INF3190 Mandatory Assignment:

Formally:

This assignment must be completed individually. The submission must be approved prior to submission of the Home Exam 1. To pass the submission must meet the requirements that are documented here, and you must be able to explain your solution.

Task

The main focus of the task is to build and test MIP over Ethernet. MIP stands for "Minimal Interconnection Protocol": it is a toy network layer protocol, created for INF3190 that has only the bare necessities in the header and avoids overhead, making it faster to process and minimizing latency. The program you write will run on mininet inside a virtual machine. You will have to write three programs:

1. A **MIP daemon**, which is permanently active on every host. This daemon can:
    - Receive a destination MIP address and data to send via IPC with a Unix domain socket from an application on the same physical host (with *sendmsg(..)* and *recvmsg(..)*). Then, it transmits the message to the provided MIP destination address over Ethernet via a raw socket.
    - Receive MIP packets from Ethernet via a raw socket, upon which an attached application (on the same physical host) receives the packet content via its blocking "receive" Unix domain socket call (if no application is waiting for data, received packets are ignored).

    Each interface (with unique MAC address) needs to be configured with a unique MIP address. The MIP address(es) of a host is(are) configured by providing it as a command-line argument when running the MIP daemon. If the MIP daemon is started in "debug-mode" by the user (another command-line argument), every communication should be logged on the console with status information (source / destination Ethernet addresses and MIP addresses, current status of ARP cache (more details below)).

2. A **ping server**, which receives a text message via IPC from its local MIP daemon. It prints the text and sends back (via IPC to its local MIP daemon) a ping message with content "PONG".

3. A **ping client**, which sends a ping message (via IPC to its local MIP daemon) with user-specified content to a user-specified MIP address. Upon receiving a response (via IPC from its local MIP daemon), it prints the time between sending the message and obtaining the response.

The ping client takes a remote MIP address and a ping "message" as a command line argument. It sends the ping, and after receiving a response and printing the result it exits. If no response arrives within 1 second, the ping client should print "timeout" and exit.

MIP-ARP: To be able to communicate across Ethernet, MIP daemons must implement our own version of "Address Resolution Protocol": MIP-ARP. To communicate to a host with a specific MIP address, its corresponding MAC address is required. For this, the sender sends a broadcast Ethernet frame into the network, and any host with the MIP address specified in this frame will reply back to the sender.

For every packet, the client needs to know the MAC address, and to avoid mapping from MIP addresses to MAC addresses all the time, it is better to locally cache the MIP-MAC-address mapping. Hence, a MIP-ARP-cache table needs to be implemented and the content of it should be printed by the MIP daemon in debug-mode.

Specific requirements for implementation

The basic elements required for the header of the network layer are: the TRA bits, Time To Live (TTL), Source Address, and Destination Address. This header, and everything else in MIP, is in big endian format.

TRA bits: if one of these bits is set to 1, it indicates that this is a **T**ransport (bit combination 100), **R**outing (bit combination 010) or **A**RP broadcast message (bit combination 001). As an exception to this rule, an ARP response is defined by the bit combination 000. Other bit combinations are reserved for possible future use and currently not allowed.
TTL: in the future, this field will be used to prevent loops in routing. A MIP daemon should set it to its maximum value (15) by default.
Src MIP address / Dst MIP address: user-configured host addresses.
Length: the length of the payload in 32-bit words (i.e. the total length of the MIP packet including the MIP header in bytes is Length*4+4).

Important general MIP protocol rule:
The payload of a MIP packet is always a multiple of 4, and the total packet size including the 4-byte MIP header must not exceed 1500.[1] MIP does not fragment

[1] This is the size of the Maximum Transmission Unit (MTU) of Ethernet. MIP is only meant to run over Ethernet, and every MIP message must fit into one Ethernet frame.

packets, i.e. it cannot accept messages that do not fulfill these constraints. If a message that does not fulfill these constraints is given to the MIP daemon for sending, this should cause an error and the message should not be sent.

To detect the frames in the data-link layer that contain these specific network layer packets, we need a protocol type in the Ethernet frame, and hence a macro ETH_P_MIP must be defined for this. ETH_P_MIP is set to 0x88B5.

