Design Document

Home Exam 1
Course: IN4320
Semester: Fall - 2019
Candidate nr: 15202

A description of your DVR with split horizon implementation.

The router daemon keeps track of the available nodes in the network it knows about using a list data structure containing the following data structure:

```
typedef struct MIPRouteEntry {
    MIP_ADDRESS destination;
    MIP_ADDRESS next_hop;
    unsigned int cost;
    long last_updated_milli;
} MIPRouteEntry;
```

When receiving a new route table on the route table socket. The main logic happens in the MIPRouteTable_update_routing.

routerd/src/lib/mip_route_table.c:105

```
void MIPRouteTable_update_routing(MIPRouteTable *table, MIPRouteTable *neighbor_table)
```

In the function each entry of the received table is matched up against the daemons own table. The routing implementation utilizes poison reverse to handle the cout to infinity problem. If a received table entry has the cost of UNIT_MAX, the entry is discarded.

```
// New never before seen node, add to table. If the node has cost of unit max it is a
poisoned entry
if((champion == NULL || (challenger->cost + 1) <= champion->cost) && challenger->cost
!= UINT_MAX){
    MIPRouteTable_update(table, challenger->destination,
neighbor_table->table_address, (challenger->cost + 1));
}
```

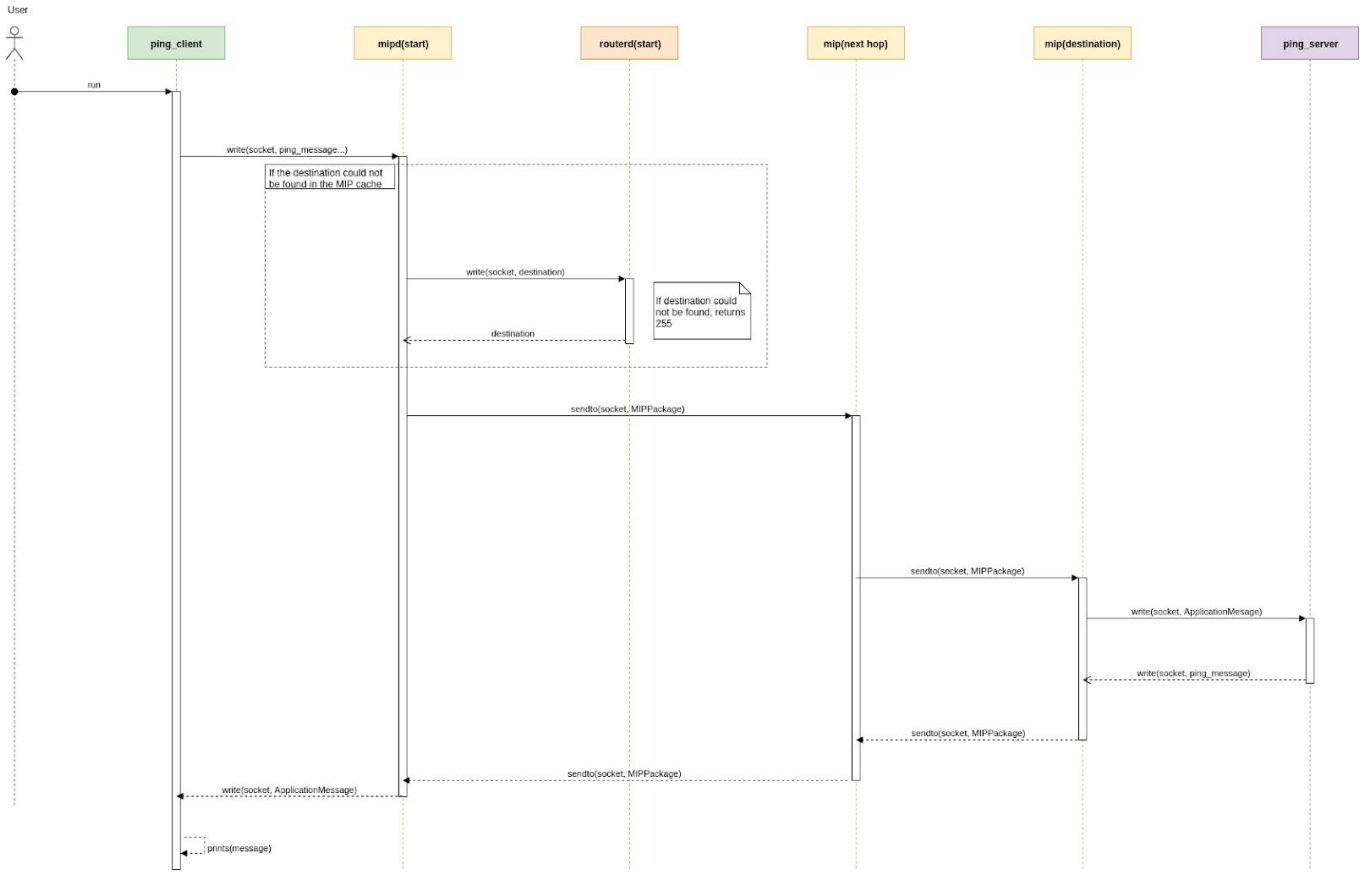
The poisoning of the route tables happens in the MIP daemon. Outgoing route tables are poisoned before they are sent to the receiving MIP daemon.

