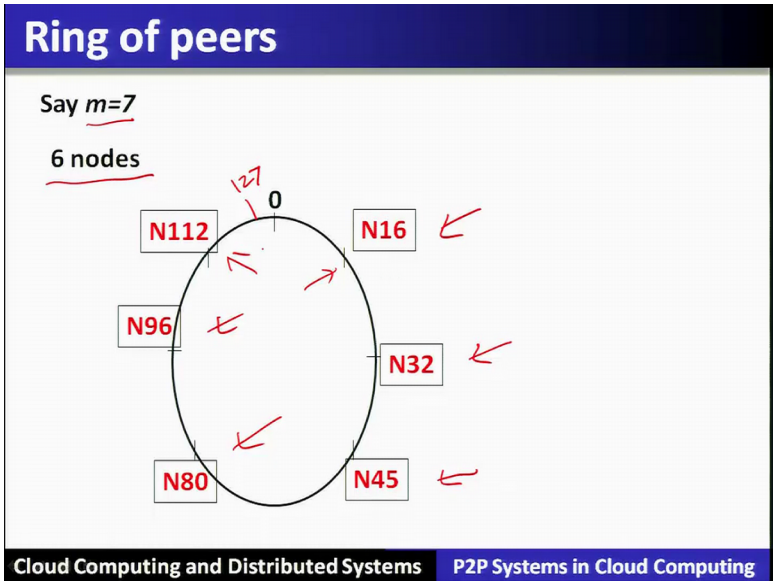


So, let us discuss the chord system chord is developed by the Berkeley and MIT scientists; here the chord provides a intelligent choice of the neighbours to reduce latency and the message cost of routing or lookups for inserts.

The chord uses consistent hashing on the peer nodes address; chord uses consistent hashing on the nodes address. So, it does using the SHA 1 function which will take IP address and the port address and will give 160 bit string which will be truncated to m bits; m bits m is the system defined parameter. These particular m bits will generate the numbers ranging from 0 to 2 raised power m minus 1 and these numbers are called peer id's.

These particular peer ids are not unique, but it will avoid the conflict if this particular numbers are large compared to the; to the peers. So, it can map the peers to one of these 2 raised power m logical points in the form of a circle.

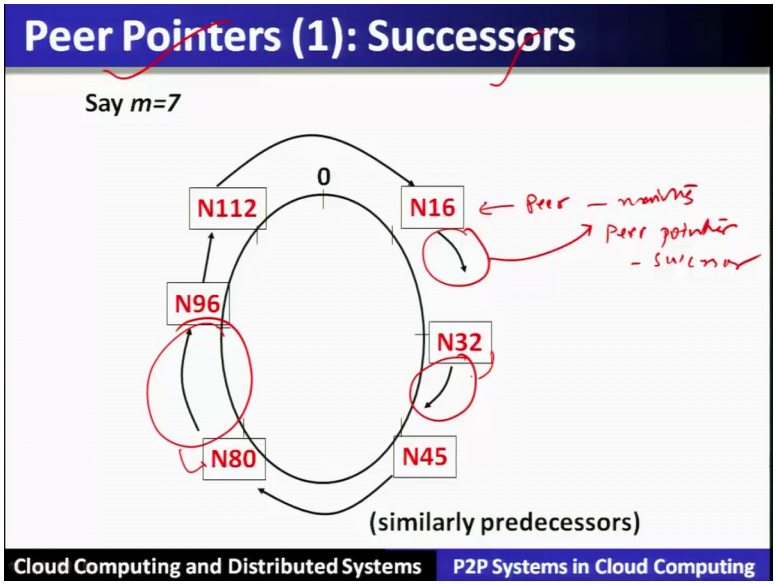
(Refer Slide Time: 28:42)



Let us take the example of this way of organizing the peers, let us say m is equal to 7 and we have less then less number of nodes that is only 6 nodes are there. So, m is equal to 7; that means, here comes 127 after 127; the 0, 1, 2 and so on they will start.

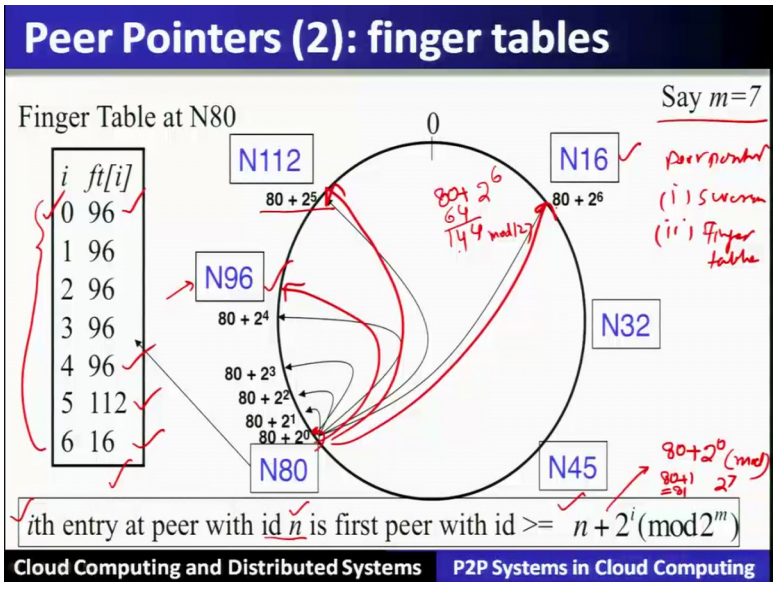
So, 16 will come and join at this particular location, node number 32 will join here at that location 45, 80, 96, 122 will be joining in the form of a ring.

(Refer Slide Time: 29:30)



Now, every peer will maintain a peer pointer which is nothing, but a successor; peer pointer is the successor. Similarly, the predecessors can also be there, but chord uses only the successor. So, every peer will maintain the pointer of its successor and this will constitute a logical ring structure.

(Refer Slide Time: 30:13)



So, besides successor; another form of peer pointer which is maintained is called a finger table, here the size of the finger table is limited to the size of m here let us say that m is equal to 7. So, the size of finger table is starting from 0 to 6 that is total 7 entries are there. The finger table uses the rule which says that the ith entry at the table with id n is the first peer id which is greater than or equal to this equation that is n plus 2 i mod m.

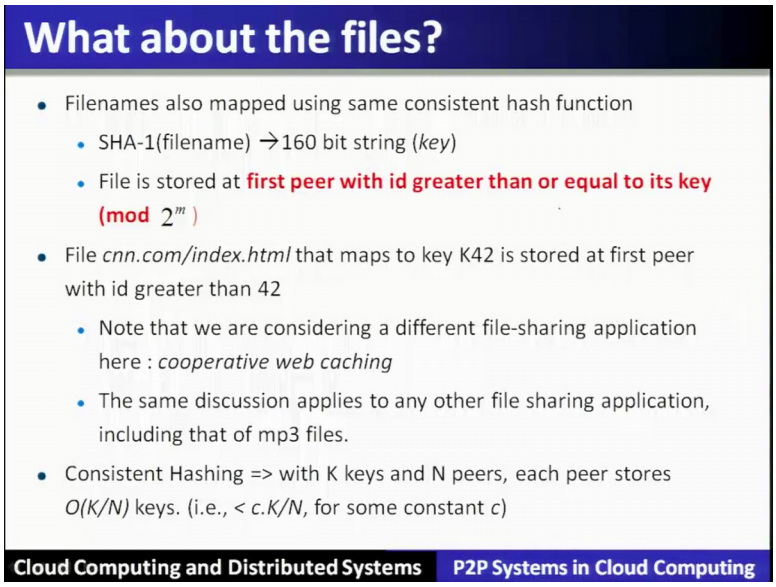
For example for peer with ID 80 this particular table is constructed where in the 0th entry; i is equal to 0. So, that becomes n is equal to 80 plus 2 raised power 0; mod 2 raised power 7 that is 80 plus 1 that is

81. So, 81 will be next to the N 80; so as far as this rule is concerned. So, it will be pointing to the first peer with the id greater than or equal to that id that is 81. So, greater than 81 the first particular peer will be 96; so 96 will be stored and this way up to 4 entries will basically be stored at 96.

