Application Note

Akademie věd České republiky
Ústav teorie informace a automatizace AV ČR, v.v.i.

Arrowhead 4.1.3 Client on Trenz TEBF0808 + TE0808-04-6EB21A SoM Running Petalinux 2018.2 Kernel with Debian Buster File System

Lukas Kohout
kohoutl@utia.cas.cz

Revision history

Rev.	Date	Author	Description
0	06.04.2021	L. Kohout	Document creation – initial version
1			
2			

Contents

Acknowledgement

This work has been supported from project Arrowhead Tools, project number ECSEL 826452 and MSMT 8A19009.

1 Introduction

This document describes an installation procedure of the Arrowhead client running on the Trenz Electronic TE0808 HW platform (TEBF0808 carrier board [1] with Trenz Electronic TE0808-04-6EB21A System on Module [2]). The procedure covers installation of required dependencies, configuration and compilation of the Arrowhead client. There are two types of the client, the provider of the service and the consumer requesting the service. As the TE0808 HW platform is meant to be a provider of the service, this text aims to describe an installation procedure of the provider type of the client.

2 Description

To follow the steps described in this text it is assumed, that you have already gone through the steps described in the application note called *Trenz TEBF0808 + TE0808-04-6EB21A SoM Running Petalinux 2018.2 Kernel with Debian Buster File System* downloadable from http://sp.utia.cz/index.php?ids=results&id=te0808-debian-buster. It is also assumed that you have already had a running PC with Arrowhead core system 4.1.3 connected to the same network as the TE0808 HW platform is (http://sp.utia.cz/index.php?ids=results&id=aht-cs-on-ubuntu18-04). To be able to install the dependencies, the internet access is required by the running TE0808 HW platform. The network structure is shown in Figure 1.

3 Arrowhead C++ Client Installation

This section describes an installation procedure of the Arrowhead client coded in C++ on the TE0808 HW platform running Petalinux kernel with the file system based on Debian OS. There are two types of the client, the provider of the service and the consumer of the service. Both clients require the same tools and libraries to be installed.

3.1 Dependencies and Preparations

1. The image described in http://sp.utia.cz/index.php?ids=results&id=te0808-debian-buster has already had all dependencies installed, to install them on fresh system that runs on the TE0808 HW platform, from its terminal execute:

```
sudo apt-get update
sudo apt-get -y upgrade
```

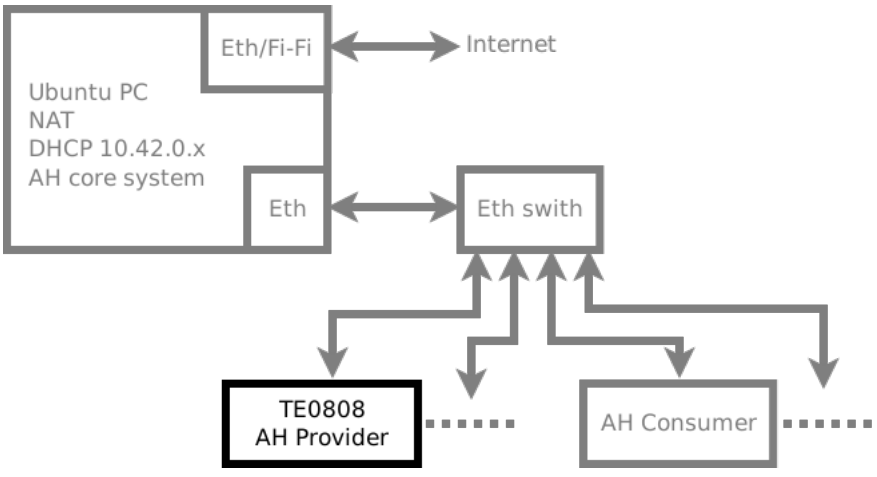

Figure 1: Network structure.

```
sudo apt-get -y install git openssl libgnutls28-dev \
     libgnutlsxx28 libssl1.1 libssl-dev libcurl4 \
     libcurl3-gnutls libcurl4-gnutls-dev libcrypto++-dev \
     libcrypto++-utils libcrypto++6 libgpg-error-dev \
     automake texinfo g++ libjson-c-dev libjsoncpp-dev

wget https://ftp.gnu.org/gnu/libmicrohttpd/libmicrohttpd-
0.9.59.tar.gz

tar -xvzf libmicrohttpd-0.9.59.tar.gz

cd libmicrohttpd-0.9.59
./configure --with-gnutls

make
sudo make install

sudo ln -sf /usr/local/lib/libmicrohttpd.so.12.46.0 \
     /usr/lib/libmicrohttpd.so.12
```

