Homework #2 (CSCI-651-01)

by Aseem Mehta

a. Bloom Filter works on the concept of set membership. In bloom filter an array of n bits is created, all of them are set to 0. Bloom Filter utilizes k- independent hash functions with hash values resulting between 0 and n-1.

Each value is passed through all the hash functions resulting in k hash values, All the values associated with these k values in array would be set to 1.

To check an element: the element would be passed through all k hash functions and the resulting values would be checked against the corresponding values in array, if even one value is not set to 1, then we can say the element is not present, otherwise the element may be present.

Bloom Filter provides the probability of presence of an element, it answers either an element is not present or may be present in the array.

As the bloom filters do not store all the values but some hash values are set to 1, the space required is lower. Hash functions could be run in parallel as they are independent of each other lowering the time taken to process.

b. Bloom filters can be used for IP lookup as follows

1. Separate bloom filters are maintained based on prefix length of input addresses.

2. Similarly separate hash table is maintained for every unique prefix length

3. All bloom filters are checked during lookup of an IP address. The process is performed in parallel. All matching prefix lengths are stored in a vector for lookup.

4. All tables are scanned for a match from greatest to lowest till either a match is found or no more values are left.

c. Under most circumstances the lookup performance is the same for both IPv6 and IPv4 as Bloom filter do not store the prefix but only the hash values are set to 1, however we may have to maintain many bloom filters and tables if there are many unique prefix lengths.

For some cases the performance of IPv6 and IPv4 lookup could be different when the distinct prefix values for both protocols is different.

Assuming data to be sent is pure 16,384 which doesn’t contain IP header lengths and all packets are TCP. Ethernet header = length of (8 Bytes Clock + 14 Bytes header + 4 Bytes CRC) = 26 Bytes

a. A will send 5 packets over network 1 consisting of packets P1 to P4 with 4010 bytes of data, ethernet header at 26 bytes, 40 bytes of IPv6 header and TCP of 20 bytes, totalling 4096 bytes, additionally a packet P5 with 344 bytes of data, ethernet header at 26 bytes, 40 bytes of IPv6 header and TCP of 20 bytes of header.

As the data reaches R, the router would realize the MTU of network 2 is lower and will inform A about the issue and ask for smaller packet size.

A in turn would resize and send back new packets based on the lower MTU value of network 2 consisting of packets P1 to P11 with 1450 bytes of data ethernet header at 26 bytes, 40 bytes of IPv6 header and TCP of 20 bytes, totalling 1536 bytes and another packet P12 with 434 bytes of data and 40 bytes of IPv6 header.

b. A would send 5 packets over network 1 consisting of packets P1 to P4 with 4030 bytes of data, ethernet header at 26 bytes, 20 bytes of IPv4 header and TCP of 20 bytes, totalling 4096 bytes, additionally a packet P5 with 264 bytes of data ethernet header at 26 bytes, 20 bytes of IPv4 header and TCP of 20 bytes.

At R packets P1 to P4 would be fragmented into 3 new packets for each P1 to P4, 2 of them with 1470 bytes of data, ethernet header at 26 bytes, 20 bytes of IPv4 header and TCP of 20 bytes, one packet with 1090 bytes of data, ethernet header at 26 bytes, 20 bytes of IPv4 header and TCP of 20 bytes.

In total on network 2: 8 packets would be sent with payload of 1536 bytes, 4 packets with 1156 bytes of data, additionally one non fragmented packet with 264 bytes.

c. A will send 5 packets over network 1 consisting of packets P1 to P4 with 4010 bytes of data, ethernet header at 26 bytes, 40 bytes of IPv6 header and TCP of 20 bytes, totalling 4096 bytes, additionally a packet P5 with 344 bytes of data, ethernet header at 26 bytes, 40 bytes of IPv6 header and TCP of 20 bytes.

R would encapsulate the IPv6 packets into IPv4 and then fragment P1 to P4 into 3 new packets for each P1 to P4, 2 of them with 1536 bytes(1470+66) and one packet with 1222 bytes and P5 would just be encapsulated and remain unfragmented at 410 bytes.

a. Yes both packets would be directed to the same socket

b. UDP uses source IP to differentiate a packet between two hosts.