When fifth entry will come; the value will be exceeded 96, but it will less than 112; so, 112 will be stored according to this particular rule. Now when i is equal to 6 that is 80 plus 2 raised power 6 that is 64; 144 mod, 127 that comes out to be 16 and it will be stored in the 16; in this way other nodes can also form their own finger table using the same rules.

So, there are 2 different peer pointers which are stored here in the chord; the first one is called successor, the second one is called finger table. The finger table if you see is growing exponentially so, that the search becomes faster search in the sense the routing becomes faster.

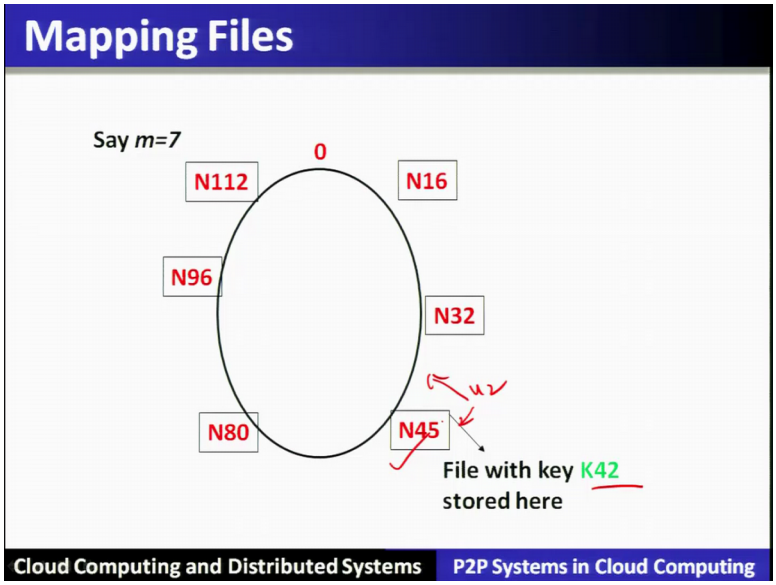
(Refer Slide Time: 33:38)



So, the finger table is used here in the chord for routing of the search keys for various operations. So, that was the indexing or the mapping of the nodes to the ring structure; what about the files? Files are also mapped using the same consistent hash function. So, the file is stored at the first peer with the id greater than or equal to its key mod 2 raised power m.

So, let us take an example that a particular file with the key K 42 which is hashed and obtained K 42 is stored at the first peer id, which is greater than 42.

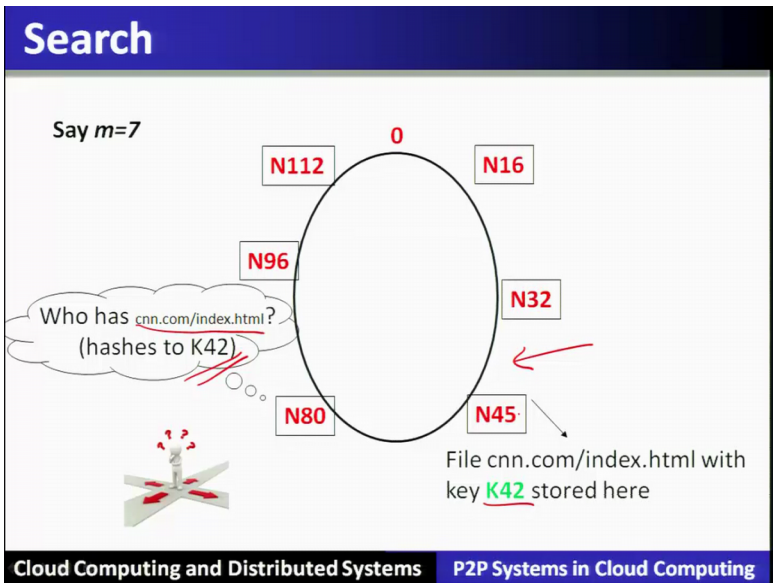
(Refer Slide Time: 34:29)



Here in this case let us say the 42 lies over here. So, the next node which is higher than 42 is 45. So, the file with the key 42 will be stored on this particular node. Now this particular method is not only used for the file sharing application, but for other applications such as cooperative web caching.

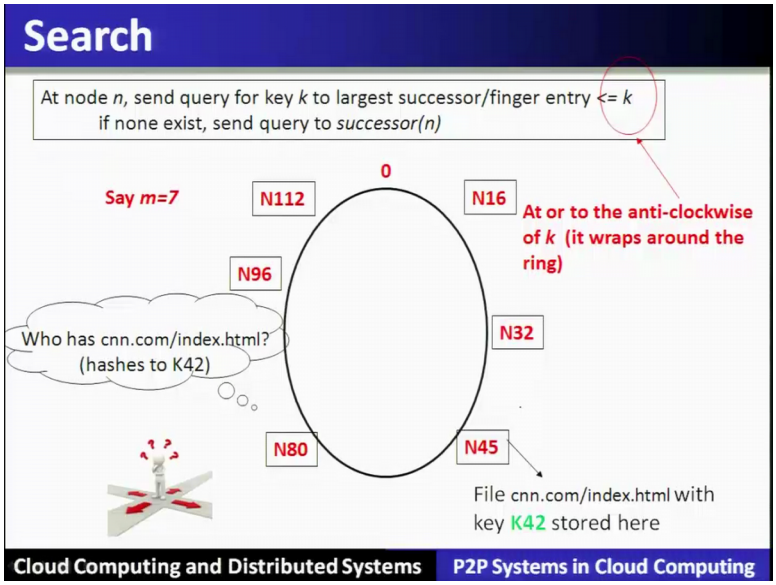
Now it uses the consistent hashing which ensures that the system with K keys and N peers will support of the order of K by N different keys per node; that means, that is bounded by some constant therefore, the load balancing is ensured here in the consistent hashing.

