Homework #2 for protok11

Peter Boström <pbos@kth.se>

2011-09-11

1. ARP

Hosts:

H1: IP=A, MAC=x
H2: IP=B, MAC=y
H3: IP=C, MAC=z
H4: IP=D, MAC=t

Bridge B1 Connects H3, H4 and router R1.

Router R1:

=> B1: IP=E, MAC=v
=> H2: IP=F, MAC=w
=> H1: IP=G, MAC=u

B1 has a MAC address table, R1 and all hosts have ARP tables, all initially empty.

Consider that host H4 sends an IPv4 unicast datagram to host H1.

a, b) Provide the state of the five ARP caches as they will appear after the IPv4 unicast datagram has been delivered to host H1, that is, after dynamic ARP resolution has been made.

H4 checks its routing table and decides to send this to IP E (R1's IP on the interface connected to the bridge). E is not in H4's ARP table, so it broadcasts a request for E's MAC address through ARP. This request includes H4's IP and MAC address. B1 learns H4's MAC and repeats the broadcast on its other interfaces. H3 learns H4's IP and MAC address, but sends nothing. R1 learns H4's IP and MAC as well, and send back its own IP and MAC address directly to H4. B1 learns R1's MAC and interface. H4 learns E's MAC address.

H4 is now ready and forwards the datagram to R1. R1 checks its routing table and decides to send the datagram to H1's IP A. It broadcast to find A's MAC address to be able to deliver it. By doing so it also sends its own IP and MAC, which gets learned by H1 before returning (A, x) to R1.

The datagram then gets delivered to H1.

a) ARP caches

H1 (IP, MAC):
G, u

H3 (IP, MAC):
D, t

H4 (IP, MAC):
E, v

R1 (IP, MAC):
D, t
A, x

H2 is empty.

b) MAC address table

B1 (MAC, Interface):
t, West
v, South

2. UDP and fragmentation

6400 octets to be sent inside an UDP datagram over a link with MTU=2272 octets.

a) How many fragments are transmitted?

An UDP datagram adds an 8-octet header => 6408 bytes. An IPv4 header will take 20 bytes which gives a maximum payload of 2252. Fragmented packets must however be in sizes divisible by 8, so we can use up to 2248 octets for each packet.

We'll need ceil(6408/2248) = 3 fragments to send this UDP datagram.

b) Give the values of the MF bit, offset and total length field of the IP header of each fragment.

Packet 1 will have MF=1 and offset = 0, it'll contain the first 2248 octets of the UDP datagram. Length = 2248. There are now 4160 octets left.

Packet 2 will have MF=1 and offset = 2248/8 = 281 and contain the 2248 following octets. Length = 2248. There are now 1912 octets left.

Packet 3 will have MF=0 and offset = 2248*2/8 = 562 and contain the remaining 1912 octets. Length = 1912. This is the final packet.

3. TCP session management

Consider the TCP SYN Flooding Attack.

a) What kind of attack is this?

A Denial-of-Service attack.

b) How is the attack done?

During a TCP handshake, a SYN is sent to the server, which sends a SYN-ACK back the attacker is expected to send an ACK back, which never happens. This gives the server a half-open connection and occupies limited resources.

An attacker will send a lot of these SYN messages and establish a lot of half-open connections, eventually the SYN queue or other limited resources will be occupied and connections can't be established by legitimate hosts.

c) What damage is caused?

New hosts will be unable to connect to the server, as it's occuped with these half-open connections.

d) How can a server alleviate the effects of the attack? Describe one solution in sufficient detail such that it becomes clear why it alleviates the attack.

Provided that the attacker is unable to spoof the source address, then you can simply limit the number of new connections per source per timeframe. This means that additional SYN messages from the same source are dropped, and the number of half-open connections from that host is limited.

Another technique is known as SYN cookies which presents a cryptographic task that the source needs to solve.

4. TCP 2

Segment to existing connection is sent at 4:30:20. The sender doesn't recieve an acknowledgement and resends at 4:30:28 and recieves an acknowledgement at 4:30:30 where it sends another segment and recieves the following 2 seconds after, at 4:30:32. When the first mentioned segment was sent the smoothed RTT (sRTT) was 4 seconds.

Give the values of the smoothed RTT (sRTT), the variation (RTTvar) and the retransmission timeout (RTO):

RFC2988:

(2.3) When a subsequent RTT measurement R' is made, a host MUST set

RTTvar <- (1 - beta) * RTTvar + beta * |sRTT - R'|
sRTT <- (1 - alpha) * sRTT + alpha * R'

The value of sRTT used in the update to RTTvar is its value before updating sRTT itself using the second assignment. That is, updating RTTVAR and sRTT MUST be computed in the above order.

The above SHOULD be computed using alpha=1/8 and beta=1/4 (as suggested in [JK88]).

After the computation, a host MUST update:

RTO <- sRTT + max (G, K*RTTvar)

(where K=4, from 2.2, G is only larger than RTTvar if RTTvar = 0)

a) after the transmission of the first-mentioned segment (4:30:20)

sRTT = 4s
RTO = 8s (as the segment was resent after 8 seconds)
RTO came from sRTT + 4*RTTvar =>
RTTvar = 1s

b) after the first retransmission (4:30:28)

RTO doubles when it triggers (5.5), no new measures for RTT are made, so sRTT and RTTvar remain the same.

sRTT = 4s
RTO = 16s
RTTvar = 1s

c) after the reception of the first acknowledgement (4:30:30)

As the packet was retransmitted, we ignore these values (Karn's algorithm).

sRTT = 4s
RTO = 16s
RTTvar = 1s

d) after the reception of the second acknowledgement (4:30:32)

RTTvar <- (1 - beta) * RTTvar + beta * |sRTT - R'|
sRTT <- (1 - alpha) * sRTT + alpha * R'
RTO <- sRTT + max (G, K*RTTvar)

RTTvar <- (1 - 1/4) * 1 + 1/4* |4 - 2| = 3/4 + 2/4 = 5/4 = 1.25s
sRTT <- (1 - 1/8) * 4 + 1/8 * 2 = 7/8 * 4 + 2/8 = 30/4 = 3.75s
RTO <- 3.75 + 4*1.25 = 8.75s

5. TCP 3

Stations A and B are connected via a 100Mbps link between the Earth and a communication satellite at an altitude of 36,000 km. Assume that the signal propagation speed equals the speed of light (300000km/s).

a) Calculate the minimum round-trip time (RTT) for the link.

The connection must travel to the satellite and back twice, that is 4*36,000 km =

= 144,000 km, to get back, this gives a RTT of 144,000/300,000 = 0.48 seconds.

b) Calculate the bandwidth-delay product of the link. Explain the meaning of this product.

BPD = 100Mbps * 0.48 seconds = 48Mb = 6MB

This is the highest amount of traffic that reside inside in the network at any time. That is, packets that have been sent but not yet recieved.

c) What's the minimum time to transfer a 25MB file from A to B? Include the connection establishment time. Consider the transfer finished when the last ACK has arrived at the sender. Assume that there are no losses. Use the RTT value computed above.

First, a three-way connection has to be established. This takes RTT time, as the sender will not start transmitting before the first SYN-ACK has been recieved.

After the connection has been established, we can push things into the pipe at 100 Mbit = 12.5MB per second. To send a 25MB file this takes about 2 seconds. After the final byte has been pushed into the pipe, we'll need to wait for the final ACK, which will take RTT time as well.

Total time: 2s + 2*RTT = 2 + 2*0.48 = 2.96 seconds.