IEEE 802.11p [CTU-IIG] on PCEngines APU1D running Voyage

Authors: Jan de Jongh (ed), Jacco van de Sluis, Dennis Heuven, TNO, The Netherlands
Alex Voronov, VIKTORIA, Sweden
Igor Passchier, TASS International, The Netherlands
Version/Date: v0.7, 20160211
Releasability: PUBLIC

Introduction

This document describes how to create a 802.11p-capable ("ITS-G5") device from an PC Engines APU1D and a suitable WLAN-card. ***This is hardly original work: Most material found here was taken from existing documentation and combined with our experiences.*** See the 'Resources' section for references to documentation from our sources. Much of the contents of the current section describe the main motivations for switching to a new "ITS-G5 implementation", and the design choices made, which may not be of interest to all readers.

1.1 Current Situation

The so-called *ITS Gateways* used by TNO and TASS (and probably most former GCDC[1] 2011 participants) in various research projects and deployments (DITCM) need updating. The ITS Gateways come in various (sometimes exotic) software and hardware configurations, but share the following characteristics (well, those relevant to this discussion):

- ALIX system board (PC Engines) as a hardware platform with AMD Geode, 256 MB RAM, typically ALIX.2, ALIX.3 or ALIX.6 boards with one or two mini-PCI slots;
- One of two Atheros-based mini-PCI WLAN 802.11 card(s)s in the mini-PCI slot(s), like the Mikrotik R52H with Atheros AR5414 (the choice of card type is critical; the card has to support the ITS G5 frequencies and the half-clock-rate operation);
- Voyage-Linux on an on-board (and sometimes *very* hard to replace/remove) CompactFlash card (typically ranging from 1 to 8 GB);
- A modified Linux/compat-wireless combo, providing IEEE 802.11p support for the ath5k driver;
- Modified iw to support the IEEE 802.11p extensions;
- Modified regulatory database in order to allow transmission in ITS-G5 (A/B sometime even D as well);
- The Peek/Imtech developed (with support from TNO) *geonet daemon*, *geonetd*, implementing ETSI GeoNetworking through a dedicated (ASN.1-based) interface.

[1] See http://www.gcdc.net/ for the upcoming 2016 GCDC

1.2 Motivations (and Wish-List) for Upgrade

Below are several motivations for upgrading the ITS Gateways (in arbitrary order):

1. Support for PCIe devices (ath9k)

The mini-PCI-based WLAN boards used in the ITS Gateways are becoming increasingly rare and hard to find. Nowadays, many WLAN cards are equipped with a PCIe interfaces. Therefore, support for PCIe devices is highly desirable. For Atheros-based solutions, this means moving away from ath5k in favor of ath9k (and possibly others).

2. Support for newer kernels

It has proven hard to maintain the required kernel/compat-wireless patches for newer kernels (say > 2.6.x). Also, more and more of the required functionality for ITS-G5 becomes available in stock kernels. It is essential to keep reasonably up-to-date with the kernel revisions so that modern kernel features become available on the ITS Gateways, and to keep minimizing the required kernel patches (let alone make sure the patches do not clash with kernel developments).

3. A Java-capable platform (higher CPU power and more memory)

A serious problem with the current ITS Gateway platforms is their inability to properly run Java applications. With the availability of an open-source Java-based GeoNetworking implementation, it becomes highly desirable to upgrade to a Java-capable platform.

4. Getting rid of the CF Card in favor of a SSD

The CF cards currently used in the ITS Gateways are often hard to remove, and are known to support only a limited amount of read/write cycles. In practice, this leads to disabling writing to the CF card in serious deployments in order to protect the card. This, for instance, makes logging difficult. A high-capacity (high-speed) SSD card is therefore highly desirable.

5. An alternative to the Peek/Imtech geonetd

The Peek/Imtech geonetd is no longer maintained openly, and formally, it has never been open-source. Although this at one time was the intention of both Peek/Imtech and TNO in the context of GCDC-2011 and the Spits project, the initiative just died out and Peek/Imtech continued to further develop the geonetd for their own (increasingly commercial) purposes. And there is nothing wrong with that, given the imbalance in efforts put into the geonetd development (Peek/Imtech invested far more resources in this). However, this means we have to consider the use of other GeoNetworking implementations.

6. An easier (non-ASN.1-based) interface towards GeoNetworking

The interface to the current geonetd relies on ASN.1, which is not easily accessible for many developers. An alternative interface to GeoNetworking is therefore desirable.