(Refer Slide Time: 35:27)



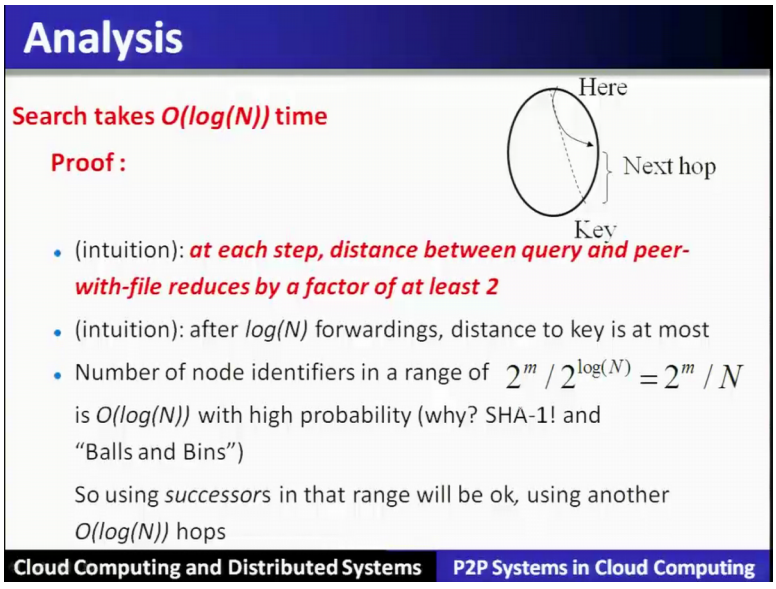
The search let us see how it happens. So, if the key if this is the file which want to be searched. So, applied to the consistent hashing let us say it generates a key 42. So, key 42 is basically the key of that file which want to access or you want to search. So, key 42 lies over here; so the next node which stores K 42 is N 45 and it will be solved in this particular manner.

(Refer Slide Time: 36:17)



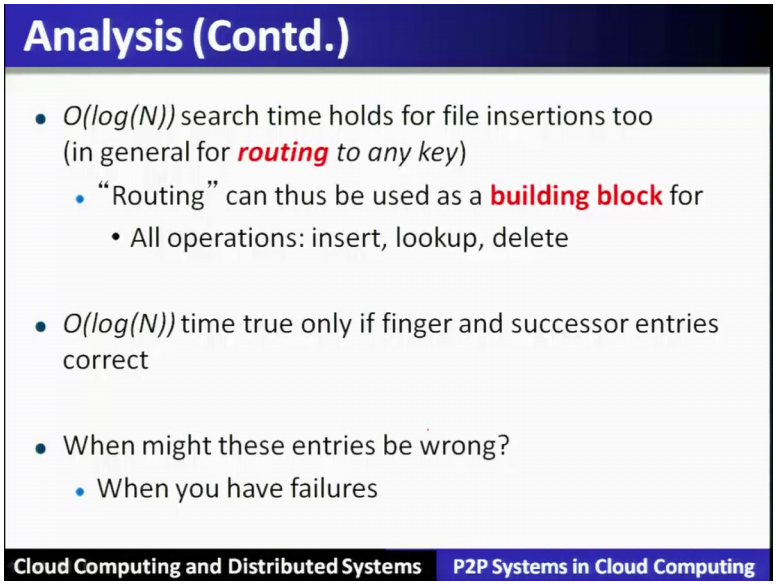
So, at node n search query for the key to the larger successor or to the finger table entry which is less than k is being used and if none of these contains that particular file then search query is passed onto the successor and so on.

(Refer Slide Time: 36:39)



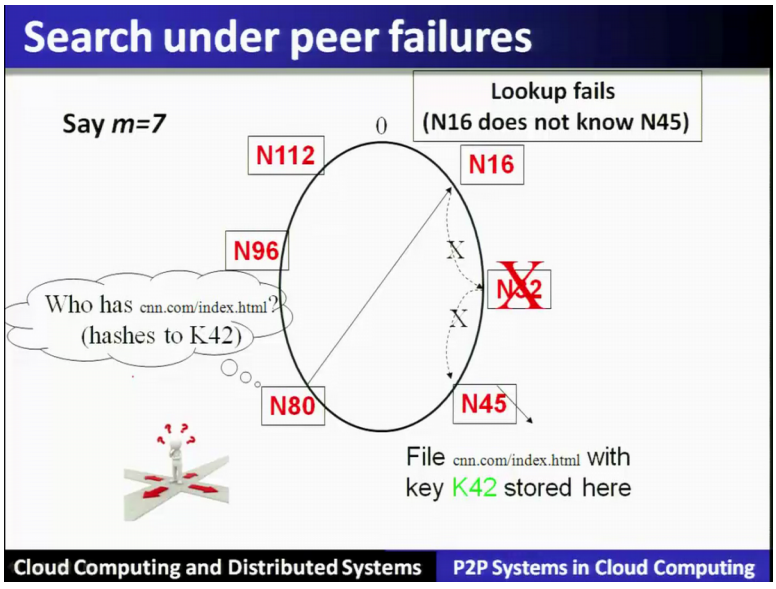
Let us see the analysis the search takes of the order log N time. So, here as far as the node is concerned every time in the search between the next hop and the key; the distance is reduced by a factor of at least 2. So, after log N forwarding the distance to the key is at most 2 raised power m divided by 2 raised power log N that is nothing, but 2 raised power m divided by N, which is of the order log N with the high probability. So, using successors in that range will be using another order of log N hops.

(Refer Slide Time: 37:33)



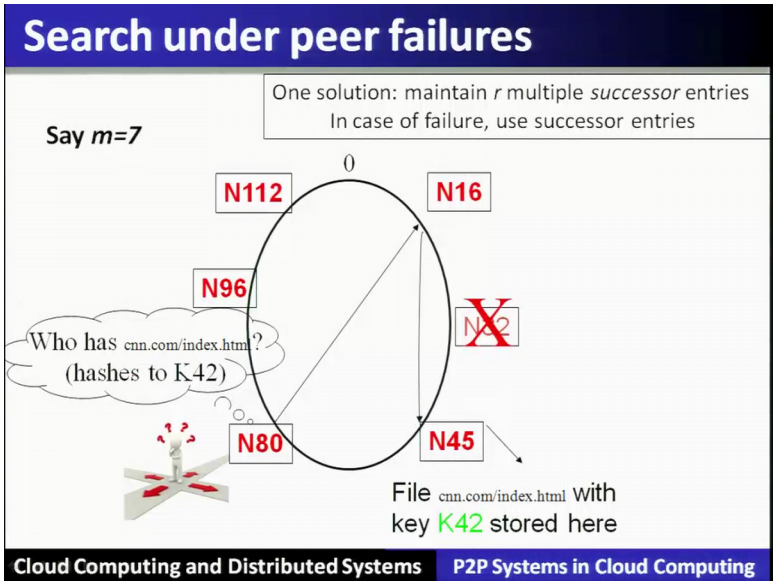
Now, this particular search which takes of the order log N; it is also required for the file insertions, but this particular search time that is of the order log N is valid if the finger and the successor entries are correct; that means, if what happens when these entries are not correct; that means, the nodes maintaining them are down; that means, the failures.

