GeoNetworking/BTP and OpenVPN Client on ALIX APU1D

Authors: Jan de Jongh (ed), Jacco van de Sluis, Dennis Heuven, TNO, The Netherlands
Alex Voronov, VIKTORIA, Sweden
Version/Date: v0.1, 20160213
Releasability: PUBLIC

Introduction

This document describes how to enable (ETSI) GeoNetworking and BTP (Basic Transport Protocol) and OpenVPN (client-side) on a PC Engines APU1D.

It is assumed throughout this document that (1) Voyage Linux (0.9.5 or later) has already been installed on the APU1D, (2) that IEEE 802.11p support with a suitable WLAN NIC has already been enabled, and (3) that an 802.11p interface is available as "wlan0". More details can be found in [1].

Except for the OpenVPN Client component, all software components described in this document have been developed in the context of the i-GAME[1] project, and therefore, the intended primary audience for this document are the GCDC-2016 [4] participants. However, we expect this document to be useful to anyone interested in the study of ETSI-based v2v and v2i communications. **All i-GAME developed software components have an open-source license.**

As we move higher-up into the ITS-S protocol stack, the number of different configuration options increases rapidly as several software components have network interfaces allowing them to be deployed on different machines than the components they depend on. In this document, we only provide an *example* deployment on the ALIX APU1D but we will not (except for two fundamental deployment choices) cover or even mention deployment alternatives.

Any feedback on this manual, ideas for improvements or extensions to this manual are highly appreciated.

1. Deployment Options

After installing and configuring a Voyage Linux distribution with IEEE-802.11p support as described in [1], we have taken (only) the first, yet very important, step at (ETSI) v2v/v2i communications. In ISO OSI/RM terminology [ref], we have taken care of Layers 1 and 2 of the Reference Model (Physical and Datalink Layers). This document attempts to tackle the next-two layer, viz., Layers 3 (Networking) and 4 (Transport), of that model by providing guidelines to the installation of relevant software components on the ALIX APU1D. Following the ETSI protocol stack [ref], we are faced with the implementation and deployment of GeoNetworking (GN) and Basic Transport Protocol (BTP).

As it turns out, we have quite some flexibility for these layers, resulting in two important design questions:

(1) Run GN+BTP on the ALIX APU1D or on another machine (e.g., for performance/maintenance reasons);

(2) Send GN+BTP frames over the "wlan0" interface or over a different medium than IEEE-802.11p (e.g., because of range limitations of IEEE 802.11p). In particular, one may want to send the frames over an OpenVPN 'tap' tunnel (with interface name "tap0" in the sequel) in order to

[1] http://www.gcdc.net/i-game The i-GAME project is organizing the GCDC 2016 and has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 612035.

verify encoding/decoding (like CAM [ref] messages) and causal relations between messages (e.g., vehicle A sends message A1 in response to a request message B1 from vehicle B, see, e.g., [ref]).

In this section, we briefly describe each of the four deployment options resulting from the combination of the two design alternatives, and we conclude with a matrix showing which software component is required for each of these options. The components themselves will be described in more details in subsequent sections.

➢ **Option A:** GN+BTP on ALIX APU1D over "wlan0"

In option A, GN+BTP both run on the ALIX APU1D, connecting to the IEEE 802.11p interface "wlan0". This is the typical configuration for use as Communication Unit in vehicles and roadside equipment. Its main advantage is that the entire protocol stack up to and including the Transport Layer (BTP) run on a single small device. The configuration is shown in the figure below.

