Computer Network And Security

Arjun Ahuja and Satish Reddy

Considering configuration {HOST1,FIREWALL,HOST2}

- HOST1:We have made a raw socket and used this to forward all the traffic generated to firewall.The traffic types can be TCP,UDP or ICMP.I have created custom headers for each of the layers and forwarded this packet to the firewall residing in a different vm.We Used struct to created custom headers of the packet which is unpacked in the firewall.
- FIREWALL:Here,We made a raw socket and kept listening for the frames destined to the mac address of the firewall.Then I unpacked all the layer header and determined whether it is a TCP,UDP or ICMP packet.Extracted data from the frame is forwarded to the client on HOST3 according to the firewall rules.
- HOST2:Here there is a client running on this host which can be configured to receive TCP,UDP or ICMP packets based on the application.

Question 2:

- ***Making a Command Line Interface(CLI):*** We created a custom command line interface to add rules, delete rules and update them.
- ***Using a dictionary to store the rules:*** I used a python dictionary to store the rules, for this part I made two major categories:Inbound and Outbound.
- ***Basic Structure of Dictionary:*** DICT { InBound {} Outbound {} } , Though for the purpose of this assignment we did not use Outbound Rules but this can be extended to that easily.
- All Rules were added through CLI to the dictionary.
- **TCP / UDP Rules**: I incorporated <,>,=,a-b(range) rules for the purpose of this assignment, all the ports that are in these rules will be rejected by the firewall all others will be checked for other rules.
- ***MAC-ADDRESS/IPv4/IPv6***: For this I have taken input address(ip/mac) all the addresses in the input will be rejected by the firewall.

Question 3:
We tested this on local host and normally on local host PPS found was around 850 packets/second and normally on vm we found it to be around 100 packets/seconds for a packet ~50 bytes for the unoptimised version done in question 2.

Change of Performance if Drop happens:

Incase of Random/Controlled traffic the dropped packet will match some of the other rules so , not all rules will be checked for this case so overall it will be faster.
And the passing packet will have to be checked against every rule so this will be slower.

In our Implementation if the packets even match one of the rules it will be dropped as we implemented rejection case only, so if it gets rejection from anywhere it will be dropped,Getting 10/50/100 rules will of course make the code run slow, but there will not be any significant change as the checking algorithm is O(n) where n is the number of rules so it does not matter much.

If the number of fields to be matched is increased then the code will have have to check against all the parts (MAC/ ports etc) major blow will be to the packets which pass through as they will have more rules to face.

However we can improve this time by keeping threads for different checking parts.

We Used controlled traffic to benchmark the system.

Here are a few results:

This is the time taken(AVG) by each packet to go from host2->firewall->host1

A. TCP Not optimised AVG:0.000644993782043 sec
B. UDP Not optimised AVG:6.41345977783e-06 sec

`I used flooding (While(1) on packets to send from host2->host1) as traffic generator.`

Question 4:

For this I used 2 optimisations:

1.If we get similar request again we used cache for checking if a similar packet(same host,destination,protocol) had arrived earlier too.for this we created a array in which we are storing the combination of srcMac/destMac/Ipv4/Ipv6/TCP/UDP/ICMP/Port etc and checking it against a similarly checked packed, if that packet was allowed we allow this too if it was not allowed we don’t allow it.

2.Second Optimisation we used is joining the port numbers giving similar conditions together this increased the complexity at the side of rule set but decreased the complexity of the firewall so overall its better to have it.

Example
1.if we have 2 rules for ports < 1000 and 500-600 then the second rule is not needed so we can merge these rules together to get a new one.
2.if we have 2 rules for ports 500-600 and 550-650 we can merge together into a single rule 500-650.