(Refer Slide Time: 38:06)



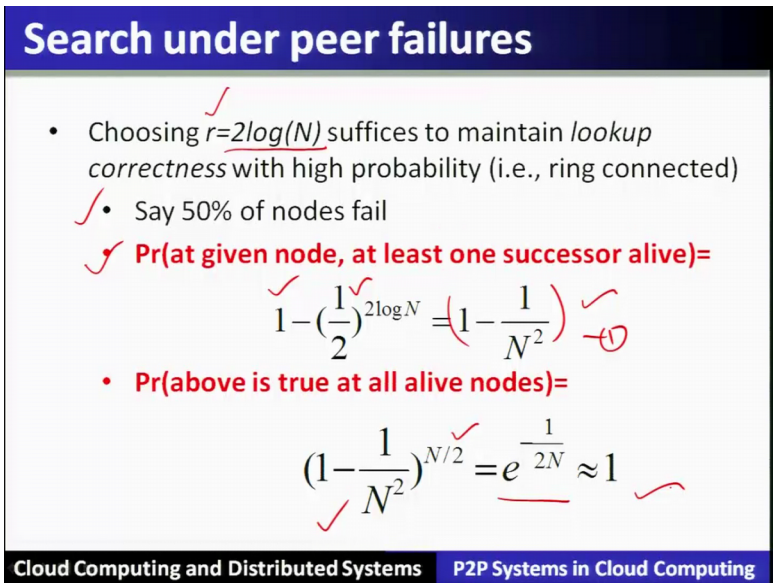
Let us see here in this example if the node 32 which contains all the information is down.

(Refer Slide Time: 38:18)



So, we have to deal with this particular solution by maintaining the successor entries by maintaining r different multiple successor entries instead of 1 to deal with the failures.

(Refer Slide Time: 38:36)

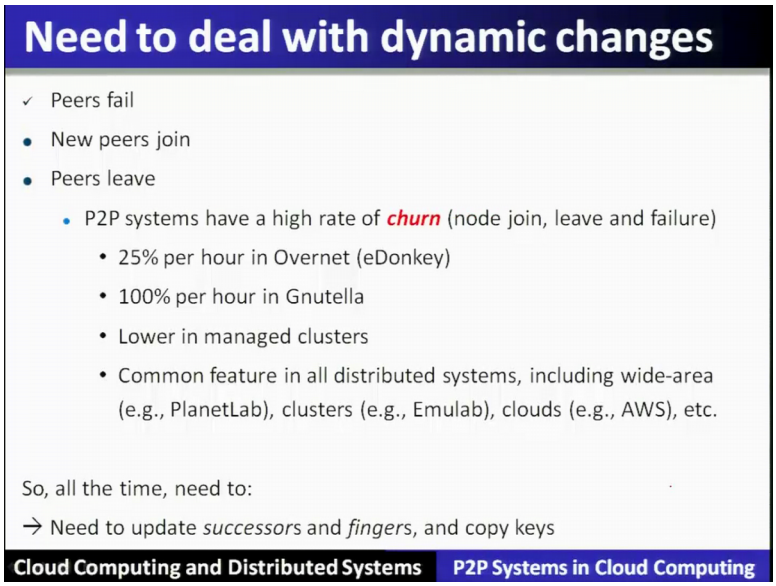


Now, the question is how many successor entries are required to deal with the peer failures? Here, the number of successor entries which are maintained to deal with the peer entries is 2 times log, which suffices to maintain the lookup correctness with the high probability.

Let us see this particular aspect now let us assume that 50 percent of the nodes fail which is very high percentage normally not it will not reach to that 50 percent not fail, but let us assume that. So, let us find out the probability that at a given node at least one successor is alive. So, the probability that the successor is failed is 1 upon 2 times 2 log N and the probability that at least one successor survives is 1 minus that figure that comes out 1 minus one upon N square.

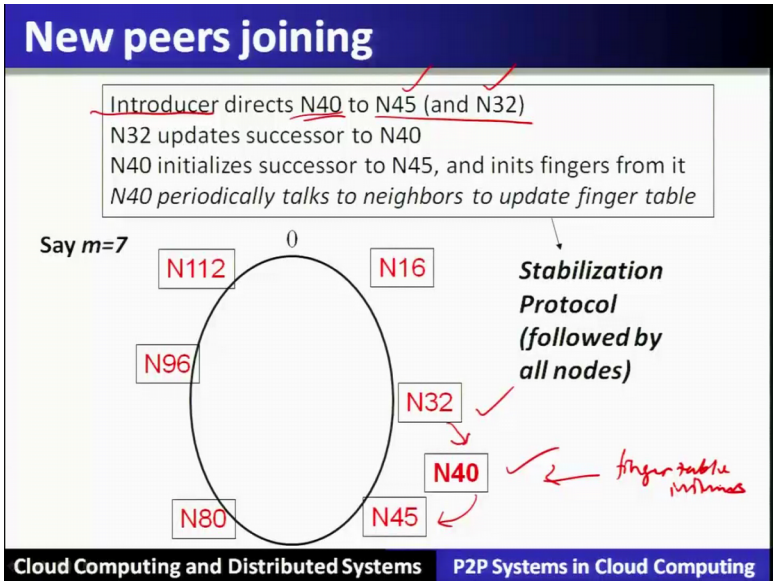
Now, probability that this is true at all alive nodes; so we will use this particular value here N raised to the power N by 2 that comes out to be e raised power 1 upon 2 N that comes out to be 1 that is if the number of nodes is too large then this becomes 1. So, these are the examples given for the understanding.

(Refer Slide Time: 40:24)



So, here we have to deal with the dynamic changes; that means, when the peer fails or when a new peer joins or the peer leaves; the peer to peer system; that means, has if it is having high churn then it needs to update the successor and the finger table and also copy some of the nodes.

(Refer Slide Time: 40:44)

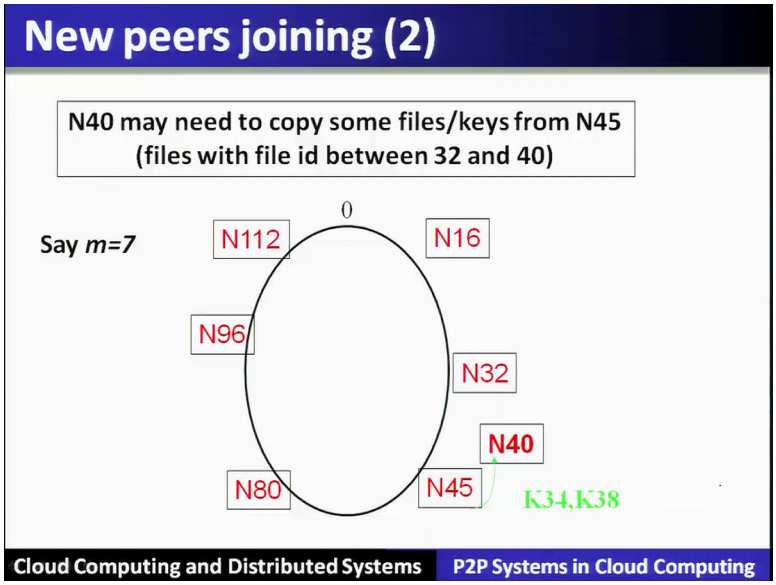


Let us see the example how the new peers join in chord system, here let us see the example that the new peer N 40 wants to join. So, it has to contact to the introducer and introducer will give the addresses of some of the peers. Here let us assume that the 2 addresses the introducer gives to N 40 they are N 45

and N 32. So, N 32 will updates its successor as N 40 and then N 40 initializes the successor to N 45 and initialize its finger table from it.

N 40 periodically talks to the neighbours to update its finger table for that it uses a stabilization protocol and it will try to ensure the proper updation.