7. Running GeoNetworking on a different platform

The ETSI GeoNetworking standards are still under development. In the current ITS Gateway deployments, these development are hard to keep up with. Not only because the *geonetd* software is not maintained in an open-source manner, but also because the ITS Gateways are difficult to

maintain: they run patched kernels, and require special skills to operate the Voyage-specifics. Often, these patched system components are installed directly (evading the distribution's package management system like dpkg/apt-get). In short, one does not gladly mess with the ITS Gateway software configuration (if safely accessible at all), and non-stable components like GeoNetworking should be maintained on other, more experiment-capable platforms.

8. Use of a (more) standard Linux distribution

Although over the year we have become use to and started to love the Voyage Linux distribution, we also recognize it has some drawbacks, notably the fact that it deviates from a standard distribution (Debian). Although, citing Ronald in 't Velt (TNO), "you can often get away with installing a Debian package", it is clear that there are several disadvantages to Voyage over (e.g.) Debian. This mostly has to do with the package dependencies and the repositories used. Putting it simply, package-management software like apt-get/dpkg is perfectly capable of finding dependencies among to-be-installed packages and keeping your system "consistent", *as long as you remain within a "realm" of repositories.* The moment you start building packages of your own, use non-standard repositories, or, even worse, start mixing repositories, you can expect serious problems in the long run, as each repository has its own conventions in terms of packaging viewpoints, version/revision management and development/source packages.

9. Building on the ITS Gateway

Currently, software developments for the ITS Gateway take place on 'other' machines, often using cross-compilation. In view of some requirements listed above (a faster platform with more memory and SSD), it should be possible to develop, build, deploy and configure software updates for the ITS Gateway directly on the gateway itself.

1.3 Directions and Considerations for Upgrade

Given the motivations and wish-list for the upgrade in the previous section, we have following directions for the upgrade:

1. PC Engines APU1D [>= 2GB] with SSD: This covers motivations **1**, **3** and **4**, and hopefully **8** and **9** as well.

2. Keep using voyage-Linux; evading installation of X-Windows, Gnome, or worse: The PC Engines APU1D are known to run Voyage Linux [even xf86_64]. This actually contradicts point **8** from the previous section, but we're probably not ready to leave the Voyage distribution :-).

3. Open-source IEEE 802.11p stack by CTU-IIG: The open-source stack designed by CTU-IIG is a very serious and promising attempt to include support for IEEE 802.11p in a stock Linux kernel. At the present time, the use of the CTU-IIG software requires modified versions of the Linux kernel and the **iw** tool, but this appears to be well documented. With CTU-IIG, one also needs to add entries for ITS-G5 operation in the *regulatory database*, but this (at first sight) appears to be no different from earlier initiatives (like the GCDC open-source patches used in GCDC-2011).

This covers **2** in particular.

4. Open-source UDP encapsulation of Ethernet frame by Jan de Jongh/TNO

Once a suitable “wlan0” device is available on the board (pre-set to some ITS-G5 channel), one may consider the use of a small daemon that provides direct access (AF_PACKET) to the interface from another machine. That way, one can run higher-layer software (like GeoNetworking, see below) on another machine.

One such daemon is the udp2eth program by the author, and available from GitHub, https://github.com/jandejongh/udp2eth.

The use of udp2eth (or a similar solution) covers **7**.

5. Open-source GeoNetworking by Alex Voronov/Viktoria

The open-source implementation of GeoNetworking by Alex Voronov from VIKTORIA Institute should be deployable on the PC Engines APU1D boards, so a GeoNetworking-capable single-board solution becomes available again (like the ITS Gateways we have already, based on earlier ALIX boards). This covers **5**, **6** and **7**.

The only step we do not yet take is to migrate away from Voyage Linux (point **8**). There is no particular reason other than that many people involved are highly familiar with it, and that it installs flawlessly without X-Windows and desktop environments.

Step-by-step Installation Guide

This section provides a step-by-step guide to installing 802.11p drivers from CTU-IIG on top of a recent Voyage Linux system (on an PC Engines APU1D or PC Engines APU1D4 with 4GB RAM). The description is specific to Voyage version 0.9.5, but has been applied to Voyage 0.10.0 as well.

2 Installing and Configuring Voyage Linux

For this we use the serial port of the device in combination with a terminal window or emulator program (eg. PuTTY). The serial setting for the APU are: speed 115200, data bits 8, stop bits 1 and parity none. The voyage distribution uses default credentials user=root, password=voyage.

2.1 Upgrading the PC Engines APU1D BIOS

First we upgrade the PC Engines APU1D firmware to the latest version:

- Prepare a USB stick with TinyCore, see http://www.pcengines.ch/tinycore.htm.
- Add a recent version of Voyage Linux on the USB stick, e.g., http://www.voyage.hk/download/voyage/amd64/voyage-0.9.5_amd64.tar.bz2.

