MCL
Mobile Communications &
Quantum Technologies Lab.

Guide for measurements of Mobile Communications and Quantum Technologies Laboratory

GeoNetworking protocol

Place of measurement:

Department of Networked Systems and Services,
Mobile Communications and Quantum Technologies Laboratory (MCL)
I.B.113.

Made by:
Ádám Knapp

Last modification:
24. October 2017

Intelligent Transport System:

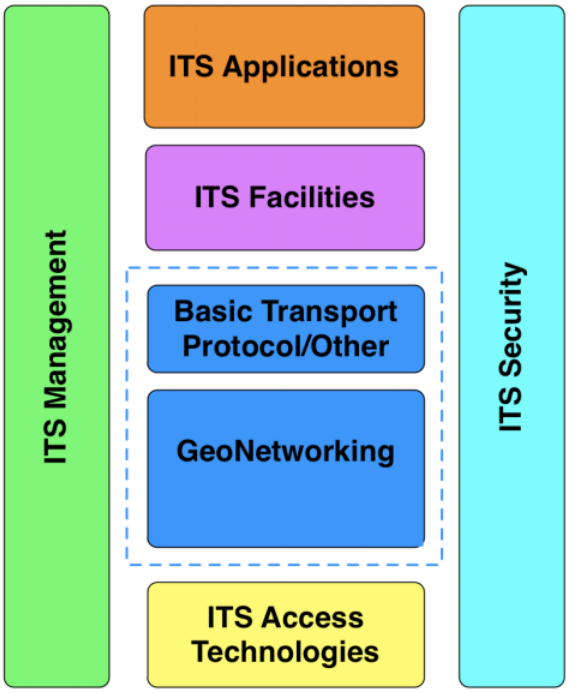

Figure 1. ITS protocol stack

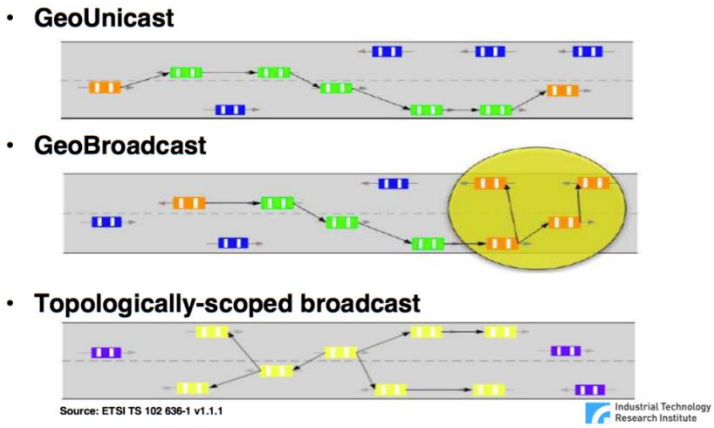

Figure 2. Operation of different message types

Measurement setup

Computer with Windows 7 OS:

- User: MCL
- Password: mcl
- IP address: 192.168.37.xxx

VirtualBox application and Linux image with cross-compiler for ITRI:

- User: user
- Password: password
- IP address: 192.168.37.xxx

ITRI devices:

- User: admin
- Password: iwcu2009
- IP address: 192.168.37.3x

Task 1.

Establish connection between devices and OS-es!

Help:

Check connection:

```
$ping 192.168.37.xxx
```

Set static IP address on Linux:

```
$ifconfig eth0 down
$ifconfig eth0 192.168.37.xx netmask 255.255.255.0 up
```

Set date on Linux:

```
$sudo date –s "2017-10-24 13:55:00"
```

Task 2.

Log in to the ITRI device using PuTTY client!

Help:

Steps for setup an SSH session:

1. Set connection type to SSH! (Default port for SSH is 22.)
2. Fill the Host Name field with the destination IP address!
3. Click to open!
4. Use the username and password to access the device!

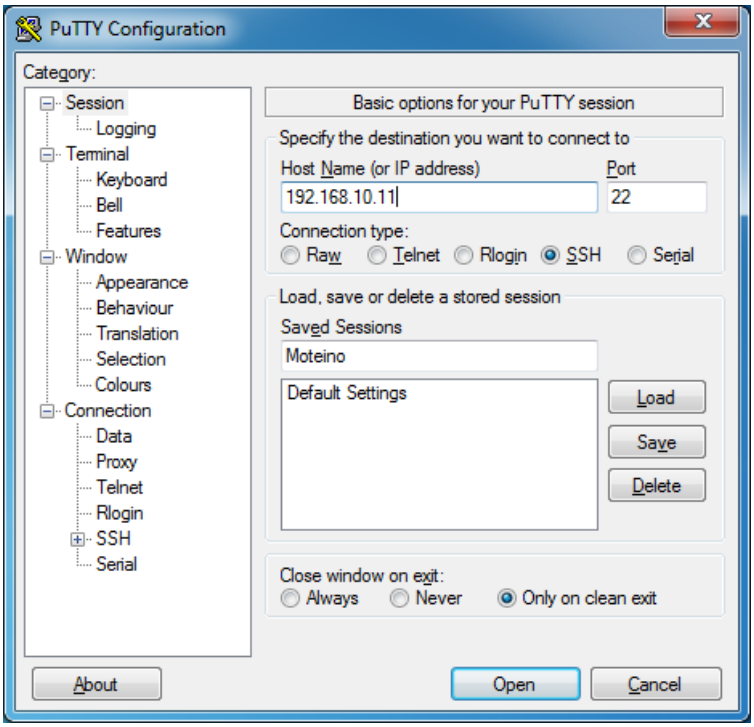

Figure 3. PuTTY client

Task 3.

