MS&E 235, Internet Commerce

Stanford University, Winter 2007-08

Instructor: Prof. Ashish Goel, Notes Scribed by Shrikrishna Shrin

Lecture 14: Internet Architecture Contd.

Note:

TCP requires the following information: Source IP address, Destination IP address, Source Port No. and Destination Port No. This is the reason why web servers are able to identify applications and computers uniquely and are able to process many connections simultaneously. However, UDP only requires the Destination IP and Destination Port information.

NAT: Network Address Translation

An IP address consists 32 bits. Therefore, the total number of unique IP addresses that can exist is fixed. However, there is no shortage of IP addresses. This is due to NAT. A machine like wireless router also runs something called NAT (Network Address Translator). The NAT box has a unique 'real' IP address. However, computers that are connected to the router are assigned a 'fake' address by the router. The internet governing body mandates that the IP address 10.0.1.* is reserved for NATs. So the NAT box assigns these IP addresses to computers that are attached to it. It maps these IP addresses and port numbers used by applications to its own port number space. Therefore, you cannot run a server behind a NAT box without configuring NAT box to enable port forwarding. Moreover, even peers cannot run behind a NAT box. This is where skype has succeeded. In skype, there are nodes and super nodes. A few of the super nodes are run by skype. Most of the super nodes are normal nodes which are not behind a NAT box, and have been promoted to a super node. Each super node maintains an index from phone numbers to IP addresses.

Skype

In other P2P networks such as kaaza, it did not matter as to which node you connect to. However, when you want to make a phone call to a particular person, it becomes important that your computer is connected to his/her computer.

In Skype's architecture, super nodes are used to maintain a directory (mapping from phone numbers and Skype user accounts to IP addresses). Super nodes also act as intermediaries for forming connections. Typically, User A(skype instance used by user A) goes to a super node to find the address of B (skype instance being used by B), and then connects to B. However, this assumes that B is listening at some port. If B is behind a NAT or a firewall, A cannot contact B directly. In such a case, a connection is established by keeping As connection with the super node as well as B's connection with the super

node alive. When A wants to call B (who is behind a NAT), the super node sends a message to B asking it to contact A. B establishes a TCP connection with A, and then A calls B.

What happens when both A and B are behind a NAT or a firewall? UDP can be used as it is not connection oriented. If B has a connection with super node, anyone else can also send UDP packets to B on the same connection. This is known as UDP hole punching. B constantly sends UDP packets to some super node(s). If A wants to contact B, super node asks B to use a particular port for UDP and A uses the same UDP port to send message to B.

The only scenario when A and B cannot talk to each other is when both of them are behind a really restrictive NAT (which does not allow UDP hole punching). In this case they communicate through the super node S. A sends a packet to S, S sends the packet to B and Vice-Versa.

Search engine architecture

In order to search through billions of web pages efficiently, search engines use a reverse index. For every keyword, search engines maintain a sorted list of documents in which that keyword appears. This list is sorted based on some measure of reputation (an example being page rank). The challenge for search engines is to be able to find relevant search results for composite queries efficiently. It is impractical to main a separate index for each composite query as this would require prohibitive amount of storage. Furthermore, a large number of queries are unique and therefore maintaining indexes for these queries is not viable.

Suppose there are N documents and M words. d_i is the number of words in document i. w_j is the number of documents in which word j occurs. The total size of the index equals $\sum_{i=1}^{N} d_i$ which is the same as the size of the reverse index $\sum_{j=1}^{M} w_j$. The size of the index is therefore the same as that of the reverse index and this number is usually much smaller than the size of the data on the web and storing them is not a problem.

Suppose you search for the phrase 'purple island', the search engine has a list of documents in which the word 'purple' occurs ranked by reputation and another list of documents in which the word 'island' occurs again sorted by reputation.

Each word can be present in millions of documents and the size of the corresponding reverse index can be quite large. Finding an intersection between these large indexes is computationally impractical. However, since the search engine only needs to return a fixed number of results (1000 in the case of Google), it is sufficient to find a certain number of intersections. Additionally, the documents present in this intersection can be subjected to post filters to personalize the search results (maintaining a separate index for each user is impossible due to the scale of the problem).

In order for such an algorithm to work, the notion of reputation needs to be context independent. This is because if a web page containing both 'purple' and 'island' were to appear toward the beginning in the reverse index for 'purple'

but toward the end in the reverse index for 'island', it may take a long time to find this web page as the 'island' index has to be traversed before it is found. Also, other (and possibly less reputed) web pages might be returned as results instead.