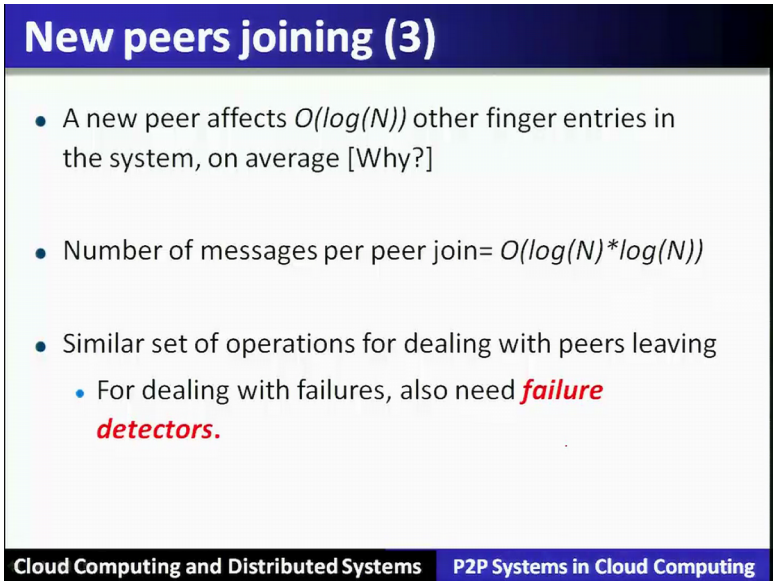
(Refer Slide Time: 42:05)



Besides this N 40 may also need to copy some of the files or the keys from N 45 that is the files with the ids between 30 and 40. So, between N 32 and 45 all the files or the keys are required to be divided here in N 40 for example, N 40 will be storing the keys between N 32 and N 40.

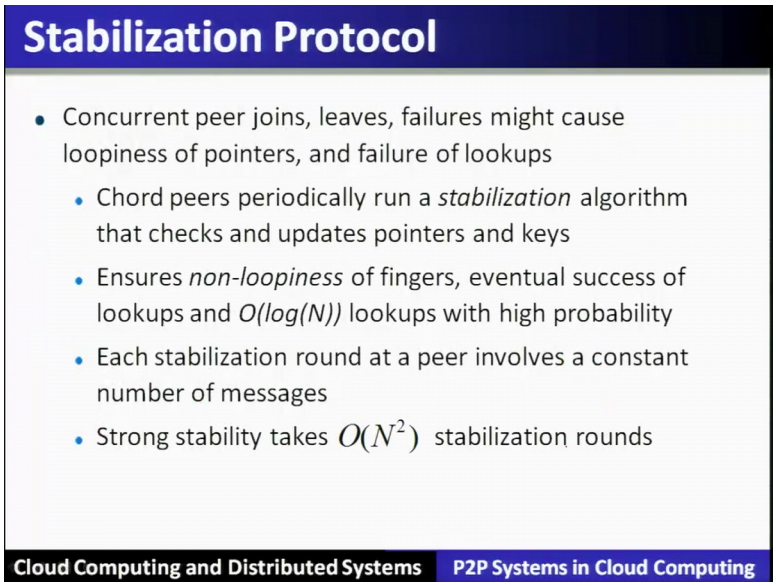
So, for example, the keys K 34 and K 38 will be copied here which was earlier maintained by N 45 and this will be copied here in N 40.

(Refer Slide Time: 42:57)



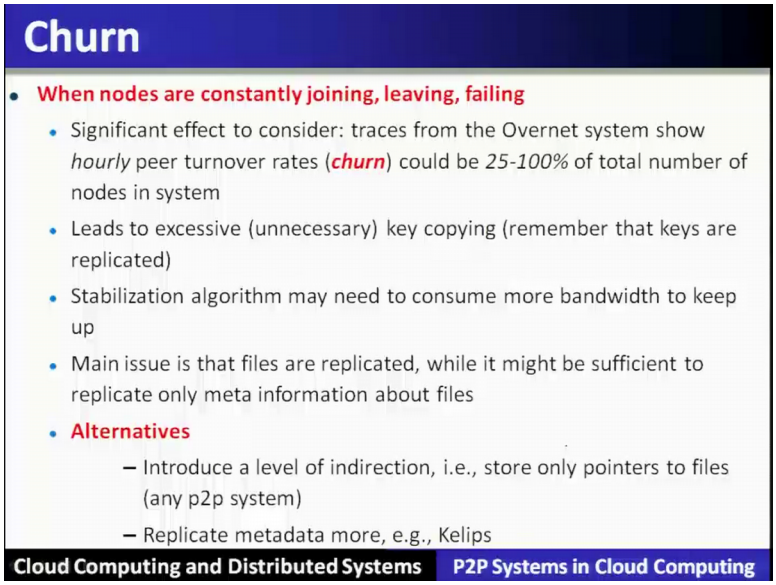
So, this way the new peers will join. So, that the new peer affects of the order log N other finger table entries in the system on an average. So, the number of messages per peer join will be of the order log N times log N; so, that will be of the order log N square. So, similar set of operations are required to deal with the peers leaving the systems or dealing with the failure detectors.

(Refer Slide Time: 43:28)



So, stabilization protocol the concurrent peer joins leaves and failures might cause loopiness of the pointers and failures of the lookup, periodically runs this stabilization protocol and checks the updates the pointers and keys.

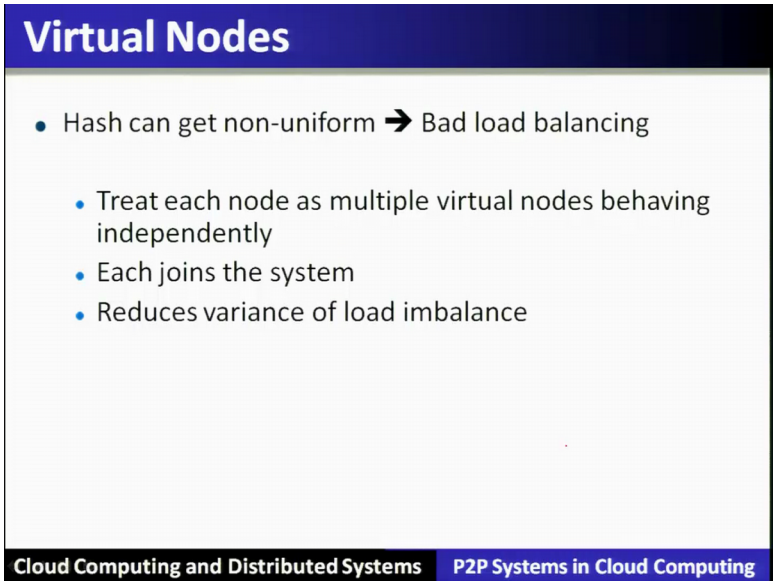
(Refer Slide Time: 43:55)



So, strong stability takes of the order N square stabilization round. So, churn in the chord system sees that the nodes are constantly joining and leaving; here it leads to the excessive key copying and stabilization protocol also requires the bandwidth. So, the issues; is that the files are replicated. So, the

