5G INDUSTRIAL IOT GATEWAY SOLUTION WITH OPENWRT ON i.MX 8M PLUS

高磊
工业边缘计算产品市场经理

2022年4月28日

SECURE CONNECTIONS
FOR A SMARTER WORLD

PUBLIC

AGENDA

- OpenWRT: A Bit of History
- Getting and Building OpenWRT
- Using OpenWRT on i.MX 8M Plus EVK
- 5G Gateway Solution
- i.MX 8M Network Benchmarks
- Q&A

OPENWRT: A BIT OF HISTORY

OpenWrt is an open source project for embedded operating systems based on Linux, used primarily on embedded devices to route network traffic. Components have been optimized to be small enough to fit into the limited storage and memory available in home routers. First released in January 2006, it is a Linux distribution that offers many features not previously found in consumer-level routers.

In May 2016, OpenWrt was forked by a group of core OpenWrt contributors due to disagreements on internal process. The fork was dubbed Linux Embedded Development Environment (LEDE). The schism was reconciled a year later. Following the re-merger, announced in January 2018, the OpenWrt branding is preserved, with many of the LEDE processes and rules used. The LEDE project name was used for v17.01, with development versions of 18.01 branded OpenWrt.

https://openwrt.org/about

GETTING AND BUILDING OPENWRT
- PREPARING THE BUILD ENVIRONMENT

- Install pre-requisites, typically:

  -
    ```
    sudo apt install build-essential ccache ecj fastjar file g++ gawk \
        gettext git java-propose-classpath libelf-dev libncurses5-dev \
        libncursesw5-dev libssl-dev python python2.7-dev python3 unzip wget \
        python3-distutils python3-setuptools python3-dev rsync subversion \
        swig time xsltproc zlib1g-dev
    ```

- Run “`make menuconfig`” to check whether all OpenWRT build pre-requisites are present on your machine
  - Install any missing packages with `sudo apt install <package_name>`

- Build the system without super user privileges:
  - Running as “root” user not recommended to compile and install the software

GETTING AND BUILDING OPENWRT
- OPENWRT DIRECTORY TREE OVERVIEW

- toolchain
  - At build time, the following two new directories are created
    - toolchain which is a temporary directory used for building the tool chain for a specific architecture
    - staging_dir where the resulting toolchain binaries are installed
  - There is no need to do anything with the toolchain directory unless a new version of one of these components is added
- target
  - target/linux/<platform/device_name> is platform-specific and contains the kernel .config file and kernel patches for the platform/device being used
  - target/linux/imagebuilder describes how to package a firmware for a specific platform
- package
  - Most of the firmware is packaged as .ipk modules (e.g. applications, drivers, libraries) which can be installed on a running system
  - This can provide new features or remove features to save space

GETTING AND BUILDING OPENWRT - OPENWRT DIRECTORY TREE OVERVIEW

- `bin:` contains the final binary images created during the build process
- `build_dir:` non toolchain source code and compiled images
- `configs:` configuration files for reference and development boards
- `dl:` local repository for downloaded OpenWrt source code packages
- `docs:` OpenWrt documentation
- `feeds:` additional predefined package build recipes for OpenWrt Buildroot
- `include:` Default core makefiles (*.mk) for OpenWrt (e.g. kernel, packages, fs)
- `packages:` OpenWrt Makefiles and source code patches
- `scripts:` scripts supporting the build makefiles
- `staging_dir:` compiled toolchain, including library includes, used to compile the rest of the distribution
- `tools:` necessary tools to build the image

GETTING AND BUILDING OPENWRT
- GETTING THE CODE

On an internet-connected Linux PC, typically a ubuntu 20.04 is good

```
andy@ubuntu:~$ git clone https://github.com/NXP/imx_openwrt.git
andy@ubuntu:~$ cd imx_openwrt
andy@ubuntu:~/imx_openwrt$ git branch -a
  * imx-openwrt-21.02.01
    remotes/origin/HEAD -> origin/imx-openwrt-21.02.01
    remotes/origin/imx-openwrt-21.02.01
    remotes/origin/master
```

GETTING AND BUILDING OPENWRT
- UPDATE OPENWRT FEEDS AND INSTALL LUCI WEB INTERFACE PACKAGE

- Update OpenWRT Feeds

```
andy@ubuntu:~/imx_openwrt$ ./scripts/feeds update –a
```

- Install single package (e.g. LuCI – GUI) package into OpenWRT build environment

```
andy@ubuntu:~/imx_openwrt$ ./scripts/feeds install luci
```

- Install all available packages into OpenWRT build environment