2. Download Arrowhead client source codes, from the TE0808 HW platform terminal:

```
cd ~/
mkdir -p ah-client
cd ah-client
git clone https://github.com/arrowhead-f/client-cpp
```

3.2 Arrowhead Provider of Service

1. Generate certificate for the provider. This step is performed on the PC running Arrowhead core system described in http://sp.utia.cz/index.php?ids=results&id=aht-cs-on-ubuntu18-04

   a) Go to the PC where the Arrowhead core system runs.

   b) Create a destination folder where the certificate will be stored, from the terminal execute commands:

   ```
   cd ~/
   mkdir –p keys
   cd keys
   ```

   c) Start KSE (Key Store Explorer) as a super user, from the terminal execute:

   ```
   sudo kse
   ```

   d) To generate the certificate, follow steps described in:

   https://github.com/arrowhead-f/core-java-spring/blob/master/documentation/certificates/create_client_certificate.pdf

   Modify the steps in the list bellow:

   - In the 1st step of the description, the cloud certificate is located in:

     ```
     /etc/arrowhead/clouds/testcloud.p12
     ```

   - In the 5th step, the *Common name (CN)* should be:

     ```
     my_sensor.testcloud.arrowhead.arrowhead.eu
     ```

- In 13th step, set alias to:

```
my_sensor
```

- In the 15th step, save the file to the prepared folder:

```
~/keys
```

As the name of the file, use:

```
my_sensor.p12
```

e) Convert the certificate file to set of files used by source code of the provider. From the terminal execute commands:

```
cd ~/keys
sudo chown devel:devel my_sensor.p12

openssl pkcs12 -in my_sensor.p12 -out
my_sensor.testcloud.cacert.pem -cacerts -nokeys

openssl pkcs12 -in my_sensor.p12 -out
my_sensor.testcloud.clcert.pem -clcerts -nokeys

openssl pkcs12 -in my_sensor.p12 -out
my_sensor.testcloud.privkey.pem -nocerts

openssl rsa -in my_sensor.testcloud.privkey.pem -pubout
-out my_sensor.testcloud.pubkey.pem
```

Password of the first command is *devel*, all other passwords are *arrowhead*.

2. Copy all generated files to the TE0808 HW platform via SFTP protocol.

a) Get an IP address of the TE0808 HW platform, from the TE0808 HW platform terminal execute:

```
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.42.0.143  netmask 255.255.255.0  broadcast
```

b) Bind a connection to the TE0808 HW platform via SFTP protocol, from the PC where the Arrowhead core system runs execute commands:

```
cd ~/keys
sftp root@10.42.0.143
```

Password is *root*. If you will be asked to create a key pair of the connection, type *yes* and press enter.

c) Copy the certificate files, use opened SFTP connection:

```
cd /root
mkdir ah-client/client-cpp/ProviderExample/keys3
cd ah-client/client-cpp/ProviderExample/keys3
put *.pem
```

d) Close SFTP connection

```
quit
```

NOTE: The certificate files can be also copied to the TE0808 HW platform with any other SFTP client (nautilus for instance) or you can use USB flash drive. But in that case, you have to mount the drive manually, auto-mounting has not supported yet.

3. On the TE0808 HW platform, modify provider source code to use generated certificate, change files:
   - ~/ah-client/client-cpp/ProviderExample/src/Interface/Https_Handler.cpp.
     - On line 56 replace string
       `keys2/tempsensor.testcloud2.clcert.pem`
       with
       `keys3/my_sensor.testcloud.clcert.pem`
     - On line 63 replace string
       `keys2/tempsensor.testcloud2.privkey.pem`
       with
       `keys3/my_sensor.testcloud.privkey.pem`
     - On line 69 replace string
       `123456`
       with
       `arrowhead`
     - On line 73 replace string
       `keys2/tempsensor.testcloud2.caCert.pem`
       with
       `keys3/my_sensor.testcloud.cacert.pem`
     - On line 315 replace string
       `123456`
       with
       `arrowhead`
   - ~/ah-client/client-cpp/ProviderExample/src/Interface/Https_Handler.hpp.
     - On line 16 replace string
       `keys2/tempsensor.testcloud2.privkey.pem`
       with
       `keys3/my_sensor.testcloud.privkey.pem`
     - On line 17 replace string
       `keys2/tempsensor.testcloud2.clcert.pem`
       with
       `keys3/my_sensor.testcloud.clcert.pem`