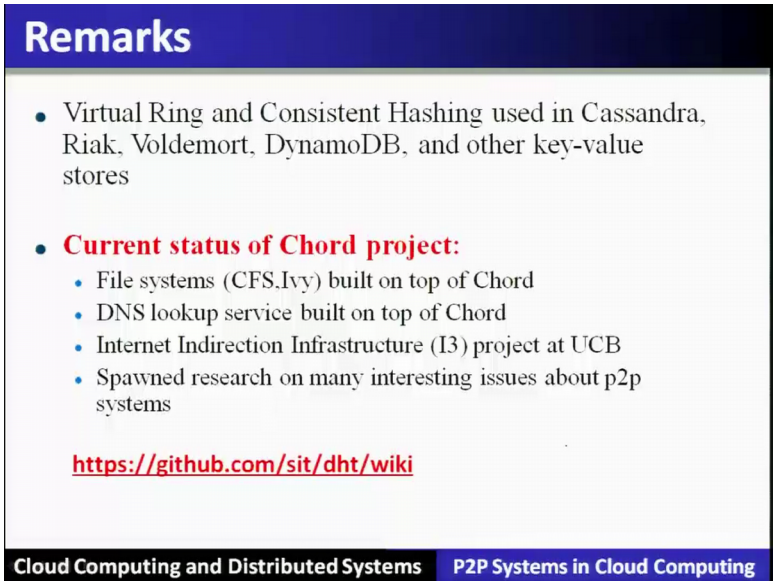
alternative is to provide another level of indirection store only the pointers instead of replicating the entire files. So, this particular issues will be handled in the next peer to peer system that is called Kelips.

(Refer Slide Time: 44:36)



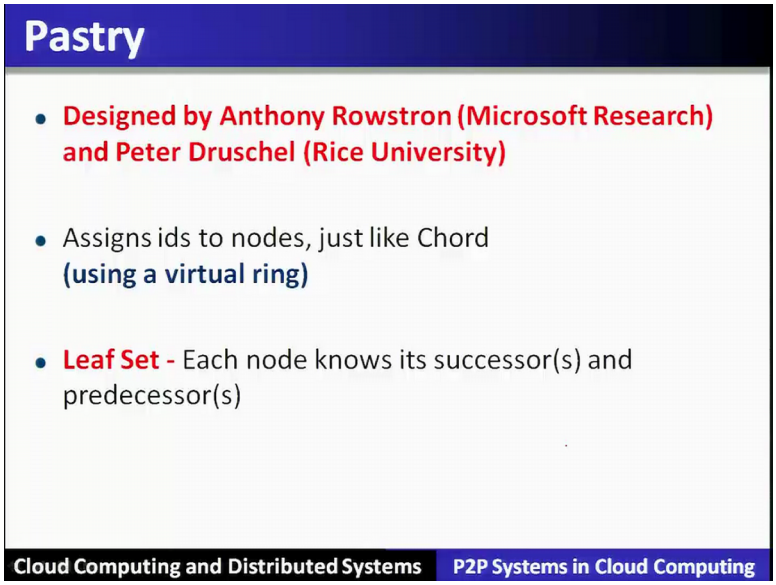
Here there is a concept of the virtual nodes for proper load balancing. So, if there are too many number of keys are stored at a particular node; then there is a concept of virtual nodes that they may be partitioned into the virtual nodes and hence it will introduce the load balancing.

(Refer Slide Time: 44:55)



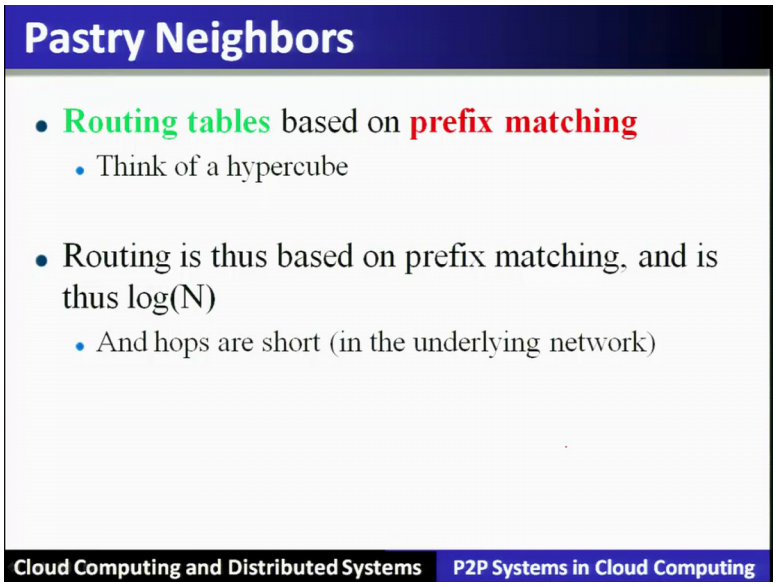
Therefore the concept of the virtual ring which was given here in the chord system and also the concept of consistent hashing which was used in the chord system; virtual ring and consistent hashing together these concepts are used in the modern key value store that is in the cloud computing systems like Cassandra, Riak, Voldemort and DynamoDB that we will see in the further lectures.

(Refer Slide Time: 45:26)



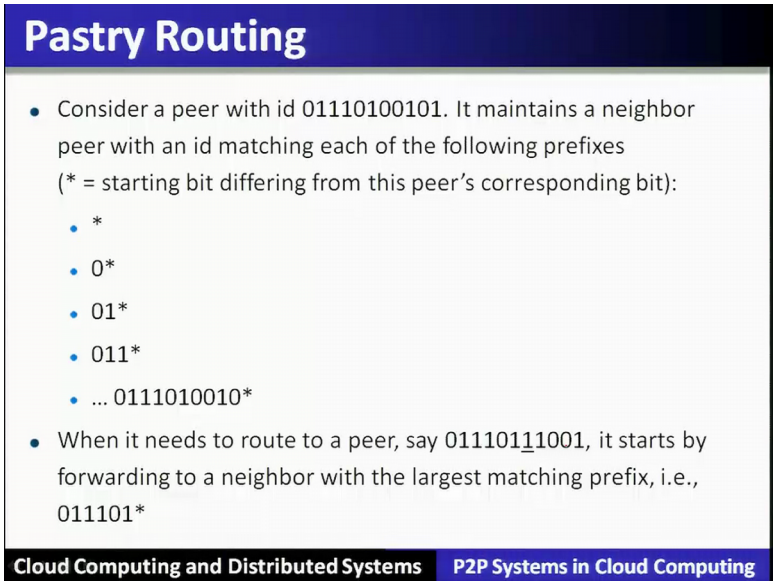
Now, we will see another peer to peer system that is called pastry, which is developed by the Rice University and Microsoft research. Pastry also assigns ids to the nodes just like the chord and uses the virtual ring, here there is a leaf set that is each node knows its successors and predecessors.

(Refer Slide Time: 45:53)



But there is a difference here the routing table in pastry is based on prefix matching which is nothing, but the based on the hypercube structure. So, routing is thus based on prefix matching and is log N.