After booting, enter

```
$ flashrom -w apu140908.rom
```

If needed, correct the ROM-file name. Below we show a (condensed) screen copy of the output:

```
TinyCore www.tinycorelinux.com

searching for home directory ...

Welcome to TinyCore running on APU
To update the BIOS type "flashrom -w apu140908.rom"
```

```
[+41.9 C][root@box:/mnt/sdb]$ flashrom -w apu140908.rom
flashrom v0.9.7-r1711-APU on Linux 3.8.13-tinycore (i686)
flashrom is free software, get the source code at http://www.flashrom.org

Using default programmer "internal".
Calibrating delay loop... OK.
coreboot table found at 0x7efdf000.
Found chipset "AMD SB7x0/SB8x0/SB9x0". Enabling flash write... OK.
Found Macronix flash chip "MX25L1605A/MX25L1606E" (2048 kB, SPI) at physical
address 0xffe00000.
Reading old flash chip contents... done.
Erasing and writing flash chip... Erase/write done.
[+51.5 C][root@box:/mnt/sdb]$
```

2.2 Repartition the SSD

It is recommended to partition the SSD into a single `/boot` partition and multiple system/data partitions (standard Linux convention nowadays). We use a single `/boot` partition and two system partitions. The primary purpose is to be able to upgrade the system while leaving the original system intact until the new system is fully tested.

Below we show how to do this using `fdisk`. Do not blindly copy this approach because it destroys all your data! Make sure to check where the SSD is located (in our case, **/dev/sda**).

```
[+50.6 C][root@box:/tmp/voyage-0.9.5_amd64]$ pwd

/tmp/voyage-0.9.5_amd64

[+50.6 C][root@box:/tmp/voyage-0.9.5_amd64]$ ls -l
total 120
-rwxr-xr-x    1 root     root         28880 Nov  5  2014 CHANGELOG
-rwxr-xr-x    1 root     root         23367 Nov  5  2014 README
-rw-r--r--    1 root     root          4264 Dec 18  2013 README.live-cd
-rw-r--r--    1 root     root          4898 Dec 18  2013 README.pxe
drwxr-xr-x    2 root     root          2120 Aug  3 04:38 bin
drwxr-xr-x    2 root     root           300 Aug  3 04:39 boot
drwxr-xr-x    3 root     root          2440 Aug  3 04:38 dev
drwxr-xr-x   65 root     root          2640 Aug  3 04:38 etc
drwxr-xr-x    2 root     root            40 Sep 21  2014 home
lrwxrwxrwx    1 root     root            30 Aug  3 04:39 initrd.img ->
boot/initrd.img-3.14.12-voyage
drwxr-xr-x   18 root     root           580 Aug  3 04:39 lib
drwxr-xr-x    2 root     root            60 Aug  3 04:37 lib64
drwxr-xr-x    2 root     root            40 Nov  4  2014 media
drwxr-xr-x    2 root     root            40 Sep 21  2014 mnt
drwxr-xr-x    2 root     root            40 Nov  4  2014 opt
drwxr-xr-x    2 root     root            40 Sep 21  2014 proc
drwxr-xr-x    3 root     root            60 Aug  3 04:39 ro
drwx------    2 root     root            80 Aug  3 04:39 root
drwxr-xr-x    9 root     root           220 Aug  3 04:39 run
drwxr-xr-x    2 root     root            40 Nov  5  2014 rw
drwxr-xr-x    2 root     root          2700 Aug  3 04:39 sbin
drwxr-xr-x    2 root     root            40 Jun 10  2012 selinux
drwxr-xr-x    2 root     root            40 Nov  4  2014 srv
drwxr-xr-x    2 root     root            40 Jul 14  2013 sys
drwxr-xr-x    3 root     root            80 Aug  3 04:38 tftpboot
drwxrwxrwt    2 root     root            40 Nov  5  2014 tmp
drwxr-xr-x   10 root     root           200 Aug  3 04:38 usr
drwxr-xr-x   11 root     root           260 Aug  3 04:38 var
lrwxrwxrwx    1 root     root            27 Aug  3 04:38 vmlinuz ->
boot/vmlinuz-3.14.12-voyage
-rw-r--r--    1 root     root             0 Nov  5  2014 voyage.1st
```

```
-rw-r--r--    1 root     root         15789 Nov  5  2014 voyage.depends.list
-rw-r--r--    1 root     root         23525 Nov  5  2014 voyage.dpkg-l
-rw-r--r--    1 root     root          4526 Nov  5  2014 voyage.dpkg.list

[+50.5 C][root@box:/tmp/voyage-0.9.5_amd64]$ df -h

Filesystem                Size      Used Available Use% Mounted on
rootfs                    3.1G    274.0M      2.8G   9% /
tmpfs                     1.7G         0      1.7G   0% /dev/shm
/dev/sdb                  3.8G     76.7M      3.7G   2% /mnt/sdb
/dev/sr0                  6.7M      6.7M         0 100% /mnt/sr0

[+50.8 C][root@box:/tmp/voyage-0.9.5_amd64]$ fdisk /dev/sda

Device contains neither a valid DOS partition table, nor Sun, SGI, OSF or GPT
disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that the previous content
won't be recoverable.


The number of cylinders for this disk is set to 7297.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): p

Disk /dev/sda: 60.0 GB, 60022480896 bytes
255 heads, 63 sectors/track, 7297 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks  Id System

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-7297, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-7297, default 7297): +4192M

Command (m for help): p

Disk /dev/sda: 60.0 GB, 60022480896 bytes
255 heads, 63 sectors/track, 7297 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks  Id System
/dev/sda1               1         511     4104576  83 Linux

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (512-7297, default 512): Using default value 512
Last cylinder or +size or +sizeM or +sizeK (512-7297, default 7297): +28672M

Command (m for help): p
```

```
Disk /dev/sda: 60.0 GB, 60022480896 bytes
255 heads, 63 sectors/track, 7297 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id System
/dev/sda1               1         511     4104576   83 Linux
/dev/sda2             512        3998    28009327+ 83 Linux

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (3999-7297, default 3999): Using default value 3999
Last cylinder or +size or +sizeM or +sizeK (3999-7297, default 7297): Using
default value 7297

Command (m for help): p

Disk /dev/sda: 60.0 GB, 60022480896 bytes
255 heads, 63 sectors/track, 7297 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id System
/dev/sda1               1         511     4104576   83 Linux
/dev/sda2             512        3998    28009327+ 83 Linux
/dev/sda3            3999        7297    26499217+ 83 Linux

Command (m for help): w

The partition table has been altered.
Calling ioctl() to re-read partition table

[+51.5 C][root@box:/tmp/voyage-0.9.5_amd64]$
```

