Oblig 1

Preben Zahl - prebenz - 20.09.21 - IN4230

Purpose

In this system a client should be able to ping a server by using MIP. If the client does not already know the mac-address to the server, the client has to find the server by using an MIP-ARP request. The server should then respond and both server and client can now cache the mac-address and map it to the MIP-address.

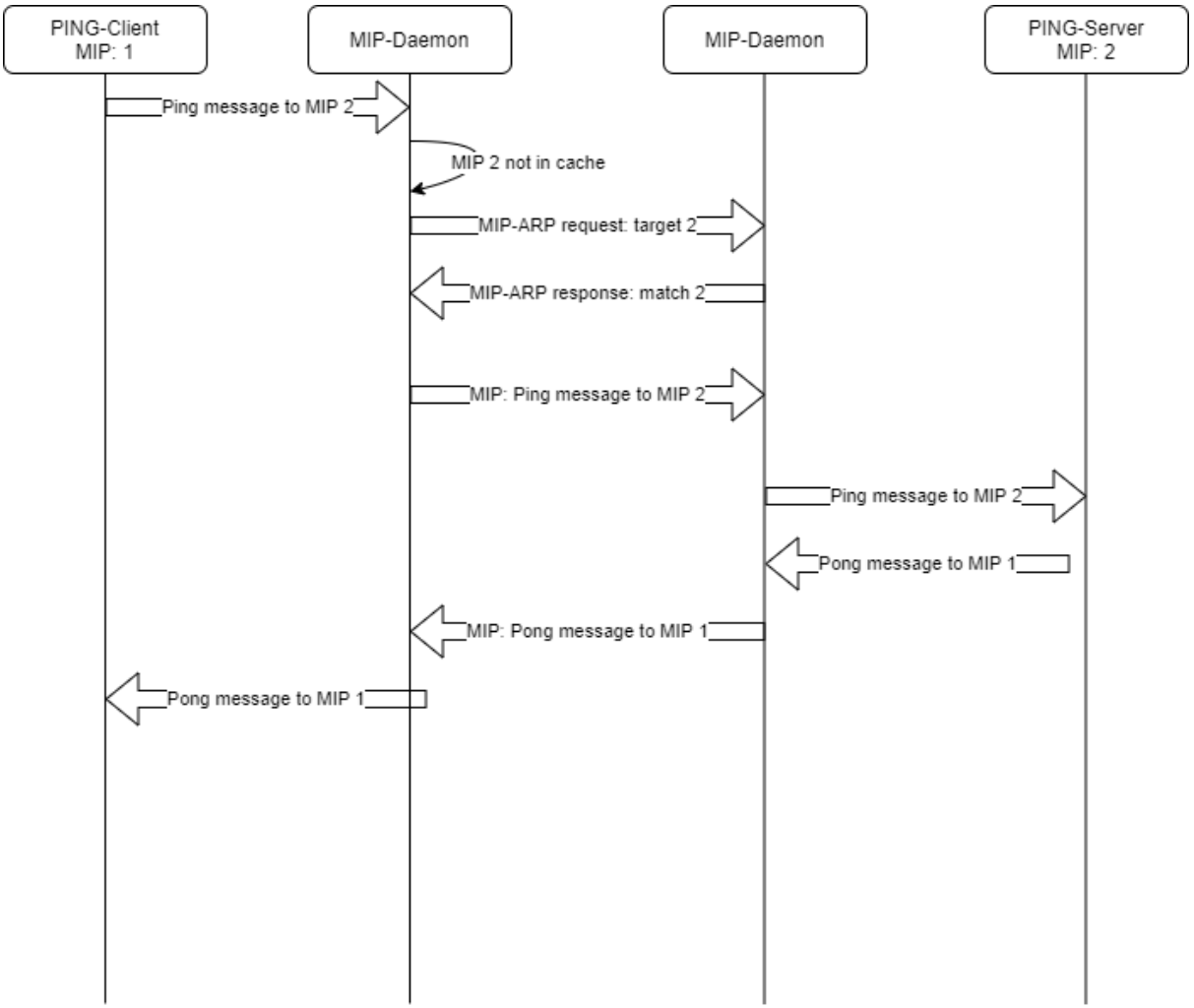
Run

I recommend using the makefile to compile the files. Then there are several make-commands to run the files easily. There are both terminating and non-terminating versions of the daemons (see Termination -t).

Implementation

Both the client and the server are going to use an unix socket to communicate with the MIP-daemon. The daemons on both sides will then use raw-sockets to communicate with each other.

Flowchart



Implementation choices

Unix message

The packets sent through the unix socket should contain both a message and the destination of the packet. Therefore, the 8 first bits of the unix packets is the MIP-dest, then the rest is the message.

Message queue

When the daemon does not have the mip-dest in the cache, a broadcast is performed. Instead of blocking while waiting for the Arp-response, the program puts the message in a queue and sends it when the response has arrived. Right now this queue has only space for one message (the waiting message variable), but this can be made into a linked list to have space for more messages.

Socketfd-list

There is an array for socketfds that is used to poll for events. Right now there is a max capacity for the number of sockets, but that should be fine since we only allow one unix socket to connect. It is possible to make this a linked list if we want more unix sockets to connect.

Termination -t

I found it practical to make the program terminate if it did not poll anything for 10 seconds. But I understand that this can be a bit short if one has to start many programs manually. Therefore I made this optional by adding the -t flag.

MIP-ARP

The MIP-ARP has to be treated carefully because it is a broadcast. That means it is sending a request to all known interfaces. If the time to live is bigger than 1 on those messages, one could create a lot of unnecessary traffic.

The MIP-ARP always sends the message to all interfaces, even if that MIP address is already cached, that is unnecessary traffic. We could just not send to those interfaces. Another change could be to cache the MIP addresses when you receive a MIP-ARP when it's not meant for you. Right now the address is only cached if the message is meant for you.

One could also implement a way to ask for others' cache-tables. That way, a new host does not have to broadcast every time it has to find a host it has not sent to before.

Comparison to IPv4

IPv4 has 32 bit addresses while MIP only has 8. This gives IPv4 a lot more unique addresses and the opportunity to have more hosts.
Another difference is that each node in MIP is given one address, unlike IPv4 where each interface has an address. This means that communication can continue from one interface to another if one goes down. It also means that packets can be arriving from different interfaces all the time, so we would need to think about the consequences here.