```
andy@ubuntu:~/imx_openwrt$ ./scripts/feeds install -a
```

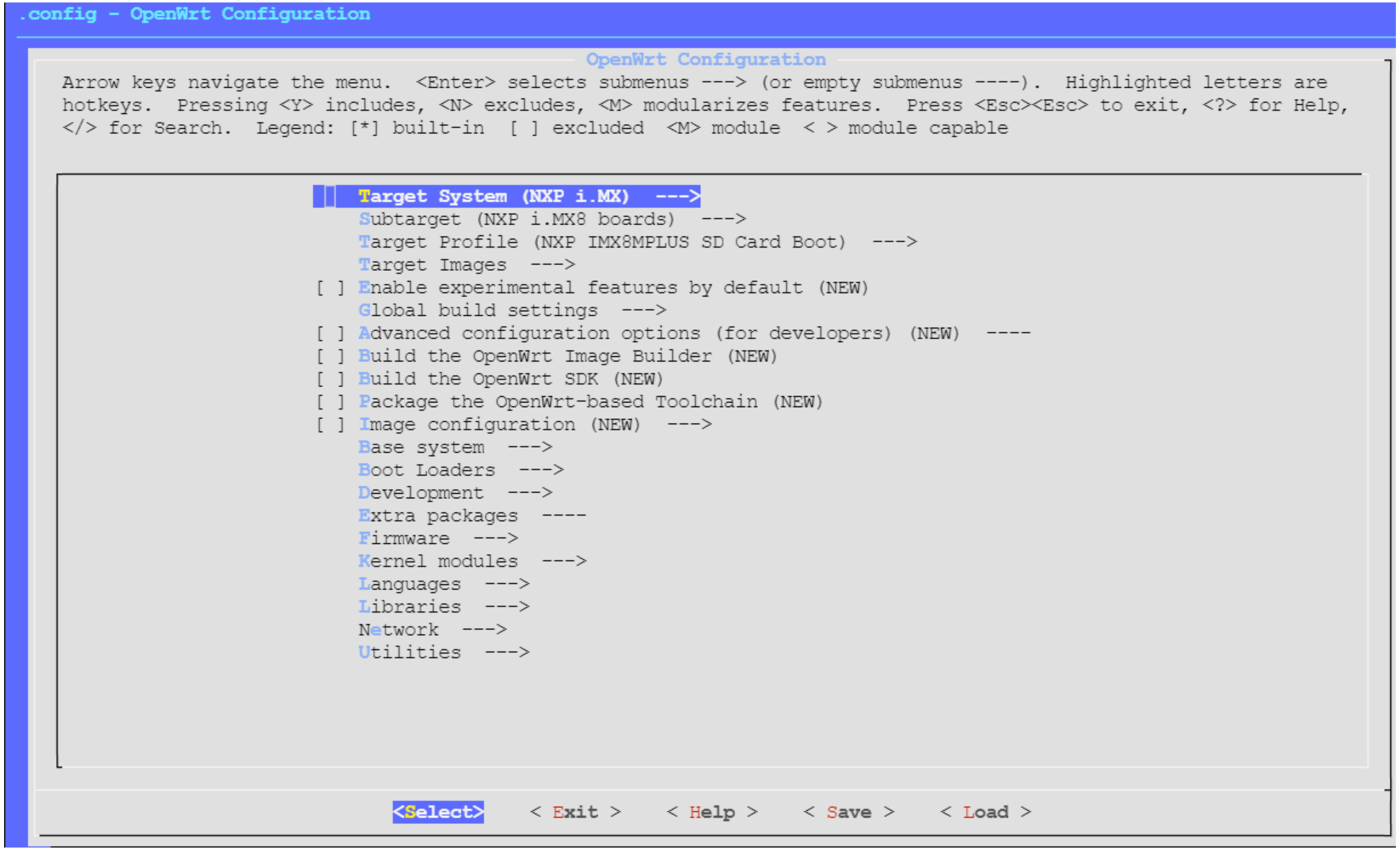
GETTING AND BUILDING OPENWRT
- CONFIGURING OPENWRT BUILD WITH MENUCONFIG

- Run menuconfig:

```
andy@ubuntu:~/imx_openwrt$ make menuconfig
```

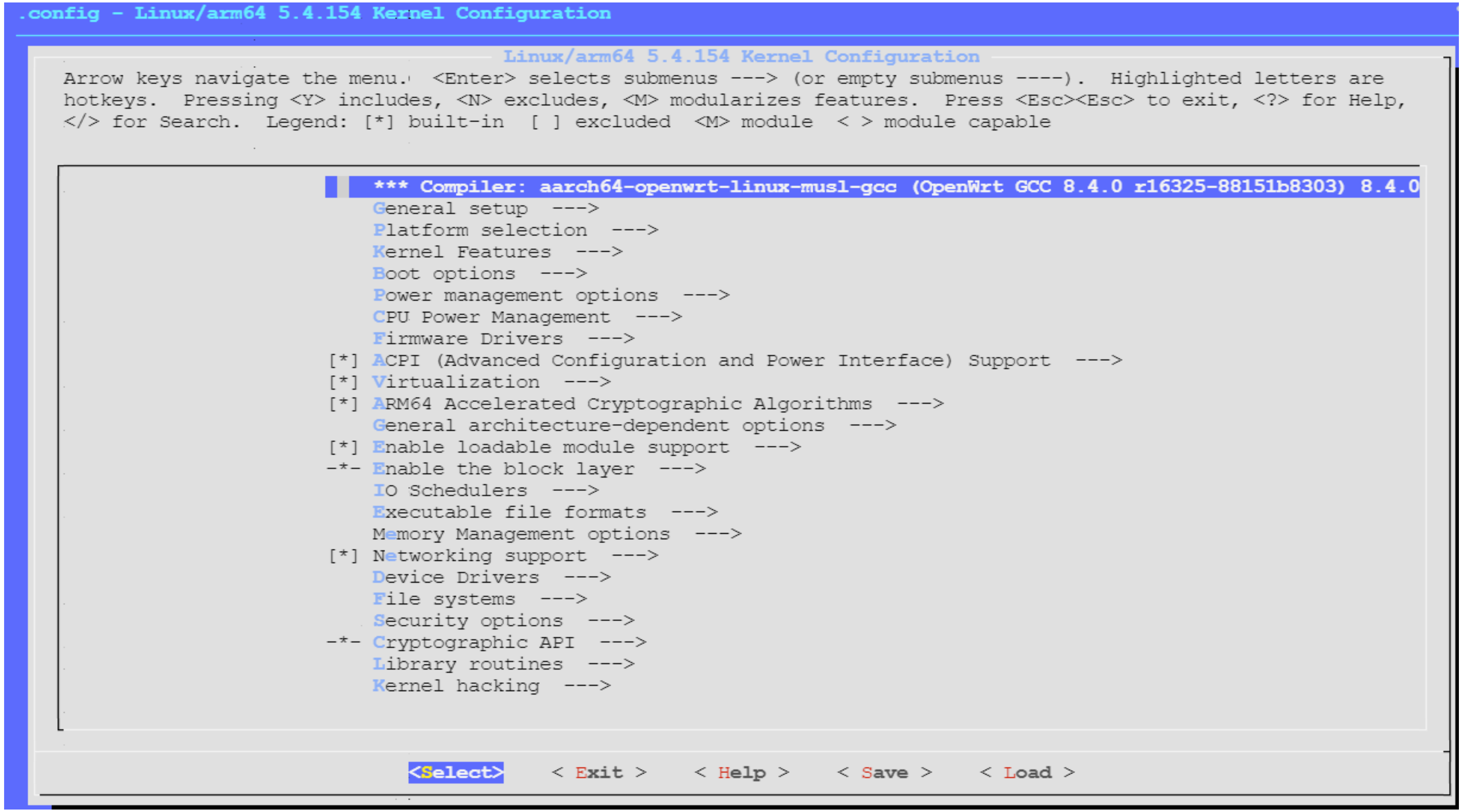
  - This performs a pre-requisite check
- Menu-driven build select and config
  - ‘Enter’ selects/enters a next-level menu
  - ‘Spacebar’ cycles a selection in square braces [] through ‘blank’, ‘M’ (sometimes), and ‘*’
    - Blank = not included/selected when the build is performed
    - M = included/selected and built, but not included in the final binary image
    - * = included/selected, built, and included in the final binary image
  - Cursor keys navigate the screen, along with options at bottom of the screen
- ‘/’ key calls up a search box
  - Handy if you know what you want to add, e.g. tcpdump, but don’t know where it lives in the menu system
- Run “`make kernel_menuconfig`” to adjust the kernel option if needed

GETTING AND BUILDING OPENWRT
- OPENWRT MENUCONFIG MAIN SCREEN