Reboot now if the kernel still uses the old partition table (and warns about that!).

Now create ext3 filesystems as shown below:

```
[+51.5 C][root@box:/tmp/voyage-0.9.5_amd64]$ mke2fs -j /dev/sda1

mke2fs 1.42.7 (21-Jan-2013)
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
257024 inodes, 1026144 blocks
51307 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1052770304
32 block groups
32768 blocks per group, 32768 fragments per group
8032 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
```

```
Writing superblocks and filesystem accounting information: done

[+52.5 C][root@box:/tmp/voyage-0.9.5_amd64]$ mke2fs -j /dev/sda2

mke2fs 1.42.7 (21-Jan-2013)
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
1753088 inodes, 7002331 blocks
350116 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=0
214 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[+57.2 C][root@box:/tmp/voyage-0.9.5_amd64]$ mke2fs -j /dev/sda3

mke2fs 1.42.7 (21-Jan-2013)
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
1656480 inodes, 6624804 blocks
331240 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=0
203 block groups
32768 blocks per group, 32768 fragments per group
8160 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

2.3 Install and boot Voyage Linux on an PC Engines APU1D

Next, still from TinyCore, we unpack the Voyage Linux distribution on the SSD. Boot up with USB stick + TinyCore linux (for Voyage 0.9.5; obvious command-line substitutions are required for other versions):

1. `[+51.5 C][root@box:/mnt/sdb] cp voyage-0.9.5_amd64.tar.bz2 /tmp/`
2. `[+51.5 C][root@box:/mnt/sdb] cd /tmp/`
3. `[+51.5 C][root@box:/tmp] tar xfvj voyage-0.9.5_amd64.tar.bz2`
4. `[+51.5 C][root@box:/tmp] cd voyage-0.9.5_amd64`

Notes :

- For Voyage 0.10.0 (and possibly beyond), change the command to unpack the Voyage archive as follows:
  - `[+51.5 C][root@box:/tmp] tar xfv voyage-0.10.0_amd64.tar.xz`

  This makes the decompression of the archive independent of the compression method used (which, apparently, has changed from **`.bz2`** into **`.xz`**).

Start the Voyage-Linux installation script :

1. `[+51.5 C][root@box:/tmp/ voyage-0.9.5_amd64] cd /`
2. `[+51.5 C][root@box:/] /tmp/voyage-0.9.5_amd64/usr/local/sbin/voyage.update`

.

More information is available on http://pcengines.ch/howto.php. The description below is for Voyage 0.9.5, however, it also works with Voyage 0.10.0 with obvious substitutions of command-line arguments.

Below we show a screenshot of running the Voyage-Linux installation script:

```
What would you like to do?
  1 - Create new Voyage Linux disk
  2 - Update existing Voyage configuration
  3 - Exit
      (default=1 [Create new Voyage Linux disk]): 1

some mandatory options are unset, please enter them interactively
Where is the Voyage Linux distribution directory?
  (default=/tmp/voyage-0.9.5_amd64):

What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=2 [Select Target Profile - this overwrites current settings]): 2

Please select Voyage profile:
  1 - Keep existing settings
  2 - 4501
  3 - 4511/4521
  4 - 4801
  5 - 5501
  6 - 6501
  7 - ALIX
  8 - APU
  9 - Generic PC
  10 - Notebook (pcmcia)
  11 - WRAP
      (default=7 [ALIX]): 8

What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=3 [Select Target Disk]): 3
```

```
Partitions information
major minor  #blocks  name

   8        0   58615704 sda
   8        1    4104576 sda1
   8        2   28009327 sda2
   8        3   26499217 sda3
  11        0       8192 sr0
   8       16    4004864 sdb

Which device accesses the target disk [/dev/hde]? /dev/sda

Which partition should I use on /dev/sda for the Voyage system [1]? 2

Device information for /dev/sda2
device     fs_type label    mount point    UUID

/dev/sda2  ext3             (not mounted)  67d7f07d-b7a3-4918-9a5d-760f3c5a58e5

Where can I mount the target disk [/mnt/cf]?

What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=4 [Select Target Bootstrap Loader]): 4

Which loader do you want (grub or lilo) [grub]?

Which partition is used for bootstrap [1]?

What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=5 [Configure Target Console]): 5

Select terminal type:
  1 - Serial Terminal
  2 - Console Interface
      (default=1 [Serial Terminal]): 1

Please choose speed:
  1 - 2400
  2 - 4800
  3 - 9600
  4 - 19200
  5 - 38400
  6 - 57600
  7 - 115200
      (default=7 [115200]): 7

What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
```

```
  6 - Partition and Create Filesystem
      (default=6 [Partition and Create Filesystem]): 6

What shall I do with your Flash Media?
  1 - Partition Flash Media and Create Filesystem
  2 - Use Flash Media as-is
      (default=1 [Partition Flash Media and Create Filesystem]): 2

What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=7 [Copy Distribution to Target]): 7


Configuration details:
----------------------

Distribution directory:   /tmp/voyage-0.9.5_amd64

Disk/Flash Device:        /dev/sda
Installation Partition:   /dev/sda2
Bootstrap Partition:      /dev/sda1

Will be mounted on:       /mnt/cf

Target system profile:    APU
Target console:           serial
Target baud rate:         115200

Bootstrap installer:      grub
Bootstrap partition:      /dev/sda1

OK to continue (y/n)? y

Ready to go ....
Copying files .... done

Removing pcmcia from update-rc.d
chroot: can't execute 'update-rc.d': Exec format error <is this bad?>
Removing dnsmasq.pxe.conf in /etc/dnsmasq.more.conf
Reconfiguring resolvconf
chroot: can't execute 'sh': Exec format error <is this bad?>
Updating /etc/hosts
Installing grub
Copy grub files from /mnt/cf to /mnt/cf/rw/grub
Setting up grub under chroot /mnt/cf
copyfiles.sh script completed
What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=8 [Exit]): 8

[+53.5 C][root@box:/tmp/voyage-0.9.5_amd64]$
```

At this point, reboot in order to see if Voyage Linux starts properly. Do not proceed until

Voyage Linux comes up with a login prompt! Note that in many cases, a 'warm reboot' leads to a kernel panic. However, a 'cold reboot' seems to solve this (we have not had problems since).

2.4 Post-Installation Configuration

At this point, we have successfully deployed a Voyage distribution on **/dev/sda2** with a '**grub**' bootloader on (the boot-loader sections of) **/dev/sda1** (please note, these device names may be different in your specific case!). However, the Voyage install script has put the (its) kernel and initrd (and related) files need in a **/boot** directory on **/dev/sda2**, and this is not the way we want it; we really want these to be in the separate /boot **(/dev/sda1)** partition.

Configure a separate /boot filesystem

After running the Voyage-Linux installation script and rebooting, we move the contents of **/boot** (on **/dev/sda2**) on a separate filesystem designated for booting (e.g., holding all installed kernel versions):

