Hanna Øster

Design document in4230: MIP deamon

Notes on errors with execution

Unfortunately, I was not able to resolve an issue where the program always resulted in timeouts when running the ping_client. The daemon appeared to build and transmit MIP frames with the correct Ethertype (0x88B5), but no frames of that type were ever received by the raw socket. To work around this, I switched to opening raw sockets with ETH_P_ALL. This confirmed that packets were being transmitted, but still no MIP frames were delivered back to the daemon. I do not know whether this problem was caused by filtering of unknown Ethertypes in the Mininet environment, or by mistakes in my own setup and handling/parsing of raw packets.

MIP vs IPv4

The MIP protocol is a much more simple protocol than IPv4. The focus of the MIP protocol is delivering packets on the Ethernet broadcast domain, where the focus of IPv4 is broader and includes global addressing, fragmentation and routing.

The simplicity of the MIP protocol also makes it not compatible with higher level protocols such as TCP and UDP, given that it is not supporting subnets or forwarding in the way that IPv4 is. The header is also much smaller since the MIP protocol only identifies its host with a single byte. However, for a networking course like in3230 these constraints make MIP more suitable as a learning too. It allows for a more general understanding of protocols and internet communication

When it comes to performance, IPv4 is able to perform on a much larger scale over different networks. This scalability and complexity can cause overhead. For example in comparison to the MIP protocol with no routing, there will be delays regarding forwarding. However the performance of MIP is limited to a single network so the lower overhead is not unutilized with this limitation.

Why MIP- ARP must be handled as a special case

MIP cannot trust the operating systems built-in ARP because it is compatible with IP. MIP has a unique address type with one byte, and the linking between this address and the MAC address is not visible since it is not a standardized protocol. Therefore, the MIP daemon

contains its own MIP ARP to be able to do the necessary lookups and updates for communication between nodes on the outgoing interface.

When it comes to alternative design choices, one option could have been to extend the existing Ethernet ARP protocol to also handle MIP addresses. That might have made the implementation simpler. However since ARP is normally used to map IPv4 addresses to MAC addresses, adding MIP entries could confuse the system or overwrite IPv4 mappings. This might cause conflicts with normal IP traffic and make debugging harder.

Another design choice could be to parse the different PDUs in the same way, and then pass them on to separate handler modules depending on the SDU type. For example, ARP messages could go to a dedicated ARP handler, while ping messages go to a ping handler. This would keep the core parsing simple and move protocol-specific logic into smaller, cleaner parts of the code.

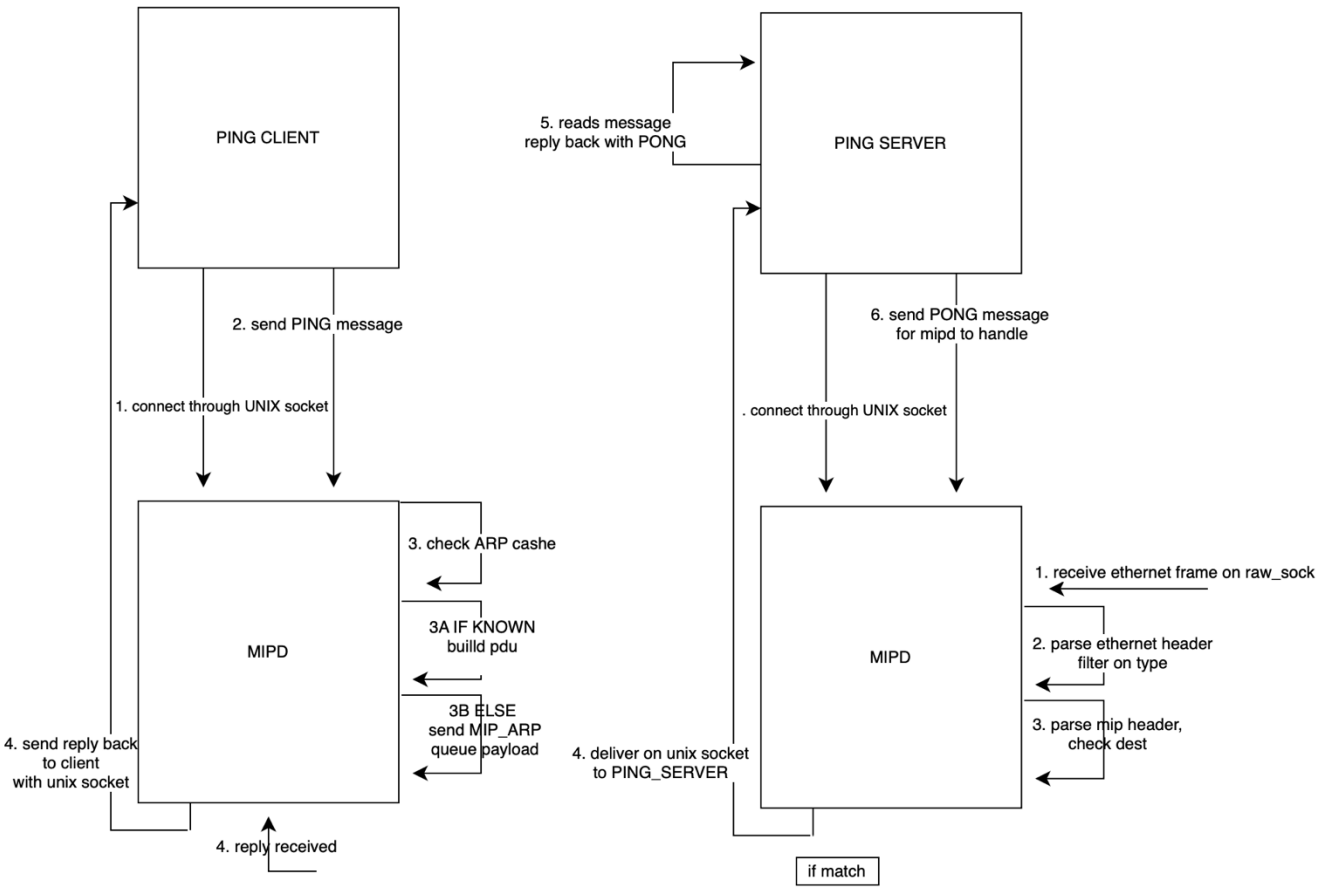
Flow of execution

Client (ping_client -- mipd)

1. ping_client connects to the daemon via UNIX socket.
2. Client sends message including destination MIP address and payload.
3. mipd receives the request, checks ARP cache
    - If MAC is known: build MIP PDU, encapsulate in Ethernet frame, and transmit
    - If MAC is unknown: send MIP-ARP request and queue the payload
4. If reply is received: deliver back to ping_client via UNIX socket

Server (mipd -- ping_server)

1. mipd receives Ethernet frames on the raw socket
2. Parse Ethernet header and filter on MIP Ethertype
3. Parse MIP header, check destination
4. If destined for local host: deliver payload to the correct UNIX socket (ping_server)
5. ping_server reads the message and sends back a reply (PONG)
6. mipd transmits reply using same ARP resolution and frame construction logic



Notes on use of AI

I used AI as a support tool to help me understand C programming as i did not have previous experiance with the language. I also used it to debug errors, and improve the structure of my code. In some cases, I also received suggestions for code snippets, which I adapted and integrated myself after verifying them against the assignment requirements.