```
.config - OpenWrt Configuration
                                OpenWrt Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are
  hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help,
  </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                          Target System (NXP i.MX)  --->
                          Subtarget (NXP i.MX8 boards)  --->
                          Target Profile (NXP IMX8MPLUS SD Card Boot)  --->
                          Target Images  --->
                      [ ] Enable experimental features by default (NEW)
                          Global build settings  --->
                      [ ] Advanced configuration options (for developers) (NEW)  ----
                      [ ] Build the OpenWrt Image Builder (NEW)
                      [ ] Build the OpenWrt SDK (NEW)
                      [ ] Package the OpenWrt-based Toolchain (NEW)
                      [ ] Image configuration (NEW)  --->
                          Base system  --->
                          Boot Loaders  --->
                          Development  --->
                          Extra packages  ----
                          Firmware  --->
                          Kernel modules  --->
                          Languages  --->
                          Libraries  --->
                          Network  --->
                          Utilities  --->

                 <Select>    < Exit >    < Help >    < Save >    < Load >
```

GETTING AND BUILDING OPENWRT
- KERNEL MENUCONFIG MAIN SCREEN



```
.config - Linux/arm64 5.4.154 Kernel Configuration

                          Linux/arm64 5.4.154 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are
  hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help,
  </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                       *** Compiler: aarch64-openwrt-linux-musl-gcc (OpenWrt GCC 8.4.0 r16325-88151b8303) 8.4.0
                           General setup  --->
                           Platform selection  --->
                           Kernel Features  --->
                           Boot options  --->
                           Power management options  --->
                           CPU Power Management  --->
                           Firmware Drivers  --->
                       [*] ACPI (Advanced Configuration and Power Interface) Support  --->
                       [*] Virtualization  --->
                       [*] ARM64 Accelerated Cryptographic Algorithms  --->
                           General architecture-dependent options  --->
                       [*] Enable loadable module support  --->
                       -*- Enable the block layer  --->
                           IO Schedulers  --->
                           Executable file formats  --->
                           Memory Management options  --->
                       [*] Networking support  --->
                           Device Drivers  --->
                           File systems  --->
                           Security options  --->
                       -*- Cryptographic API  --->
                           Library routines  --->
                           Kernel hacking  --->

                     <Select>    < Exit >    < Help >    < Save >    < Load >
```

GETTING AND BUILDING OPENWRT
- CONFIGURE OPENWRT FOR i.MX 8M PLUS EVK

- Select the following:
  - Target System->`NXP i.MX`
  - Subtarget Profile->`NXP i.MX8 boards`
  - Target Profile->`NXP IMX8MPLUS SD Card Boot`
    - <Exit> to return to main Menuconfig menu
  - Network->`tcpdump`
  - Utilities->`sysstat`
    - If you have not done "./scripts/feeds install –a", you'll need to do a "./scripts/feeds install sysstat"
  - Base System->`busybox->Customize->Linux System Utilities->lspci and lsusb`
  - Check: Base System->`dnsmasq` is enabled
    - DHCP Server mainly
  - LuCI->`1. Collections -> luci`
    - Web Interface

GETTING AND BUILDING OPENWRT
- RE-CONFIGURE FOR ANOTHER BOARD

- If, having built for one i.MX 8M Plus, you want to build for a different board, you should start with a clean config. Do the following:
  - `make clean`
  - Suggest making a copy of .config in <BUILDROOT>/config
    - e.g. `cp .config config/config_imx8mp`
  - Delete the OpenWRT config file
    - `rm .config`
  - Run menuconfig as in previous steps to select the new platform and necessary config
  - Note: currently only imx8mplus_lpddr_evk board is supported. More board will be added soon.

GETTING AND BUILDING OPENWRT
- BUILD AND FLASH OPENWRT IMAGE

- Build the image:
  - ‘make’ or ‘make –j4’ to use 4 CPU cores
  - First time build takes some time and downloads various items
  - Needs around 12GB
- Resulting image:
  - `bin/targets/imx/imx8/openwrt-imx-imx8-imx8mplus-squashfs-sdcard.img`
  - This is a complete, monolithic, read/write SD boot image, including everything from uboot to squash-rootfs
  - Flash this to the SD card and boot it
    - Usage instructions, including how to flash the image to SD card are in the OpenWRT source tree:

```
andy@ubuntu:~/imx_openwrt$ cat target/linux/imx/README

andy@ubuntu:~/imx_openwrt$ sudo dd if=bin/targets/imx/imx8/openwrt-imx-imx8-imx8mplus-squashfs-sdcard.img
of=/dev/sdX bs=1M conv=fsync && sync
```

USING OPENWRT ON i.MX 8M PLUS EVK - BOOT IMAGE AND INSTALL PACKAGES IF NEEDED

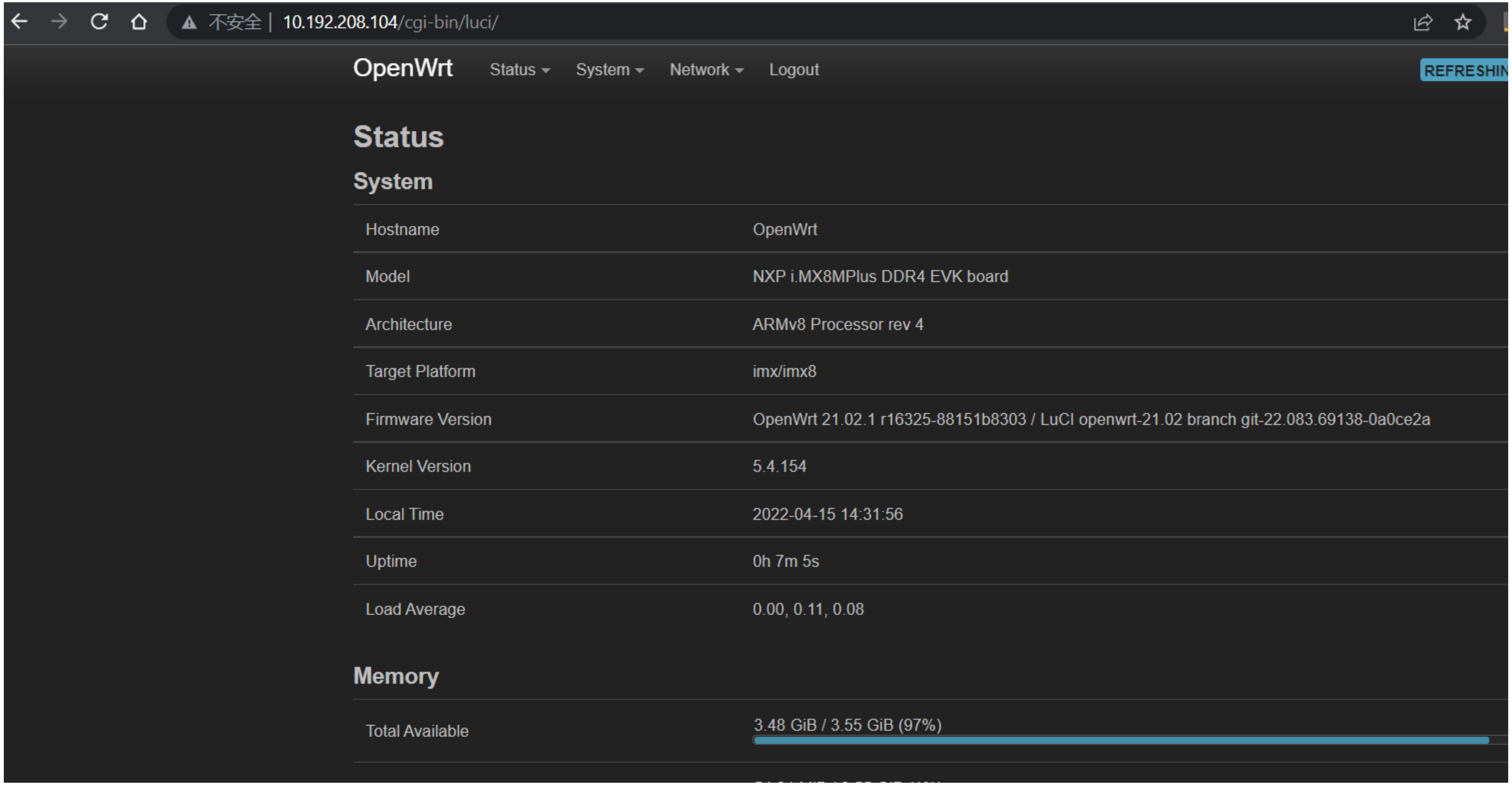
- Once booted, set a (root) password from the console, using ‘passwd’ command
  - This password is used for the root login via the GUI

```
root@OpenWrt:/# passwd
Changing password for root
New password:
Retype password:
passwd: password for root changed by root
```

  - Install luci web tool if you want to use LuCI GUI