```
# remountrw
# mkdir /mnt/tmp
# mount /dev/sda1 /mnt/tmp
# cp -drp /boot/* /mnt/tmp/
# vi /mnt/tmp/grub/menu.lst
```

Add the following entry (preferably as default entry):

```
title Voyage Linux 0.9.5 (Build Data 20141105) from /boot
root(hd0,1)
kernel (hd0,0)/vmlinuz-3.14.12-voyage root=LABEL=VOYAGE_FS console=ttyS0,115200n
initrd (hd0,0)/initrd.img-3.14.12-voyage
```

The easiest way to do this is to copy the existing boot entry for Voyage and simply prepend the **(hdX…)** lines. Note that obvious substitutions are required for other Voyage releases!

Now boot from the new entry and check whether Voyage Linux comes up.

After logging in as root again, clean the `/boot` directory (be careful that you did not already mount the **/boot** directory!) and add an entry to `/etc/fstab` for mounting **/dev/sda1** under **/boot** (again, make sure that this is in agreement with your own configuration):

```
# rm /boot/*
# vi /etc/fstab
```

Add the /boot entry:

```
/dev/sda1     /boot   ext3    defaults      0     0
```

Now mount `/boot`:

```
# cd /
# umount /mnt/tmp if still needed!
# rmdir /mnt/tmp
```

```
# mount -a
```

Check the presence and contents of the **/boot** partition.

Reboot again (using the right entry), check for correct boot, login as root check again for the presence and contents of the **/boot** partition. At this stage, it is unlikely that the directory is empty or contains 'unexpected' files.

Edit boot menu and remove Voyage symbolic links in /

remountrw
rm /vmlinuz /initrd.img
vi /boot/grub/menu.lst

- ➔ Remove original Voyage-Linux entry (optional).
- ➔ Rename new entry (reassess that it is the default entry).