- On line 18 replace string

  `keys2/tempsensor.testcloud2.cacert.pem`

  with

  `keys3/my_sensor.testcloud.cacert.pem`

- ~/ah-client/client-cpp/ProviderExample/src/Provider/ProvidedService.h.
  - On line 11 replace string

    `SecureTemperatureSensor`

    with

    `my_sensor`

  - On line 12 replace string

    `IndoorTemperature_ProviderExample`

    with

    `my_sensor_example`

  - On line 14 replace string

    `keys2/tempsensor.testcloud2.privkey.pem`

    with

    `keys3/my_sensor.testcloud.privkey.pem`

  - On line 15 replace string

    `keys2/tempsensor.testcloud2.pubkey.pem`

    with

    `keys3/my_sensor.testcloud.pubkey.pem`

4. Compile the provider, from the TE0808 HW platform terminal execute commands:

```
cd ~/ah-client/client-cpp/ProviderExample/
make clean
make
```

5. Configure the provider to run on the TE0808 HW platform, the location and name of the configuration file:

```
~/ah-client/client-cpp/ProviderExample/
ApplicationServiceInterface.ini
```

   The configuration file consists of the following items.

   - sr_base_uri – an address of the Arrowhead registration service running in insecure mode and corresponding port.
   - sr_base_uri_https – an address of the Arrowhead registration service running in secure mode and corresponding port.
   - port – a port number where the Provider will be available on, set 8000. In this case, port 8000 is dedicated for insecure mode, for secure mode the provider automatically select port 8001.
   - address – Provider IP address.

- Address6 - Provider IP address in IPV6

The configuration file example for the provider running on the TE0808 HW platform:

```
[Server]
sr_base_uri="http://10.42.0.1:8443/serviceregistry/"
sr_base_uri_https="https://10.42.0.1:8443/serviceregistry/"
port="8000"
address="10.42.0.143"
address6="[fe80::20a:35ff:fe00:2201]"
```

6. Start the provider, from the TE0808 HW platform terminal execute:

```
./ProviderExample --secureArrowheadInterface \
                  --secureProviderInterface
```