Host C would assign two new sockets dedicated to the particular event for the incoming requests from two different hosts. Both the sockets would have same port number. As TCP is a connection-oriented connection allocated sockets would not change throughout the Lifecyle. Receiver and sender send and receives bytes from the newly allocated sockets.

a. Benefits of using Multipath TCP:

1. Increase in throughput

2. decrease in disruption

3. Handover improves because of the protocol; application don’t know the change.

4. Can use multiple IP addresses at once

5. Can work on either of IPv4 and IPv6

b. No the packet sent over Wi-Fi cannot simply be resent over ethernet. The packet sent over Wi-Fi may need to be broken into smaller pieces depending on the packet’s size and ethernet’s MTU (packet size < ethernet’s MTU) as MTU of ethernet is lower than Wi-Fi.

c. Wi-Fi’s MTU is larger than ethernet’s, because of which a bigger payload could be sent over Wi-Fi network.

If packet’s need to be retransmitted than they would have to be resized according to the ethernet’s MTU, this would take additional processing time by the computers. To avoid this issue sender could use the lowest MTU size available for transmission as a basis of payload size, a packet which could be sent on the lowest MTU could easily be resent on another type of network with larger MTU’s.

Imp: Neither channel will reorder packets.

Checksum, modified ACK and timers to provide a reliable data transfer.

In addition to these we will need to flags: fin (finish) and pfin (part finish).

Divide the data into smaller parts and send the data into these parts. This would be helpful when the network MTU or the receiver window is smaller than the data size.

Flag pfin: use pfin flag similar to fin flag, just we use it at the end of one part. This would ensure proper delivery of parts, if a packet gets lost or corrupted only the part needs to resent.

Modified ACK: returns count of no. of packets received since a. start of transmission b. last time pfin flag was checked.

Timer: waits for an ACK after a parts last packet and the last overall packet.

Checksum: confirms if the data received in packet is correct, otherwise the sender would send a pre-emptive ACK with count as 0, on reception the receiver will retransmit the part.

Idea: We divide the number of packets to send in smaller parts. Now for reliable delivery we only for the part to be delivered correctly.

So, we start transmission of part 1(10 packets) and the sender is keeping count of number of packets sent in the part last packet would have pfin flag checked and fin flag unchecked.

Receiver also keeps a count from packet one till it receives either pfin or fin flag checked,

If all are ok, receiver sends back the count in ACK field. Sender will check the count against its own count and if it matches it can move to the next part.

Otherwise sender would retransmit the whole part again.

When checksum of a packet is wrong receiver doesn’t have to wait till end of part transmission but can send a pre-emptive ACK with count as 0, on reception the receiver will retransmit the part.

If the timer runs out, sender would retransmit the part.

Using this we would be able to provide a correctly working reliable data transfer protocol.

a. Seq no. = 165

source port no. = 303

destination port number = 80

b. Seq no. = 165

source port no. = 80

destination port number = 303

c. ACK no. = 145

a.

|  |  |
| --- | --- |
| Connectionless | Connection-oriented |
| 1. Directly start data transmission | 1. Connection has to be established for transaction to start |
| 2. Doesn’t verify if destination is online before transmission | 2. Verifies if destination is online before transmission |
| 3. Doesn’t require authentication | 3. authentication is possible |
| 4. Doesn’t check if message is received by destination | 4. Checks if message is received by destination |
| 5. Un reliable service | 5. Reliable service |
| 6. Message based service | 6. Stream based service |
| 7. e.g. UDP | 7. e.g. TCP |

b. Connectionless (UDP) places a heavier burden on the network

c. Connection-oriented (TCP) places a heavier processing load on computers communication.

1 Mbits = 125,000 Bytes

When handshake is performed receiver would provide a buffer window of 150,000 Bytes. Sender will start transmitting, during first second sender would send 125,000 Bytes to receiver at the rate of 1Mbits/Sec and receiver would remove at the same rate. At t = 1 sec received and removed data is 125,000 Bytes and the buffer window advertised is 150,000 Bytes.

Between Second 1 and 2, sender would transmit at 2 Mbits/Sec and receiver would remove at 1 Mbits/Sec. 250,000 Bytes would be transmitted during this second and 125,000 would be removed by the receiver and 125,000 would be placed in buffer window. At t = 2 sec receiver would advertise the buffer window at 25,000 Bytes.

Average ERTT = 0.6\* 20 + 0.3\*40 + 0.1\* 100

Average ERTT = 34ms

ERTT = Average ERTT = 34ms

Timeout = 2 \* ERTT

Timeout = 68ms

Packets taking more than 68ms would be assumed lost, therefor only 1/10 packets would be assumed lost.