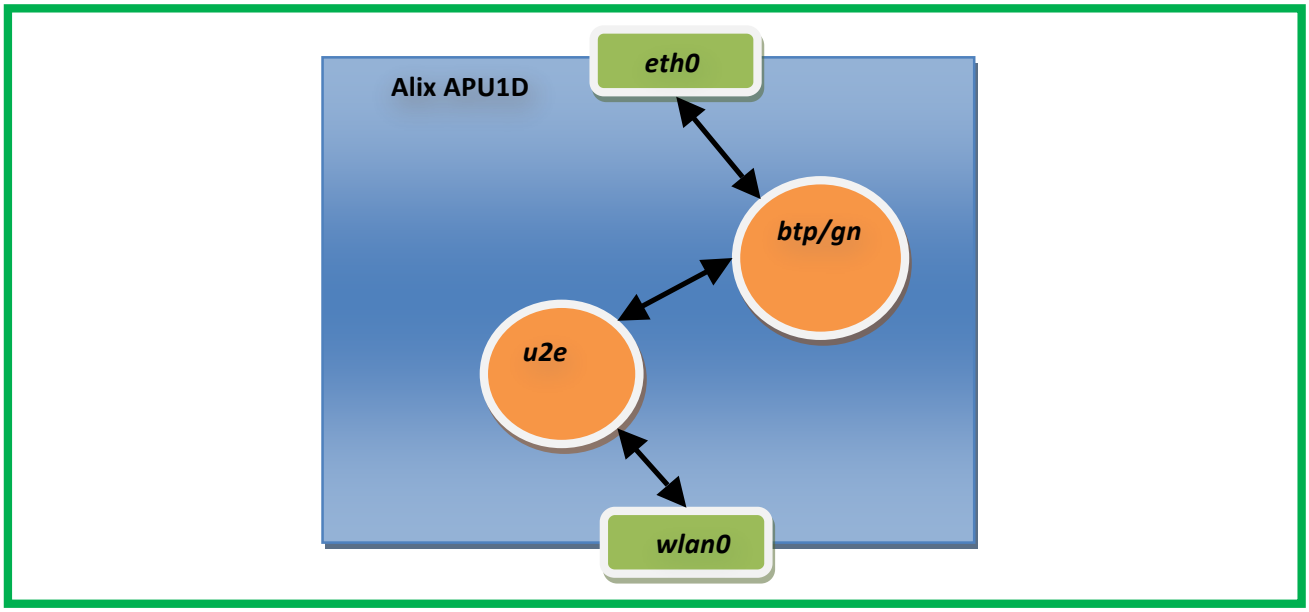

Figure 1: Configuration A.

The GN+BTP implementation was written in Java and is available as open-source from the GitHub page of Alex Voronov of VIKTORIA Institute [ref].

Note that even though we use the IEEE 802.11p interface, we still need a UDP to Ethernet converter, either *udp2eth* or *u2epy*. This is because of the implementation of the interface between GN and the Linux network interface (i.c., "wlan0").

➢ **Option B:** udp2eth/u2epy on ALIX APU1D over "wlan0"

In option B, we do *not* run GN+BTP on the ALIX APU1D, but instead, redirect the traffic on the IEEE 802.11p interface, "wlan0" to another machine. Although we assume that that machine runs the GN+BTP implementation, this is by no means the only possible option: any Layer-3 protocol that runs over Ethernet-like interfaces on Linux can be deployed.

This is the typical configuration, for instance, in roadside equipment that needs geographically spread IEEE 802.11p communication equipment, but uses a centralized implementation of GeoNetworking and BTP. Motivations for doing so include ease of software management (of the GN+BTP stack), performance concerns with running the Java-based GN+BTP stack on the ALIX APU1Ds, and the advantages of doing centralized (GN) routing for a deployed (and static) roadside infrastructure (including, for instance, the option to forward GN traffic between roadside-CU's over *other* communication means than IEEE 802.11p).

The configuration is shown in the figure below.