```
root@OpenWrt:/# opkg update && opkg install luci
```

USING OPENWRT ON i.MX 8M PLUS EVK - CONFIGURE OPENWRT BY LUCI GUI



Status

System

Hostname	OpenWrt
Model	NXP i.MX8MPlus DDR4 EVK board
Architecture	ARMv8 Processor rev 4
Target Platform	imx/imx8
Firmware Version	OpenWrt 21.02.1 r16325-88151b8303 / LuCI openwrt-21.02 branch git-22.083.69138-0a0ce2a
Kernel Version	5.4.154
Local Time	2022-04-15 14:31:56
Uptime	0h 7m 5s
Load Average	0.00, 0.11, 0.08

Memory

Total Available	3.48 GiB / 3.55 GiB (97%)

USING OPENWRT ON i.MX 8M PLUS EVK
- ENABLE AND DEPLOY NEW OPENWRT PACKAGE

- Not all available items are enabled in default build, this is an embedded system…
- Plenty of Menuconfig items can simply be enabled, built, and installed on a running system
- Make process downloads new package from online repo if needed, confirms MD5 and builds it
- Example of sysstat if you haven't already installed it
  - Includes mpstat, sar, and other handy monitoring utils

```
$ make menuconfig
```

- Under 'Utilities' Menu, scroll down to 'systat', and hit <spacebar> until '<M>' is seen.
- Hit <ESC> and select <Yes> to save config.

```
$ make V=99 package/feeds/packages/sysstat/compile
```

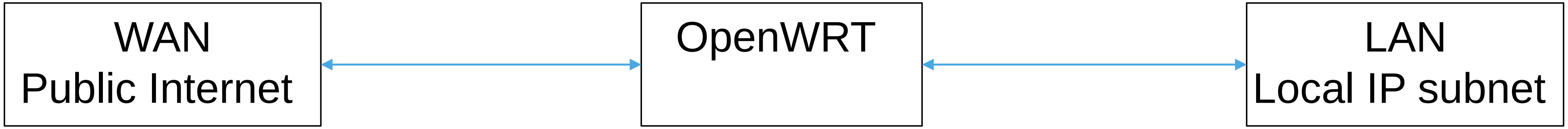
USING OPENWRT ON i.MX 8M PLUS EVK
- ENABLE AND DEPLOY NEW OPENWRT PACKAGE

- The resulting sysstat package `sysstat_11.6.0-2_aarch64_generic.ipk` is found in <BUILDROOT>/bin/packages/aarch64_generic/packages directory
- Take the new .ipk, transfer to board (e.g. scp, USB), and install:

```
opkg --install <package_name>
```

- It will tell you if you need to install a pre-requisite package first
  - In the case of sysstat, no additional pre-reqs

OPENWRT MAIN NETWORK FUNCTIONS



- NAPT Router between LAN and WAN Regions
  - Network Address and Port Translation – a Layer 4 function

- Stateful Packet Inspection (SPI) Firewall
  - Allows LAN->WAN connections to proceed, and matches associated incoming WAN->LAN connections
  - Other WAN->LAN connections are not allowed

- DHCP Server to LAN
  - Provides IP addresses to end-points on LAN
  - Typically all LAN interfaces are bridged together

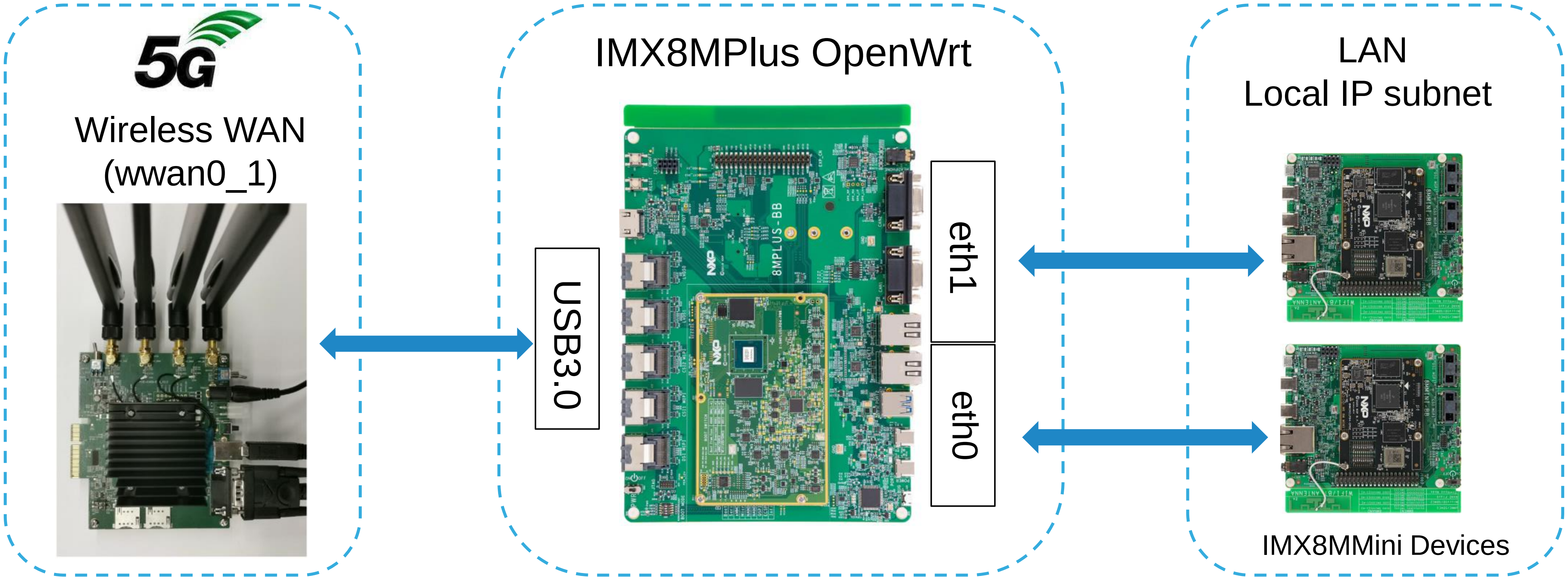
DEFAULT OPENWRT i.MX 8M PLUS NETWORK INTERFACE NAMES

Port Name	Interface Name in OpenWRT	Default OpenWRT Config	Demo config	
ENET1.RGMII	eth1	wan	lan	Bridge: br-lan
ENET.RGMII	eth0	lan	lan	
5G (USB or PCIe)	wwan0_1		wan	

Refer to the link for OpenWrt network configuration:

https://openwrt.org/docs/guide-user/base-system/basic-networking

5G INDUSTRIAL IOT APPLICATION TOPOLOGY



5G

Wireless WAN
(wwan0_1)

USB3.0

IMX8MPlus OpenWrt

eth1

eth0

LAN
Local IP subnet

IMX8MMini Devices

WAN interface is DHCP Client

LAN interfaces are bridged together. Static IP DHCP server at 192.168.1.1, serving LAN DHCP clients with addresses in 192.168.1.x subnet.

NETWORK CONFIGURATION FOR 5G IOT APPLICATION

Out of the Box Config