(Refer Slide Time: 46:16)

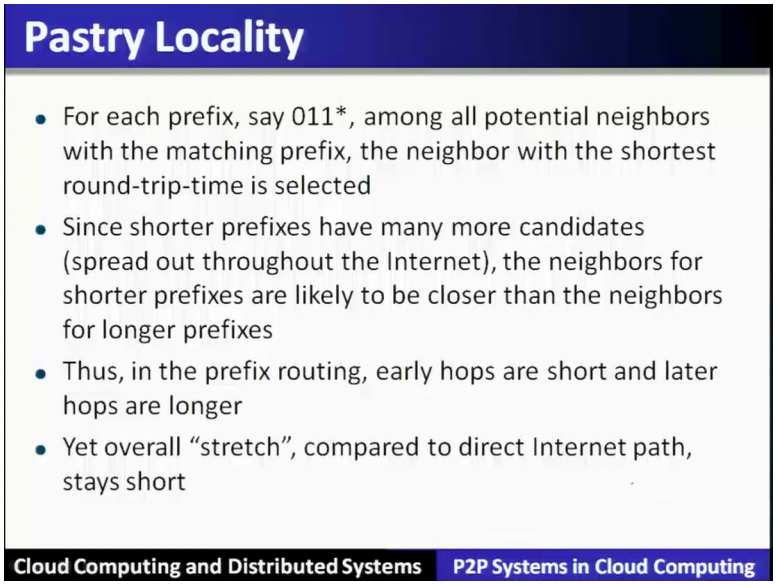


Let us consider how the routing in the pastry is done using prefix matching; consider a peer with the id shown over here and it maintains the neighbour peers with an id matching each of the following prefixes.

So; that means, star means that it is by starting bit differing from this peers corresponding bit for example, if it is starting with a star; that means, if the bit is 0 so; that means, it is starting from 1; that means, its neighbour. Similarly 0 star means the neighbours first bit is matched with this peer 0, the second bit should be 0 instead of 1 in our peer.

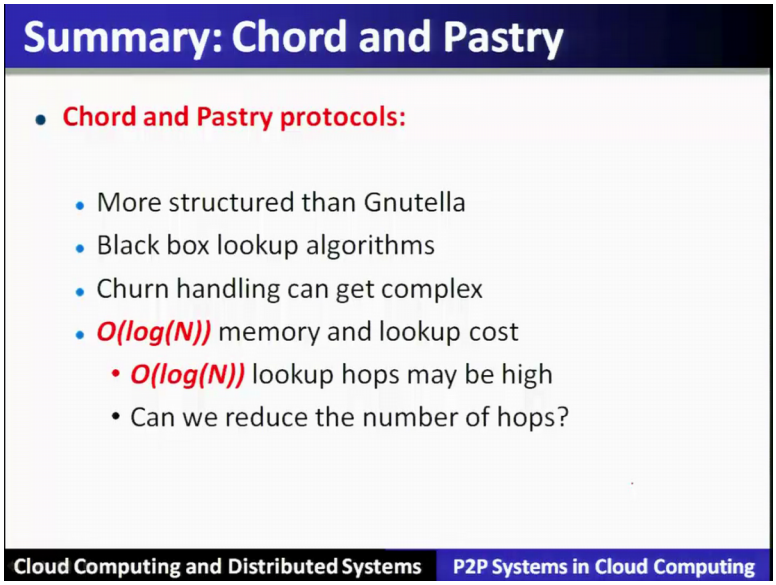
So, this way the one which is having the largest prefix match will be will be the neighbour and therefore, in the routing it will be forwarded to that particular neighbor.

(Refer Slide Time: 47:23)



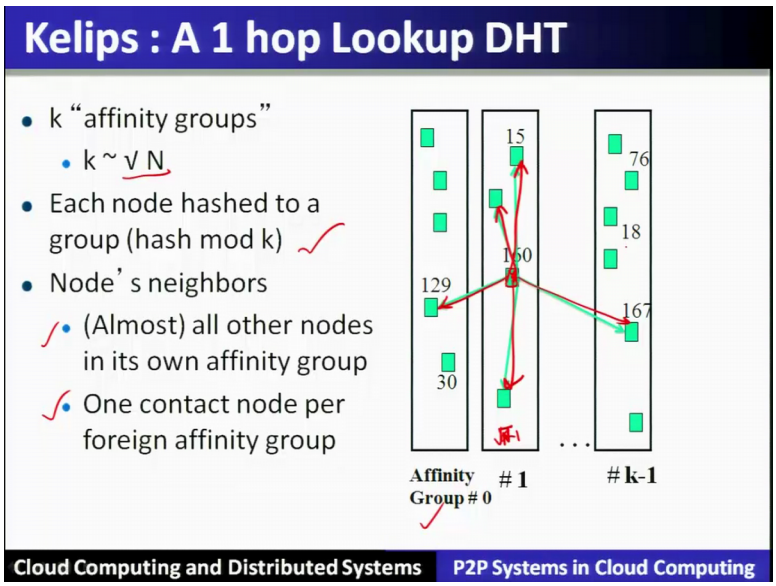
So, using this prefix matching the neighbour with the shortest round trip time will be selected; using the properties of this structure that is nothing, but the hypercube. Since the shorter prefixes have more candidates the neighbour for the shorter prefixes are likely to be closer than the neighbours for the longer prefixes. Therefore, prefix routing provides and that is the early hops are shorter and the later hops are longer; overall stretch compared to the direct internet path stays short.

(Refer Slide Time: 48:04)



So, if we compare the chord and pastry; both are more structure and churn handling can complex the memory requirement is of the order log N and lookup host is of the order log N can be reduced this number of hops.

(Refer Slide Time: 48:26)

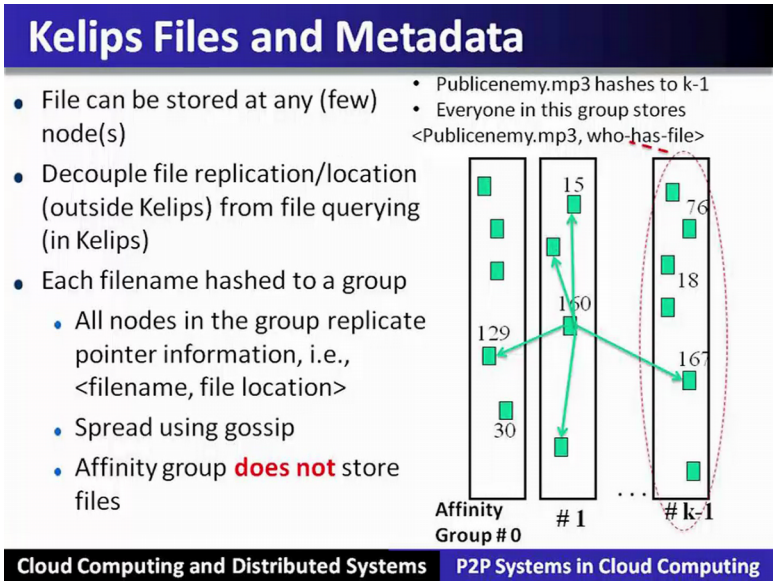


So, the next peer to peer system is called Kelips is to provide one hop lookup DHT method; that means, instead of log N in the previous methods; it will provide of the order 1 hop lookup that is Kelips. So,

Kelips supports affinity groups; so, it supports k affinity groups. So, k is of the order root N. So, here affinity group are number from 0 to k minus 1 and the value of k is root N. So, each node is hashed to one of these groups that is hash mod k.