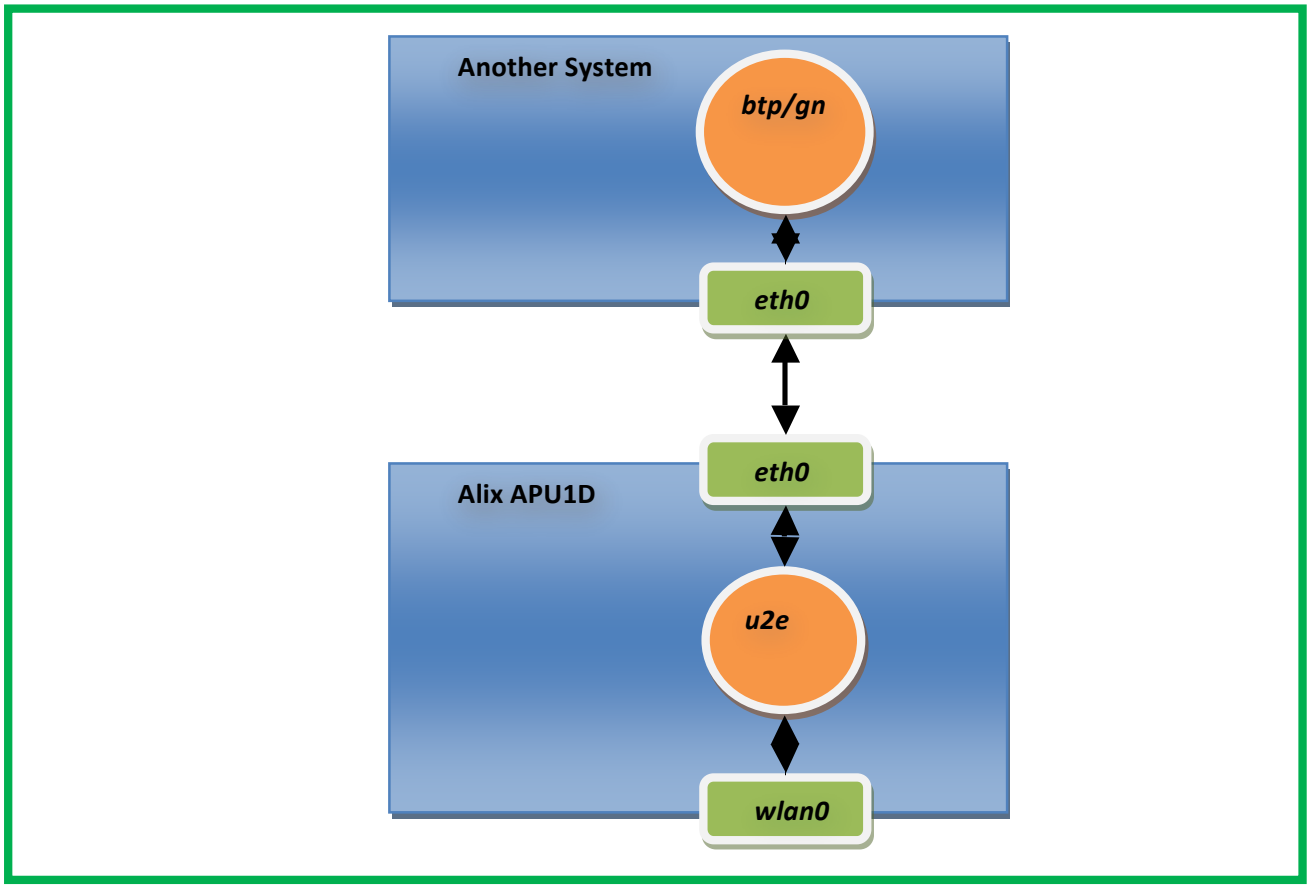

Figure 2: Configuration B.

➢ **Option C:** GN+BTP on ALIX APU1D over "tap0"

In Option C, GN+BTP run on the ALIX APU1D, but does not use the IEEE-802.11p interface "wlan0", but "tap0" instead, where "tap0" is an OpenVPN L2-tunnel endpoint (in other words, a "tap" device, *not* a so-called "tun" device). As explained with Option B, this implies that a UDP to Ethernet converter must be deployed as well on the ALIX APU1D. However, this option does *not* require the presence of an IEEE-802.11p interface on the ALIX APU1D. In other words, on can also run this on a 'stock' Voyage distribution.

This deployment option is typically used for testing *all* the software running and the ALIX APU1D (so including GN+BTP) *except that* the physical communications interface is replaced with a tunnel device. This allows, with some obvious restrictions, for testing implementations of ITS application without the need for physical proximity of the (hardware) components. For instance, a traffic-light application could run in, say, Vienna, testing the application with two vehicles, one in say Helmond and one in for instance Göteborg.