```
config globals 'globals'
        option ula_prefix 'fdea:973b:b559::/48’
```

```
config device
        option name 'br-lan'
        option type 'bridge'
        list ports 'eth0’

config interface 'lan'
        option device 'br-lan'
        option proto 'static'
        option ipaddr '192.168.1.1'
        option netmask '255.255.255.0'
        option ip6assign '60
```

```
config interface 'wan'
        option device ‘eth1'
        option proto ‘dhcp'
```

```
config interface 'wan6’
        option device ‘eth1’
        option proto 'dhcpv6'
```

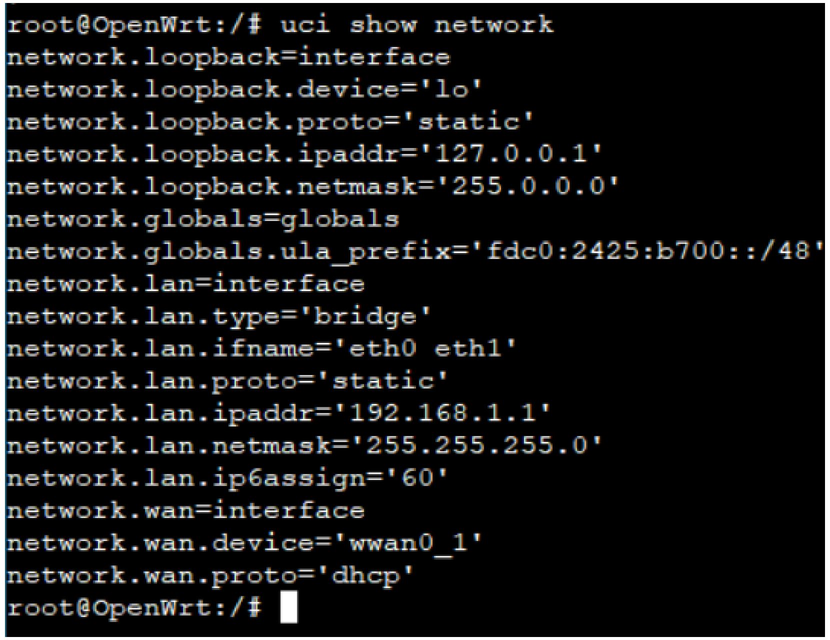
5G IoT Application Config

```
config interface 'lan'
        option type 'bridge'
        option ifname ‘eth0 eth1'
        option proto 'static'
        option ipaddr '192.168.1.1'
        option netmask '255.255.255.0'
        option ip6assign '60'
```

```
config interface 'wan'
        option ifname ‘wwan0_1'
        option proto ‘dhcp'
```

DHCP Client – WAN interface expects IP address to be provided by a DHCP Server on WAN

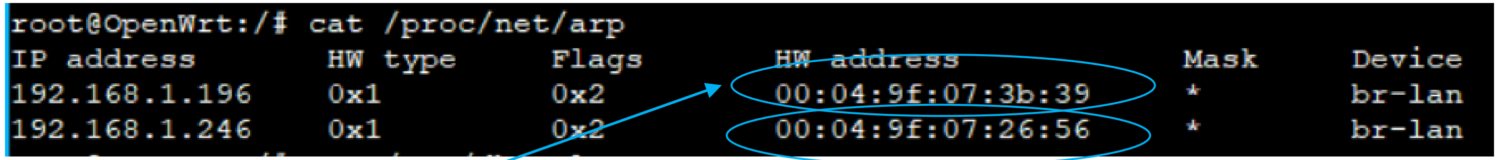
```
#config interface ‘wan6’
#       option ifname ‘wwan0_1’
#       option proto ‘dhcpv6'
```