Configure the ITRI device!

Set bandwidth and carrier frequency:

Check current value:

```
$cat /proc/sys/wave/channel
$wave0 10@5920
```

Set new value:

```
$echo wave0 10@5890 > /proc/sys/wave/channel
```

Possible values:

Setting Channel Number	Command
172	echo wave0 10@5860 >/proc/sys/wave/channel
174	echo wave0 10@5870 >/proc/sys/wave/channel
176	echo wave0 10@5880 >/proc/sys/wave/channel
178	echo wave0 10@5890 >/proc/sys/wave/channel
180	echo wave0 10@5900 >/proc/sys/wave/channel
182	echo wave0 10@5910 >/proc/sys/wave/channel
184	echo wave0 10@5920 >/proc/sys/wave/channel

Set transmission power

Check current value:

```
$cat /proc/sys/wave/txpower
```

Set new value:

```
$echo "wave0 18" >/proc/sys/wave/txpower
```

Set modulation and coding scheme

Check current value:

```
$cat /proc/sys/wave/txrate
```

Set new value:

```
$echo "wave0 4" >/proc/sys/wave/txrate
```

Possible modulation and coding schemes:

Value	Modulation	Data Rate at 10MHz Bandwidth
0	BPSK 1/2	3Mbps
1	BPSK 3/4	4.5Mbps
2	QPSK 1/2	6Mbps
3	QPSK 3/4	9Mbps
4	16QAM 1/2	12Mbps
5	16QAM 3/4	18Mbps
6	64QAM 2/3	24Mbps
7	64QAM 3/4	27.5Mbps

Get statistics

```
$cat /proc/sys/wave/stats
```

Device information (MAC)

```
$cat /proc/net/gn/lpv
```

Neighbor device information (MAC)

```
$cat /proc/net/gn/loc
```

Set location information

Check current value:

```
$cat /proc/sys/net/gn/local_longitude
```

Set new value:

```
$echo 511234567 > /proc/sys/net/gn/local_latitude
$echo 51234567 > /proc/sys/net/gn/local_longitude
```

Note: The above format follows the WGS-84 format meaning that the latitude is set to 51.1234567 and the longitude is set to 5.1234567.

Task 4.

Develop a GeoNetworking sender and/or receiver application!

Using cross-compiler of the Linux VM

1. Copy the source codes in `/user/home/example`!
2. Go to the following directory:
   ```
   $cd /home/user/powerpc-e300c3-linux-gnu/bin
   ```
3. Compile the code with the following command:
   ```
   $./powerpc-e300c3-linux-gnu-gcc /home/user/example/proba.c
         /home/user/example/gn.h -o /home/user/example/proba
   ```

After successful compilation, the executable binary will be available under */home/user/example.*

Copy the binary file to the ITRI using *scp*:

```
scp <files to copy> username@<ip address >:<destination directory>
```

Example:

```
scp proba user@192.168.37.30:/home/user/
```

Run the application on the ITRI device!

Help:

Socket initialization:

```
int socket(int domain , int type, int protocol)
```

Example:

```
sd = socket(PF_BTP, SOCK_RAW, 0);
```

Binding:

```
int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen)
```

Example:

```
err = bind(sd, (struct sockaddr*) &sbs, sizeof(sbs));
```

GeoNetworking header fields in structure:

```
struct sbs {
      unsigned short    btp_family;
      unsigned char     btp_type;
      unsigned short    sport;
      unsigned short    dport;
      unsigned short    dport_info;
      unsigned char     packet_type;
      union gn_dest     dest;
      struct long_pv    src_pv;
      unsigned char     commun_profile;
      unsigned char     sec_profile;
      struct life_time  packet_lifetime;
      unsigned short    max_repeat_time;
      unsigned short    repeat_interval;
      unsigned char     hop_limit;
      struct traffic    tc;
};

union gn_dest {
      struct gn_addr    addr;
      struct gn_area    area;
};

struct gn_addr {
      unsigned short    manual:1,
                        type:5,
                        country_code:10;
      unsigned char     mid[MAC_SIZE];
};

struct gn_area{
      unsigned char     area_type;
      unsigned int      pos_latitude;
      unsigned int      pos_longitude;
      unsigned short    distance_a;
      unsigned short    distance_b;
      unsigned short    angle;
};

struct long_pv {
      struct gn_addr    addr;
      __u32             timestamp;
      __s32             latitude;
      __s32             longitude;
       __u8             pai : 1;
```

```
        __s16               speed : 15;
        __u16               heading;
};
```

Example for accessing an element:

```
printf("heading= %d\n", sbs.src_pv.heading);
```

Data transmission:

```
int sendto(int sockfd, const void *buf, size_t len, int flags,
const struct sockaddr *dest_addr, socklen_t addrlen);
```

Example:

```
sendto(sd, msg, 2048, 0, (struct sockaddr *) &sbs, sizeof(sbs)));
```

Data reception:

```
int recvfrom(int sockfd, void *buf, size_t len, int flags,
struct sockaddr *src_addr, socklen_t *addrlen);
```

Example:

```
int recvfrom(sd, msg, 2048, 0, (struct sockaddr *) &sbs, &len);
```

Closing socket:

```
close(sd);
```