<picture>

➢ **Option D:** udp2eth/u2epy on ALIX APU1D over "tap0"

In option D, an OpenVPN tunnel endpoint is created on the ALIX APU1D, and traffic to and from this endpoint is redirected using a UDP to Ethernet converter.

The only use case we can think of is testing an option-B deployment, using "tap0" instead of "wlan0". For instance, one might want to stress test the UDP to Ethernet converter.

<picture>

➢ Configuration/Component Matrix

In the table below, we summarize the required software components on the ALIX APU1D for each of the four deployment options described earlier.

Configuration	IEEE 802.11p	GN+BTP	udp2eth/u2epy	OpenVPN Client

A [GP+BTP -> wlan0]	yes [1]	yes	yes	no
B [u2e -> wlan0]	yes [1]	no	yes	no
C [GN+BTP -> tap0]	No	yes	yes	yes
D [u2e -> tap0]	No	no	yes	yes

2. Installing and Configuring *GN+BTP* on ALIX APU1D

In this section we describe how to obtain the open-source GN+BTP implementation written by Alex Voronov from VIKTORIA (and contributors) and made available on his GitHub page [5]. The repository is named *geonetworking*, but in fact, the software covers a lot more ground than just the BTP and GN parts. It is in fact a basis for developing a full ITS Stations, including CAM and DENM generation. In this section, though, we *only* cover the BTP and GN parts.

TODO: Describe building elsewhere; **`mvn package`**, etc.

➢ Downloading and Installing Java and Maven

Before installing the BTP+GN software from Alex's GitHub repository, we need the proper built tools. The *geonetworking* software requires *maven* and the *Java 1.7 JDK* (and for the tests, Java 1.7 JRE as well):

```
# apt-get install maven
# apt-get install openjdk-7-sdk openjdk-7-jre
# update-alternatives --config java [select java 7]
# update-alternatives --config javac [select javac 7]
```

➢ Downloading and Installing *gpsd*

In order to run the BTP+GN software, a connection to a *gpsd* is required. For the moment, we choose the local gpsd, hence we hook up a known GPS receiver, install the software, and start the required service:

```
# apt-get install gpsd
# update-rc.d gpsd defaults
# service gpsd start
[hookup a GPS receiver and test]
```

➢ Downloading GN+BTP

Now we can download the software from Alex's repository:

```
# cd
# git clone https://github.com/alexvoronov/geonetworking
```

➢ Building and Installing GN+BTP

Thanks to the 'maven' build environment, building and installing the BTP+GN software is as easy as:

```
# cd
# cd geonetworking
# JAVA_TOOL_OPTIONS=-Dfile.encoding=UTF8 mvn clean install
```

If a test fails during the build, you can omit them using:

```
# JAVA_TOOL_OPTIONS=-Dfile.encoding=UTF8 mvn clean install -DskipTests
```

➢ Running GN+BTP

Once the *maven* build of the BTP+GN software has succeeded, one can start the BTP+GN part with

```
# cd
```

```
# cd geonetworking
# <TB DESCRIBED LATER!! SEE ALSO GITHUB PAGE ALEXEY VORONOV>
```

3. Installing and Configuring *udp2eth* on ALIX APU1D

In this section we describe how to obtain and install *udp2eth* on the ALIX APU1D.

➢ Downloading *udp2eth*

You can obtain *udp2eth* from GitHub:

```
# git clone https://jandejongh/udp2eth
```

➢ Building *udp2eth*

Building *udp2eth* is done as follows:

```
# cd
# cd udp2eth
# make
```

➢ Installing *udp2eth*

In order to install *udp2eth* in the **`/usr/local`** tree:

```
# make install
```

➢ Configuring *udp2eth*

Configuring *udp2eth* is best explained by showing its '**`--help`**' output:

```
root@voyage:~/udp2eth# udp2eth --help
udp2eth version 0.2
Copyright (C) 2011-2015 TNO/GCDC/I-GAME

Usage: udp2eth [OPTIONS]

    -h, --help                          Show this help
        --version                       Show version information
    -l, --list-parameters-only          List parameters on stderr and exit
    -v, --verbose                       Verbose debugging
        --device=VALUE                  (Ethernet) Device name
    -p, --promiscuous                   Attempt to open device in promiscuous mode
        --server=VALUE                  Server address
        --client=VALUE                  Client address

root@voyage:~/udp2eth# udp2eth -l
[udp2eth] Device name    : wlan0
[udp2eth]   Promiscuous : No
[udp2eth] Server address: 0.0.0.0:1235
[udp2eth] Client address: 127.0.0.1:1236

[udp2eth] Listing parameters only, exiting!
```