```
root@OpenWrt:/# uci show network
network.loopback=interface
network.loopback.device='lo'
network.loopback.proto='static'
network.loopback.ipaddr='127.0.0.1'
network.loopback.netmask='255.0.0.0'
network.globals=globals
network.globals.ula_prefix='fdc0:2425:b700::/48'
network.lan=interface
network.lan.type='bridge'
network.lan.ifname='eth0 eth1'
network.lan.proto='static'
network.lan.ipaddr='192.168.1.1'
network.lan.netmask='255.255.255.0'
network.lan.ip6assign='60'
network.wan=interface
network.wan.device='wwan0_1'
network.wan.proto='dhcp'
root@OpenWrt:/#
```

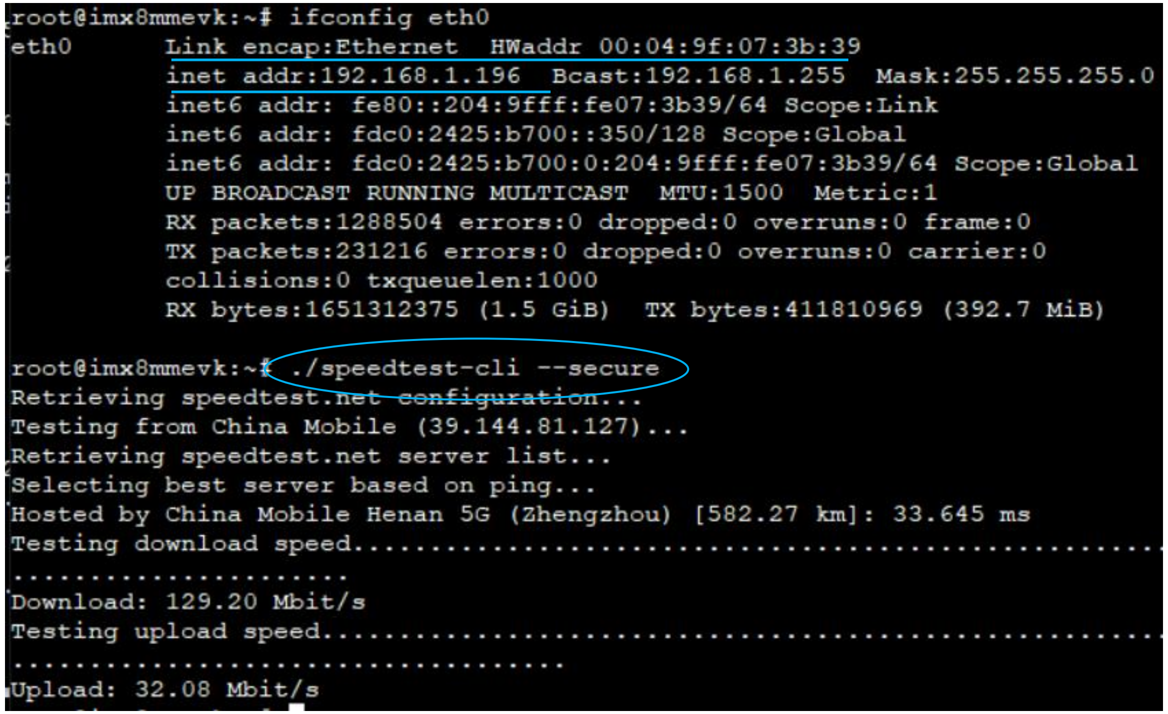
5G NETWORK SPEEDTEST

OpenWrt get device connected



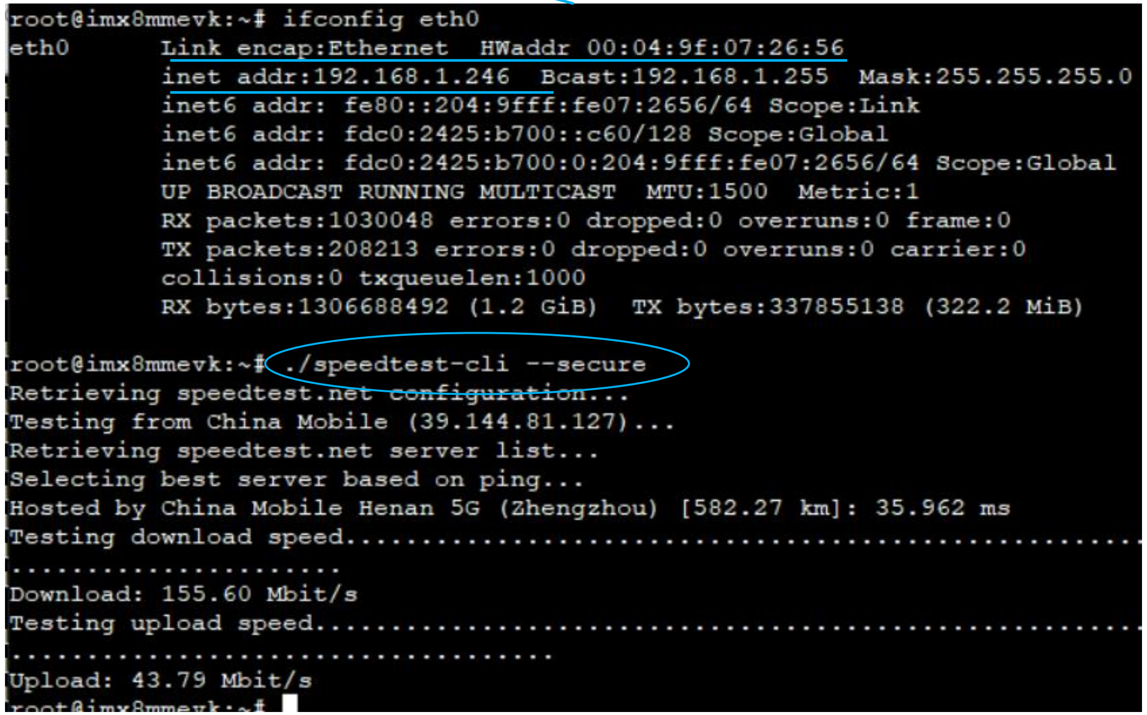
```
root@OpenWrt:/# cat /proc/net/arp
IP address       HW type     Flags       HW address            Mask     Device
192.168.1.196    0x1         0x2         00:04:9f:07:3b:39     *        br-lan
192.168.1.246    0x1         0x2         00:04:9f:07:26:56     *        br-lan
```

LAN device1 (IMX8MMini)