mipd/src/lib/router/routing.c:45

```
void poison_reverse(MIPRouteTablePackage *package, MIP_ADDRESS destination){
    int i = 0;
    for(i = 0; i < package->num_entries; i++){
        if(package->entries[i].next_hop == destination){
            package->entries[i].cost = UINT_MAX;
        }
    }
}
```

Sequence Diagram

Sequence diagram of the first hop of a ping message. Shows the sequence from a user runs a ping_client against a local MIP daemon to how the message travels through, in this case three separate MIP daemons before it is delivered to the ping server. The ping server responds and the response travels back through the MIP daemons to the ping client.



Documentation on running and using your programs.

Project Layout

IN4230-HE

├── bin/	**<- Directory containing the compiled executables**
├── build/	
├── commons/	**<- source code for common libraries and definitions**
├── mipd/	**<- MIP daemon source files**
├── ping-client/	**<- Ping client source files**
├── ping-server/	**<- Ping server source files**

```
├── routerd/          <- Router daemon source files
├── LICENSE
├── Makefile          <- Full project Makefile, builds and tests all project artifacts
└── README.md
```

Project structure for mipd and routerd

The daemons have a project structure with src and tests directories at the top and with lib and main directories in the src directory. The source code is split in library code and main code to facilitate unittesting of the daemon components through static linking.

```
mipd/routerd/
├── build
├── src
│   ├── lib           <- Code tested by unit tests
│   └── main          <- Entrypoint code for running the daemon
└── tests             <- Unittests for the daemon
```

Building the program

Running a terminal in the project root

BUILD WITH DEBUG
make dev

BUILD ALL PROJECT EXECUTABLES
make

BUILD AND RUN TESTS
make tests

Executing the program

The executable files are located in the bin/ directory in the project root.
NOTE: all the daemons and applications require sudo to run

START THE MIP DAEMON
mipd [-h] [-d] <socket_application> <route_socket> <forward_socket> [MIP addresses ...]

START THE ROUTER DAEMON
The mipd process routerd is to connect to must have been started before attempting to start
routerd [-h] [-d] <route_socket> <forward_socket> [MIP addresses ...]

START THE PING CLIENT
ping_client [-h] <destination_host> <message> <socket_application>

START THE PING SERVER
ping_server [-h] <socket_application>

Dependencies

The project uses some utility libraries for handy data structures such as lists and queues:

commons/src/hashmap.h/c
commons/src/list.h/c
commons/src/darray.h/c
commons/src/queue.h/c

https://github.com/zedshaw/liblcthw/tree/master/src/lcthw

Bstring library

commons/src/bstrlib.h/c
where gathered from: http://bstring.sourceforge.net/