In the example above, the tool listens on *any* interface (0.0.0.0) to UDP port 1235 (i.e., the **`server`** argument), and forwards the UDP payload to interface **`wlan0`**, using a raw socket. In reverse direction, it forwards received frames as UDP payload to port 1236 on **`localhost`** (**`127.0.0.1`**, the **`client`** argument).

For debugging, add the **`–v`** flag; for promiscuous mode (e.g., remote **`wireshark`** sniffing), use the **`–p`** flag.

4. Installing and Configuring *u2epy* on ALIX APU1D

This section describes *u2epy*, the Python variant of *udp2eth*.

➢ Downloading *u2epy*

You can obtain *udp2eth* from GitHub:

```
# cd
# git clone https://alexvoronov/u2epy
```

[currently fails due to restricted access?]

➢ Building *u2epy*

TBW

➢ Installing *u2epy*

TBW

➢ Configuring *u2epy*

TBW

5. Installing and Configuring *OpenVPN Client* on ALIX APU1D

In this section we describe how to install and configure an OpenVPN client on the ALIX APU1D.

➢ Installing the OpenVPN Client

In order to obtain the client, we can simply use **`apt-get`**:

```
# apt-get install openvpn
```

➢ Configuring the OpenVPN Client

In order to configure the **`OpenVPN`** client, we need three things:

- A certificate from the server, typically named **`XXX.crt`** where XXX refers to the name under which the client is known at the server;
- A key (file), typically named **`XXX.key`** (this file is supposed to be kept *secret*);
- An **`OpenVPN`** client configuration file, typically named **`XXX.ovpn`** which we shall create in our home directory in a new directory **`~/openvpn-tno`**.

```
# cd
# mkdir openvpn-tno
# cd openvpn-tno
# vi XXX.ovpn
```

[Insert the following contents into the file:]

```
setenv FORWARD_COMPATIBLE 1
client
server-poll-timeout 4
nobind
remote 95.211.189.156 1194 tcp

ca ca.crt
```

```
cert XXX.crt
key XXX.key

dev tap

dev-type tap
reneg-sec 604800
sndbuf 100000
rcvbuf 100000

comp-lzo no
verb 3
setenv PUSH_PEER_INFO
```

➢ The TNO OpenVPN Server for i-Game

TNO runs an **OpenVPN** server for participants of i-Game:

- **Address:** **95.211.189.156**
- **Protocol:** **TCP**
- **Port:** **1194**

➢ Running the OpenVPN Client

In order to run the OpenVPN client:

```
# cd
# cd openvpn-tno
# openvpn XXX.ovpn
```

6. Way Ahead

Script Dennis; /etc/rc.d stuff; service file ; say it is not wise to put OpenVPN client in startup scripts.

TBW

Uit Doc Dennis:

sudo apt-get install openjdk-7-jre-headless
Copy the file 'vehicle-adapter-0.0.1-SNAPSHOT-jar-with-dependencies.jar' to the gateway.
Copy the APU4.sh script to the same directory and modify this file to your needs. Take special care to update the 'SimulinkIP' field for your needs.
Copy the logback.xml file to the same directory.
copy gateway boot file to /dev/init.d and chmod+x
'Update-rc.d gateway defaults'

7. Resources

1. Jan de Jongh (ed.), Jacco van de Sluis, Dennis Heuven, Alex Voronov and Igor Passchier, "*IEEE 802.11p [CTU-IIG] on ALIX APU1D running Voyage*", v0.8, 2016, *http://www.gcdc.net*.
2. *https://github.com/jandejongh/udp2eth*
3. *https://github.com/alexvoronov/utoepy*
4. *http://www.gcdc.net*

5. *https://github.com/alexvoronov/geonetworking*