In MIP_ARP, the broadcast frame (request) looks like a normal MIP packet at the link layer (i.e. protocol type ETH_P_MIP[2]) with TRA bits set to 001 and the MIP length field set to 0; the Ethernet destination MAC address is set to all ones in binary. A MIP-ARP response is also a normal MIP packet, with TRA bits set to 000 and the MIP length field set to 0.

Command line arguments for MIP daemon:

MIP_daemon [-h][-d] <Socket_application> [MIP addresses …]

-d:	debugging;
<Socket_application>:	name of the socket for communicating the application program (ping client/server).
-h:	prints help and exits the program

Command line arguments for ping client:

Ping_client [-h] <destination_host> <message> <Socket_application>

<destination_host>:	MIP address of the destination host;
<message>:	the message that needs to be sent;
<Socket_application>:	name of the socket that MIP_daemon uses at the ping client.
-h:	prints help and exits the program

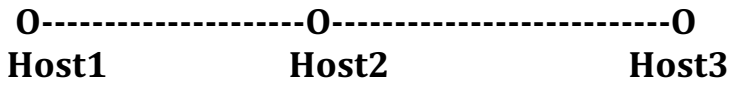
Command line arguments for ping server:

Ping_server [-h] <Socket_application>

<Socket_application>:	name of the socket that MIP_daemon uses at the ping server.
-h:	prints help and exits the program

[2] This is different from the Internet, where ”normal” ARP uses a different Ethernet type and has an entirely different header than IP.

Example Scenario:



Initialization: all empty.

1. Host1 pings Host2:
    1. Host1 broadcasts ``who is Host2?" (a broadcast Ethernet frame containing a MIP packet with TRA bits 001, without payload and Length=0).
    2. Host2 receives the broadcast message, replies back to Host1 with a MIP-ARP response, and both update their caches.
    3. Host1 sends a ping packet to Host2, and Host2 responds.

2. Host3 pings Host2:
    1. Same procedure as above.

3. Host1 pings Host2 again:
    1. Host1 should use the MIP-ARP-cache table instead of broadcasting a MIP-ARP message. It should directly send the ping to the already known Host2, where Host1 is also already known and a "PONG" response can directly be sent.

4. Host1 cannot ping Host3.

Submission

You must submit the following:

1. A *design document* that contains:
    - A discussion of how MIP is different than IPv4, and how its performance compares to IPv4.
    - Present a graph of the function calls both at the client and server side.
    - Detailed documentation on how the program should be executed.
    - Listing of files in the program (C files, header files, etc).

2. *Program code*, where the code is well commented. Document all the variables and definitions. For each function in the program, the following documentation must be contained:
    - What the function does.
    - What input and output parameters mean and how they are used.
    - Which global variables affect the function.
    - What the function returns.
    - Other characteristics that are important to know about (eg. error situations).

Requirements for submission

A **design document** will be printed using an appropriate tool, eg. LaTeX, Word, etc. The document should contain the things listed above, as well as a front page where the following information is provided:
Name - username - date - course

Before delivery, the document shall be converted to **pdf** format. Neither a Word / Works / Open Office / TeX - document nor a regular editor file (plain text) is accepted.

The **code** must consist of compilable text files.

The scope of the document need not necessarily be large, but must contain sufficient information to meet the requirements described under 'task'. The important thing is to document understanding of the relevant topics, in addition to the actual implementation.

Electronic submission: Everything must be submitted electronically where all files (Makefile, * .c, * .h, readme.pdf, etc.) are collected in one directory using your UiO username as a filename. Create one compressed tar-ball containing that directory. Use the command:
tar -zcvf username.tgz username.
A link for online file submission will be provided at the INF3190 website.

Deadline: Friday 02 March 2018 at 23:59:59

Remember that you cannot deliver a copy of answers from others but must deliver your original solution. The requirements for deliveries can be found at:
http://www.mn.uio.no/ifi/studier/admin/obliger/

Explanation Requirements

Teachers will conduct spot checks among the submissions that meet the requirements for approval. Students can be called in for informal meetings. If the explanation is not sufficient, the submission will not be considered as accepted. If the submission is not accepted based on the spot check, it is possible to ask for a formal oral test with a teacher ("faglærer") which covers the same material for approval of the mandatory assignment.