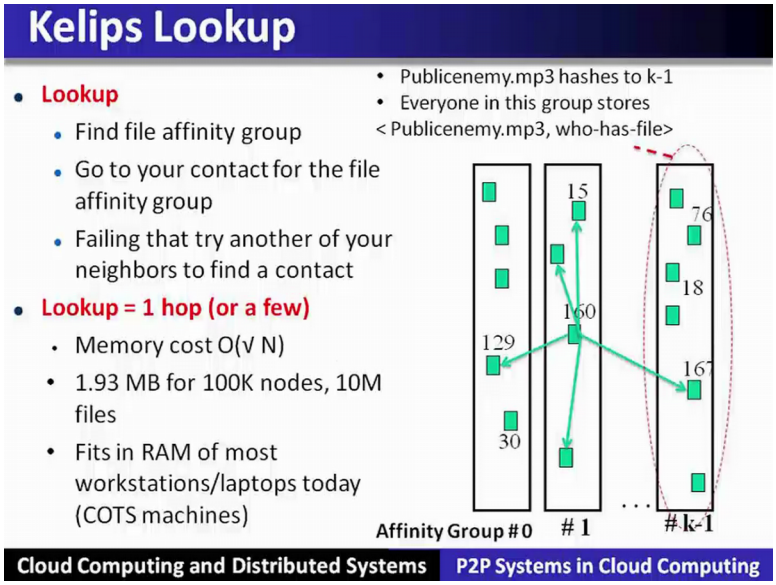
Now, nodes neighbours are almost all the nodes in that affinity group for example, 160 is having the neighbour with its other group members that is in this affinity group. So, they maybe k minus 1 because this membership is root N. Beside this this particular node is also having one contact node for foreign affinity group. So, this is; so this is the affinity group its neighbour affinity groups; that means, with all other affinity group with one member it is also having in its neighbourhood lists.

(Refer Slide Time: 50:13)



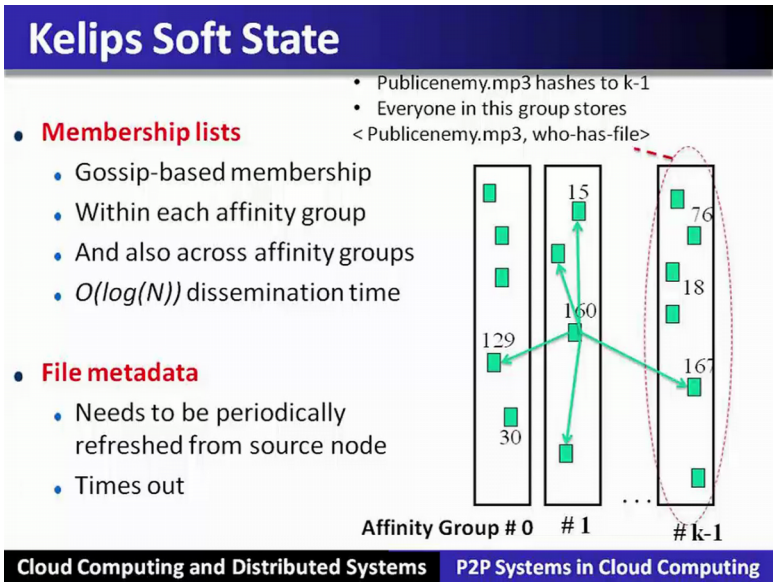
So, therefore, let us see how the file is stored the file can be stored at any of these few nodes. Now let us decouple the file replication with the location from querying in the Kelips. So, each file name is hashed to a group and all the nodes in the group replicate the pointer information not the entire file; this information is spread using the gossip protocol.

(Refer Slide Time: 50:48)



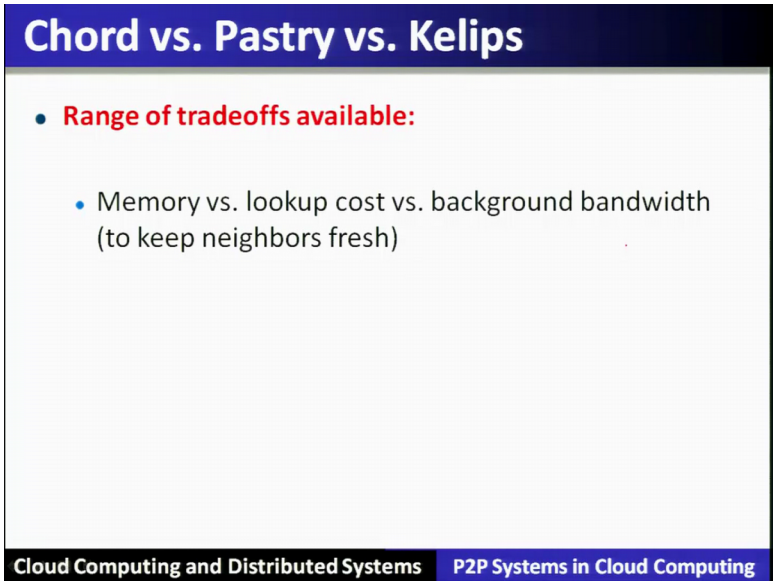
So, affinity group does not store these particular files; similarly as far as the lookup is concerned it will find the affinity group and go to the contact for the file affinity group failing, that try another neighbour to find out the contact. So, lookup will be of the order that is of the order 1; so, 1 hop.

(Refer Slide Time: 51:18)



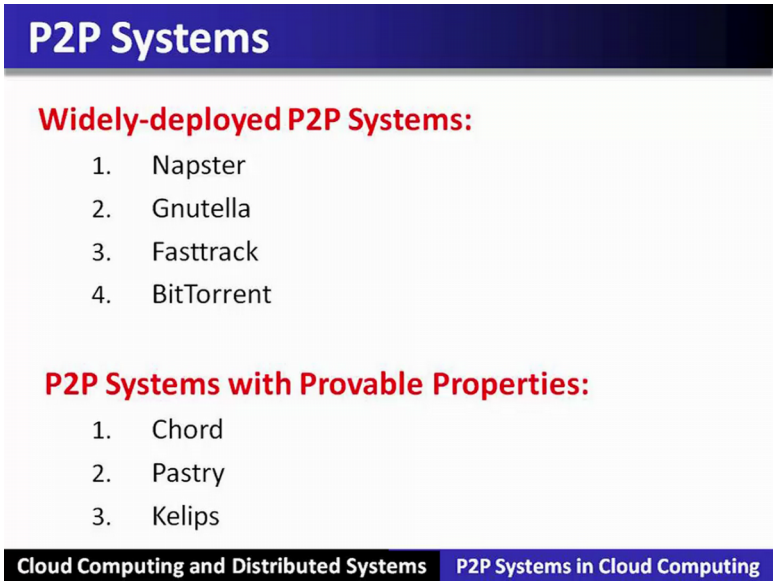
Memory cost is of the order root N; membership lists are being maintained using gossip based membership protocol within each affinity group is maintained and also across the affinity group and; that means, it requires of the order log N dissemination time; file metadata needs to be periodically refreshed and when time out.

(Refer Slide Time: 51:39)



So, as far as if we see the trade off with chord pastry and Kelips. So, it is the memory versus lookup cost versus background bandwidth to keep.

(Refer Slide Time: 51:53)



So, in conclude we have seen that there are widely deployed peer to peer systems Napster, Gnutella, Fasttrack and BitTorrent; they differ and they improve further up to the BitTorrent. We have also seen the peer to peer systems came out from academia with all provable properties such Chord, Pastry and Kelips.

Thank you.