```
serial --speed=115200
terminal serial

timeout 5
default 0

title Voyage Linux 0.9.5 (Build Data 20141105)
root(hd0,1)
# Note: the 'console' part is really on the same line!
kernel (hd0,0)/vmlinuz-3.14.12-voyage root=LABEL=VOYAGE_FS
console=ttyS0,115200n8
initrd (hd0,0)/initrd.img-3.14.12-voyage
```

Reboot again, and check!

Other post-installation configuration

For additional post-install configuration, consider the following:

- Ethernet/IPv4/IPv6 configuration in **/etc/network/interfaces**.
- Install and enable **sshd** (if not done so already).
- Configure the dpkg repositories if needed.
- Make sure you have a working Internet connection from the PC Engines board for the next steps.

3 Building, Installing and Configuring CTU-IIG 802.11p Linux

3.1 Install required Debian packages (root)

For the next steps, in particular the CTU-IIG driver installation, a few packages are needed (some may be installed already):

- `# remountrw`
- `# apt-get update`
- `# apt-get upgrade`
- `# apt-get install git`
- `# apt-get install kernel-package libncurses-dev fakeroot wget bzip2 sharutils`

- `# apt-get install bc`
- `# apt-get install openssl`
- `# apt-get install pkg-config`

The fifth line is the "classic" action before compiling custom kernels on Debian(-based) systems. Should **`libncurses-dev`** not be found, you may want to try **`ncurses-dev`**. We found out that in our particular case, the **`bc`** package is required as well in order to compile Linux (probably, this is installed by default on most Debian deployments, but removed from the Voyage Linux base install). The **`pkg-config`** package was needed in order to compile the **`iw`** tool.

3.2 Install CTU-IIG 802.11p Linux

3.2.1 802.11p-linux

- Follow the instructions on https://gist.github.com/lisovy/80dde5a792e774a706a9 for the kernel, copied below.
- Make sure you enable (static or as a module) the **RealTek r8169** Ethernet driver in the **make menuconfig** step below, or else your Ethernet will not work with the new kernel. It is under Device drivers / network device support / ethernet driver support.

1. **# cd**
2. **# git clone** https://github.com/CTU-IIG/802.11p-linux.git
3. **# cd 802.11p-linux/**
4. **# git checkout its-g5_v3**
5. **# mkdir _build**
6. **# make O=_build x86_64_defconfig**
7. **# cd _build**
8. **# make menuconfig**
9. **# make -j2**
10. **# make modules_install**
11. **# make install**

Notes:

- In the **`O=_build`** line we use capital-letter **`O`**, not zero.
- After rebooting, check the boot configuration, for instance in **/boot/grub/menu.lst**. In our case, the last (necessary) step overwrites the symlink in the root / to the kernel and initramfs. Possibly, this is a conflict between the ways in which Voyage Linux maintains kernels, and the way stock Debian does it. We haven't looked into this in more detail, but we consider to remove the symlinks in the root directory, and remove the stock Voyage boot entry in the **menu.lst** file (the **make install** did create entries for the stock Voyage Linux kernel as well).
- A `.config` file (to be placed in the `_build` directory, overwriting the generated kernel config file, is available). Copy it into the `_build` directory *after* the **make O=_build… command.**
- Put the new kernel at the head of the list of kernel entries in **/boot/grub/menu.lst** (or use another way to make sure that an unattended (re)boot starts the right kernel). If you want console output (on the serial RS-232 connection) during boot (of any entry), add the following the 'kernel' field of an entry: **`console=ttyS0,115200n8`**.
- Although in our case, the new kernel works, it is unknown if the ALIX-specific features in the kernel configuration are also present in the CTU-IIG kernel config (which seems to be a stock Debian kernel configuration file). We still intend to somehow merge these two

configuration files.

- Should you get the following message during the **`# git clone`** command:

  # git clone https://github.com/CTU-IIG/802.11p-linux.git
  Cloning into '802.11p-linux'...
  error: Problem with the SSL CA cert (path? access rights?) while accessing https://github.com/CTU-IIG/802.11p-linux.git/info/refs
  fatal: HTTP request failed

  Then try the following:

  # git config --global http.sslVerify false

3.2.2 802.11p-iw

After making sure the **`pkg-config`** package is installed, compile and install the modified **`iw`** tool as explained in https://gist.github.com/lisovy/80dde5a792e774a706a9, again copied below:

1. **# cd**
2. **# apt-get install libnl-dev**
3. **# git clone https://github.com/CTU-IIG/802.11p-iw.git**
4. **# cd 802.11p-iw**
5. **# git checkout its-g5_v3**
6. **# make**
7. **# PREFIX=/ make install**

As suggested in https://gist.github.com/lisovy/80dde5a792e774a706a9, you can test the new **`iw`** features with [output shown in red]:

1. **# /sbin/iw | grep -i ocb**
2. **# dev <devname> ocb join <freq in MHz> <5MHZ|10MHZ>**
3. **# dev <devname> ocb leave**

Notes:

- We actually had lots of trouble with the **`# apt-get install libnl-dev`** command; at many times it just failed complaining it could not find the package. We found the following workarounds:
  - In the **`Makefile`**, replace '**`ocb.c`**'with '**`ocb.o`**'in the '**`OBJS =`** ...' line.
  - **`# apt-get install libnl-3-dev`**
  - **`# apt-get install libnl-genl-3-dev`**
  - Then try '**`# make clean`**' and '**`# make`**' again.

3.2.3 802.11p-wireless-regdb

Compile and install the modified **`wireless-regdb`** as explained in https://gist.github.com/lisovy/80dde5a792e774a706a9, again copied below:

1. **# apt-get install python-m2crypto**
2. **# git clone https://github.com/CTU-IIG/802.11p-wireless-regdb.git**
3. **# cd 802.11p-wireless-regdb**
4. **# git checkout its-g5_v1**
5. **# make**
6. **# sudo PREFIX=/ make install**

Customizing the regulatory database

The default installation only allows the ITS frequencies for Germany (DE) in the regulatory database. In order to change this (or add more countries), you need to edit **db.txt** in the **802.11p-wireless-regdb** directory. The entry to look for looks something like:

```
# For ITS-G5 evaluation
(5850 – 5925 @ 20), (100 mW), NO-CCK, OCB-ONLY
```

Copy both lines to the applicable country entries (in our case: BE, DE, ES, FR, GE, GB, LV, NL, SE). Do not forget to **make** and **make install** after changing the db.txt.

3.2.4 802.11p-crda

Compile and install the modified **crda** as explained in https://gist.github.com/lisovy/80dde5a792e774a706a9, again copied below:

1. **# apt-get install python-m2crypto**
2. **# apt-get install libgcrypt11-dev**
3. **# git clone https://github.com/CTU-IIG/802.11p-crda.git**
4. **# cd 802.11p-crda**
5. **# git checkout its-g5_v1**
6. **# cp /lib/crda/pubkeys/root.key.pub.pem pubkeys/**
7. **# make**
8. **# PREFIX=/ REG_BIN=/lib/crda/regulatory.bin make install**

Location of the crda directory [/lib/crda versus /usr/lib/crda]

It appears that in Voyage Linux, the `crda` directory is in /lib instead of the (expected) /usr/lib. This can be solved easily through:

1. **# cd /usr/lib**
2. **# ln –s /lib/crda**

After which you should repeat the procedure for building and installing **802.11p-crda** as outline above again.

Notes:

- We ignore the following message (we do not understand it):
  - **Country 00: invalid**
- Check the (binary) regulatory database (for your country):

1. **# regdbdump /lib/crda/regulatory.bin**

or

2. **# regdbdump /lib/crda/regulatory.bin | grep OCB**

3.3 Interface Configuration

A permanent configuration for `wlan0` can be entered in **/etc/network/interfaces** or as a single file in the **/etc/network/interfaces.d/** directory. The entry looks something like:

```
# JdJ20151204: OCB entry (802.11p)
auto wlan0
iface wlan0 inet static
      address 192.168.97.101        # replace with your IPv4 address
      netmask 255.255.255.0
      broadcast 192.168.97.255
      pre-up iw reg set NL          # replace with your CC
      pre-up iw dev wlan0 set type ocb
      post-up iw dev wlan0 ocb join 5900 10MHZ
```

3.4 Ath9k Hardware Compatibility List

The setup described in the previous sections has been tested (successfully) on:

- Unex DHXA-222 with AR9642 chipset
- Anatel with AR9280 chipset

The driver is also supposed to work with other cards with the same chipset and with the AR9380 chipset, but we did not test this. More information on hardware is available at: https://github.com/alexvoronov/geonetworking/blob/master/HARDWARE.md

4 Way Ahead

After successfully installing and configuring Voyage Linux and the CTU-IIG Linux kernel, there are several possible ways to proceed. We name only a few below.

- In order to create IEEE 802.11(p) frames, encapsulate them into UDP and send/receive them from another machine or from and application that uses UDP for ITS-G5 embedding, use utoepy (https://github.com/alexvoronov/utoepy) or udp2eth (https://github.com/jandejongh/udp2eth).

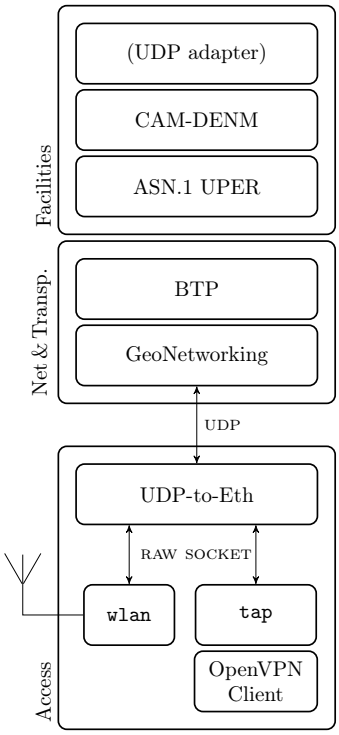
- If you want to run GeoNetworking, you may want to consider the open-source implementation by Alex Voronov (VIKTORIA Institute) available from https://github.com/alexvoronov/geonetworking.

- Use of application/facility layer software for interfacing with your vehicle control software. Or testing software like a set up for automated conformance testing using open-source TTCN-3 Eclipse Titan and ETSI ITS library. Or use Interaction Test Tool (ITT) software as provided by the i-GAME[2] project for GCDC participants to completely test the correctness of the Vehicle-2-Vehicle communications between connected vehicle clients (VeS). See for software and details: https://github.com/alexvoronov/itt-gt

Below an illustration of the ITS-G5 GeoNetworking stack is given taken from:

[2] 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.

https://github.com/alexvoronov/geonetworking



Application and Facilities layer
- Sending CAM via Single Hop Broadcast and DENM via GeoBroadcast is supported.

Network and Transport layer
- BTP and GeoNetworking – Java 7

- Interface between GN and netw.iface
  - GN has an UDP API towards lower layer native code
    - Via eth2udp (AF_SOCKET, Linux-only)
    - Or via utoepy (Python + libpcap)

Access layer
- Networking interfaces
  - wlan – normal radio network
  - tap – OpenVPN for testing using Interactive Test Tool (ITT)

Editorial notes:
Any feedback on this manual, ideas for improvements or extensions to this manual are much appreciated.

5 Resources

- http://http://wiki.voyage.hk/voyage_kernel.txt
- https://gist.github.com/lisovy/80dde5a792e774a706a9
- http://wiki.openwrt.org/toh/pcengines/apu
- http://www.pcengines.ch/pdf/apu1.pdf
- http://www.pcengines.ch/tinycore.htm
- http://pcengines.ch/howto.php
- https://github.com/alexvoronov/geonetworking
- https://github.com/alexvoronov/geonetworking/blob/master/HARDWARE.md
- https://github.com/alexvoronov/itt-gt
- https://github.com/alexvoronov/utoepy
- https://github.com/jandejongh/udp2eth
- http://www.gcdc.net/