ssmping
<draft-venaas-mboned-ssmping-00.txt>

Stig Venaas
venaas@uninett.no
ssmping

- A tool for testing multicast connectivity and more
- Behaviour is a bit like normal icmp ping
- Implemented at application layer using UDP
  - No additional requirements on the operating system
  - The operating system and network must support SSM
- A server must run ssmpingd
- A client pings server by sending unicast ssmping query
- The server replies with both unicast and multicast ssmping replies
- In this way a client can check that it receives SSM from the server
  - You can run your own server, also several public IPv4 and IPv6 servers on the Internet
  - And also parameters like delay, number of router hops etc.
How it works

User runs ssmping <S>

Client joins S,G

Clients sends unicast to S

Server receives unicast ssmping query

Responds with ssmping unicast reply and multicast reply to G

Client receives replies and prints RTT and hops from server

Client sends a new query every second
$ ssmping -c 5 -4 flo.nrc.ca
ssmping joined (S,G) = (132.246.2.20,232.43.211.234)
pinging S from 158.38.63.20
  unicast from 132.246.2.20, seq=1 dist=13 time=122.098 ms
  unicast from 132.246.2.20, seq=2 dist=13 time=122.314 ms
multicast from 132.246.2.20, seq=2 dist=13 time=125.061 ms
  unicast from 132.246.2.20, seq=3 dist=13 time=122.327 ms
multicast from 132.246.2.20, seq=3 dist=13 time=122.345 ms
  unicast from 132.246.2.20, seq=4 dist=13 time=122.334 ms
multicast from 132.246.2.20, seq=4 dist=13 time=122.371 ms
  unicast from 132.246.2.20, seq=5 dist=13 time=122.360 ms
multicast from 132.246.2.20, seq=5 dist=13 time=122.384 ms

--- 132.246.2.20 ssmping statistics ---
5 packets transmitted, time 5003 ms
unicast:
  5 packets received, 0% packet loss
  rtt min/avg/max/std-dev = 122.098/122.286/122.360/0.394 ms
multicast:
  4 packets received, 0% packet loss since first mc packet (seq 2) recv'd
  rtt min/avg/max/std-dev = 122.345/123.040/125.061/1.192 ms
What does the output tell us?

- 13 unicast hops from source, also 13 for multicast
- Multicast RTTs are slightly larger and vary more
  - The difference in unicast and multicast RTT shows one way difference for unicast and multicast replies, since they are replies to the same request packet
- The multicast tree is not ready for first multicast reply, ok for 2nd
- No unicast loss, no multicast loss after tree established
Is it useful?

- Complements multicast beacons
- Useful for “end users” or others that want to perform a “one-shot” test rather than continuously running a beacon
- Beacons don’t show how long it takes to establish the multicast tree, they only show the “steady state”
  - We’ve seen cases where it takes much longer than expected
- Neither do they compare unicast and multicast
- Are there other data than RTT and hops that should be measured?
  - Hops are measured by always using a ttl/hop count of 64 when sending replies
Also asmping. Example output:

sv@xiang /tmp $ asmping 224.3.4.234 ssmping.uninett.no
ssmping joined (S,G) = (158.38.63.22,224.3.4.234)
pinging S from 152.78.64.13
   unicast from 158.38.63.22, seq=1 dist=23 time=57.261 ms
   unicast from 158.38.63.22, seq=2 dist=23 time=56.032 ms
   multicast from 158.38.63.22, seq=2 dist=7  time=207.876 ms
   multicast from 158.38.63.22, seq=2 dist=7  time=208.567 ms (DUP!)
   unicast from 158.38.63.22, seq=3 dist=23 time=56.852 ms
   multicast from 158.38.63.22, seq=3 dist=21 time=70.352 ms
   multicast from 158.38.63.22, seq=4 dist=21 time=57.208 ms
   unicast from 158.38.63.22, seq=4 dist=23 time=57.910 ms
   unicast from 158.38.63.22, seq=5 dist=23 time=56.206 ms
   multicast from 158.38.63.22, seq=5 dist=21 time=57.375 ms
Protocol overview

- All messages have following format
  - Message type, one octet (Q or A)
  - Options in TLV format
  - Client sends Q message with some options
  - Server sends two identical replies, one unicast and one multicast
  - Changes Q into A, echoes back all options, may add some
    - Server should only add options when requested?
  - Responses have ttl/hop count of 64
Client options

- Client identifier
  - IP address, PID, hashed?, some random number?
  - Used by client to know it is not a reply for someone else
- Sequence number
  - 1 for first request, increased by 1 for each request
- Timestamp in microseconds (also for servers)
- Multicast group
  - Only for ASM (or?), see later slide
- Option request option
  - Client might ask server to include certain options
- Reply size
  - Client asks server to send response of a given size
  - Can it be used for DoS attacks? Should client instead pad its queries? May be hard to know response size if server is asked to add options
Server options

- Server may append options (only by request?)
- Timestamp in microseconds (also for clients)
- Version
  - Free text vendor/implementation version etc (UTF-8?)
- Pad
  - If client asks for given reply size
Server behaviour

- What should server do if it is overloaded?
  - It’s been suggested that server can multicast generic/common replies to clients. Is that useful?
  - Should it simply not respond?
  - Should it respond with some “leave me alone” message
    - Might also be useful if server restricts which clients to serve
Even more useful to have tool for ASM (IMO)
  - Registers/MSDP, multiple forwarding paths...

Want to allow client to pick multicast group (or prefix)
  - For IPv6 we should use fixed group id and allow /96 prefix to be specified
  - Useful to choose group to choose different RPs or scopes

Can client pick address that is used by some multicast session in order to attack it?

How to reduce the security issue?
  - Server rate limit?
  - Fixed destination port?
Next steps

- Want to reserve port number and/or SRV name
  - Open question whether client should use a fixed port and whether it can be the same as the server port

- Reserve IPv4 SSM address?
  - The source might be running other multicast applications

- Reserve IPv6 Group IDs
  - Used for both SSM and ASM

- Don’t think reserving anything for IPv4 ASM is doable

- Need input to improve protocol