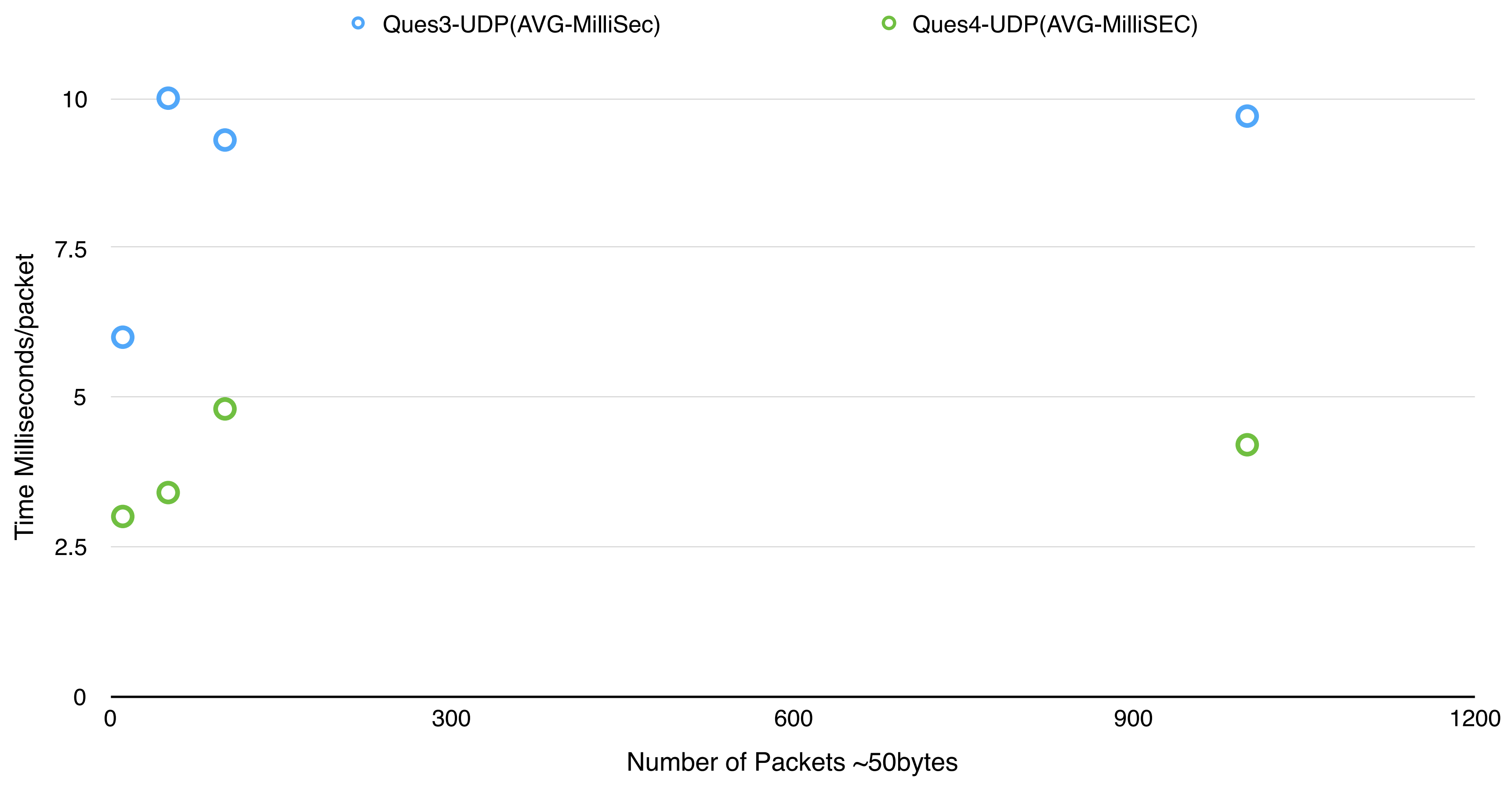
Performance:

This is the time taken(AVG) by each packet to go from host2->firewall->host1

A. TCP Not optimised AVG:0.000160098075867 sec

B. UDP Not optimised AVG:3.32233456783e-06 sec

We see that for TCP ~4 times faster and for UDP it is ~2 times faster than the original one.(for 10 packets for rest see graph)



Ques3-UDP(AVG-MilliSec)
Ques4-UDP(AVG-MilliSEC)
10
7.5
5
2.5
0
0
300
600
900
1200
Time Milliseconds/packet
Number of Packets ~50bytes