The provider registers itself in the arrowhead database. The listing of successfully started provider:

```
============================
Provider Example - v4.1.3
============================

----------------------------
ProvidedService
----------------------------
Custom URL : /this_is_the_custom_url
System name : my_sensor
Service definition : my_sensor_example
Service interface : HTTP-SECURE-JSON
Private key path : keys3/my_sensor.testcloud.privkey.pem
Public key path : keys3/my_sensor.testcloud.pubkey.pem
Meta values:

(HTTP Server) started - 10.42.0.143:8000
(HTTPs Server) started - 10.42.0.143:8001

Measured value received from: (Base Name: this_is_the_sensor_id)
Provider is not registered yet!

REGISTRATION (Secure Provider, Secure AHInterface)

pubkeyContent:
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt+GxGuV7Stnrom1arkNEYOZ8nV5OUiIUY
P61aJ5CHVwqC+lzQQAinzxtBVP/xtfg1Zyg7wALRtvc2tjU9sZcUnUF8sRRol+6x1lbXfuYUmH/Ci
OONrrOOfgs/q6zR+cKsA+iQJ06zGQ2bQuTiD9On8AOTxAIkJXeoZ+vcWSLTa9qKrs9TzBOYN1+Wp5
zOoIvlRkpPaCb6JkE+vuhBB/kt7rlIPMKWITZRh+rTloi/g+Fvbcb8WHb61KPAAKEbt6joM9SjVbs
1mYI+WqufLq7nn9QstkkgFUT+CyBqWOvpxJeBeD5joqXVlqI6n+wnt4ZVjs9CPYtKFNTnJWZbkLhz
QIDAQAB


{ "serviceDefinition": "my_sensor_example", "serviceUri":
"\/this_is_the_custom_url", "version": 1, "secure": "TOKEN",
"providerSystem": { "systemName": "my_sensor", "address": "10.42.0.143",
"authenticationInfo":
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt+GxGuV7Stnrom1arkNEYOZ8nV5OUiIU
YP61aJ5CHVwqC+lzQQAinzxtBVP\/xtfg1Zyg7wALRtvc2tjU9sZcUnUF8sRRol+6x1lbXfuYUmH\
/CiOONrrOOfgs\/q6zR+cKsA+iQJ06zGQ2bQuTiD9On8AOTxAIkJXeoZ+vcWSLTa9qKrs9TzBOYN1
+Wp5zOoIvlRkpPaCb6JkE+vuhBB\/kt7rlIPMKWITZRh+rTloi\/g+Fvbcb8WHb61KPAAKEbt6joM
9SjVbs1mYI+WqufLq7nn9QstkkgFUT+CyBqWOvpxJeBeD5joqXVlqI6n+wnt4ZVjs9CPYtKFNTnJW
ZbkLhzQIDAQAB", "port": 8001 }, "interfaces": [ "HTTP-SECURE-JSON" ],
"metadata": { "unit": "Celsius", "security": "token" } }
SendHttpsRequest: https://10.42.0.1:8443/serviceregistry/register
HTTPs Post sent (SenML baseName = this_is_the_sensor_id)
HTTPs Post return value: 400
```

```
Already registered?
Try re-registration
SendHttpsRequest:
https://10.42.0.1:8443/serviceregistry/unregister?service_definition=my_senso
r_example&system_name=my_sensor&address=10.42.0.1&port=8001
Unregistration is successful

{ "serviceDefinition": "my_sensor_example", "serviceUri":
"\/this_is_the_custom_url", "version": 1, "secure": "TOKEN",
"providerSystem": { "systemName": "my_sensor", "address": "10.42.0.1",
"authenticationInfo":
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt+GxGuV7Stnrom1arkNEYOZ8nV5OUiIU
YP61aJ5CHVwqC+lzQQAinzxtBVP\/xtfg1Zyg7wALRtvc2tjU9sZcUnUF8sRRol+6x1lbXfuYUmH\
/CiOONrrOOfgs\/q6zR+cKsA+iQJ06zGQ2bQuTiD9On8AOTxAIkJXeoZ+vcWSLTa9qKrs9TzBOYN1
+Wp5zOoIvlRkpPaCb6JkE+vuhBB\/kt7rlIPMKWITZRh+rTloi\/g+Fvbcb8WHb61KPAAKEbt6joM
9SjVbs1mYI+WqufLq7nn9QstkkgFUT+CyBqWOvpxJeBeD5joqXVlqI6n+wnt4ZVjs9CPYtKFNTnJW
ZbkLhzQIDAQAB", "port": 8001 }, "interfaces": [ "HTTP-SECURE-JSON" ],
"metadata": { "unit": "Celsius", "security": "token" } }
SendHttpsRequest: https://10.42.0.1:8443/serviceregistry/register
Provider Registration is successful!
```

3.3 Arrowhead Consumer of Service

To run a consumer of the service on the PC where the Arrowhead core system runs, follow steps described in http://sp.utia.cz/index.php?ids=results&id=aht-cs-on-ubuntu18-04

4 References

[1] Trenz Electronic, „UltraITX+ Baseboard for Trenz Electronic TE080X UltraSOM+,“ [Online]. Available: https://shop.trenz-electronic.de/en/TEBF0808-04A-UltraITX-Baseboard-for-Trenz-Electronic-TE080X-UltraSOM.

[2] Trenz Electronic, „UltraSOM+ MPSoC Module with Zynq UltraScale+ XCZU6EG-1FFVC900E, 4 GB DDR4,“ [Online]. Available: https://shop.trenz-electronic.de/en/TE0808-04-6BE21-A-UltraSOM-MPSoC-Modul-with-Zynq-UltraScale-XCZU6EG-1FFVC900E-4-GB-DDR4.

Disclaimer

This disclaimer is not a license and does not grant any rights to the materials distributed herewith. Except as otherwise provided in a valid license issued to you by UTIA AV CR v.v.i., and to the maximum extent permitted by applicable law:

(1) THIS APPLICATION NOTE AND RELATED MATERIALS LISTED IN THIS PACKAGE CONTENT ARE MADE AVAILABLE "AS IS" AND WITH ALL FAULTS, AND UTIA AV CR V.V.I. HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and

(2) UTIA AV CR v.v.i. shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under or in connection with these materials, including for any direct, or any indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or UTIA AV CR v.v.i. had been advised of the possibility of the same.

Critical Applications:
UTIA AV CR v.v.i. products are not designed or intended to be fail-safe, or for use in any application requiring fail-safe performance, such as life-support or safety devices or systems, Class III medical devices, nuclear facilities, applications related to the deployment of airbags, or any other applications that could lead to death, personal injury, or severe property or environmental damage (individually and collectively, "Critical Applications"). Customer assumes the sole risk and liability of any use of UTIA AV CR v.v.i. products in Critical Applications, subject only to applicable laws and regulations governing limitations on product liability.