```
root@imx8mmevk:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:04:9f:07:3b:39
          inet addr:192.168.1.196  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::204:9fff:fe07:3b39/64 Scope:Link
          inet6 addr: fdc0:2425:b700::350/128 Scope:Global
          inet6 addr: fdc0:2425:b700:0:204:9fff:fe07:3b39/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1288504 errors:0 dropped:0 overruns:0 frame:0
          TX packets:231216 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1651312375 (1.5 GiB)  TX bytes:411810969 (392.7 MiB)

root@imx8mmevk:~# ./speedtest-cli --secure
Retrieving speedtest.net configuration...
Testing from China Mobile (39.144.81.127)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by China Mobile Henan 5G (Zhengzhou) [582.27 km]: 33.645 ms
Testing download speed................................................................................
......................
Download: 129.20 Mbit/s
Testing upload speed..................................................................................
....................................
Upload: 32.08 Mbit/s
```

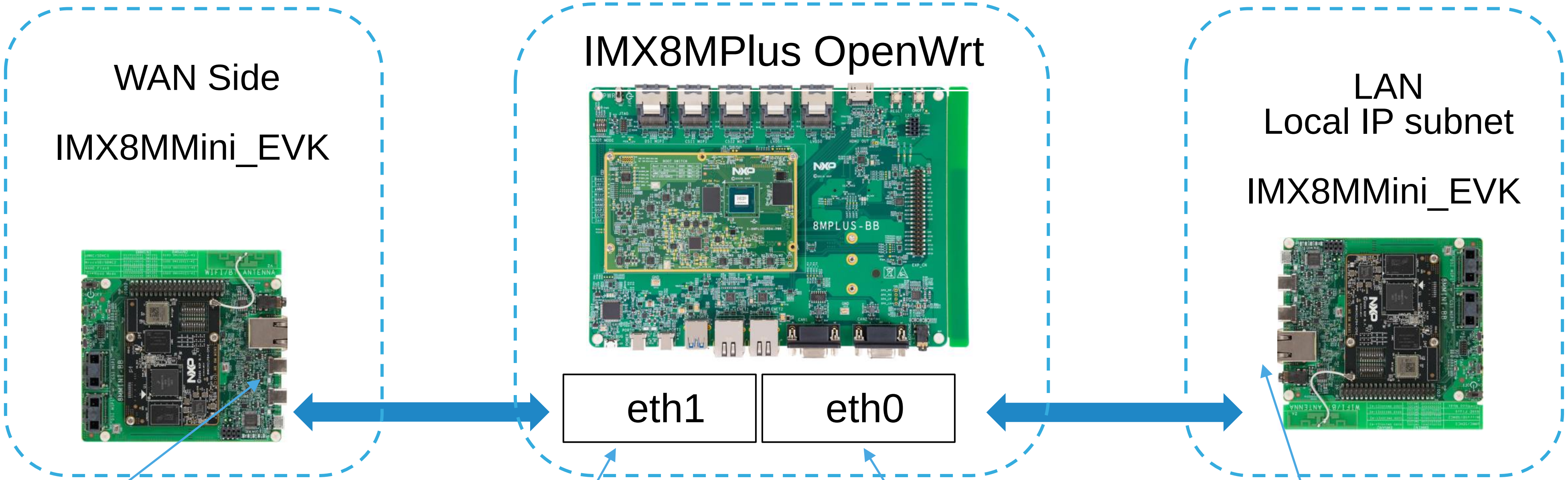
LAN device2 (IMX8MMini)



```
root@imx8mmevk:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:04:9f:07:26:56
          inet addr:192.168.1.246  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::204:9fff:fe07:2656/64 Scope:Link
          inet6 addr: fdc0:2425:b700::c60/128 Scope:Global
          inet6 addr: fdc0:2425:b700:0:204:9fff:fe07:2656/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1030048 errors:0 dropped:0 overruns:0 frame:0
          TX packets:208213 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1306688492 (1.2 GiB)  TX bytes:337855138 (322.2 MiB)

root@imx8mmevk:~# ./speedtest-cli --secure
Retrieving speedtest.net configuration...
Testing from China Mobile (39.144.81.127)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by China Mobile Henan 5G (Zhengzhou) [582.27 km]: 35.962 ms
Testing download speed................................................................................
......................
Download: 155.60 Mbit/s
Testing upload speed..................................................................................
....................................
Upload: 43.79 Mbit/s
root@imx8mmevk:~#
```

NOTE: The speed of the 5G network depends on the communication service provider, the test location and time period, etc.

ETHERNET BENCHMARK



By default, IMX8MMini_EVK doesn’t have WAN-side DHCP server, so need to use static IP addresses on the WAN side. Configure iMX8MMini_EVK_WAN with static IP of 10.207.203.169.

WAN interface has static IP of 10.207.203.172

LAN interface has static IP at 192.168.1.1, and serves LAN DHCP clients with addresses in 192.168.1.x subnet

Configure iMX8MMini_EVK_LAN to be DHCP client, IP is 192.168.1.<something>

```
root@imx8mmevk:~# iperf3 -s
```

From IMX8MPlus console, monitor CPU load using pidstat if you built it into the OpenWrt image, otherwise use top, or the GUI Load tab.

```
root@imx8mmevk:~# iperf3 -c 10.207.203.169

root@imx8mmevk:~# iperf3 -c 10.207.203.169 –t 120
```

NETWORK CONFIGURATION FOR BENCHMARK

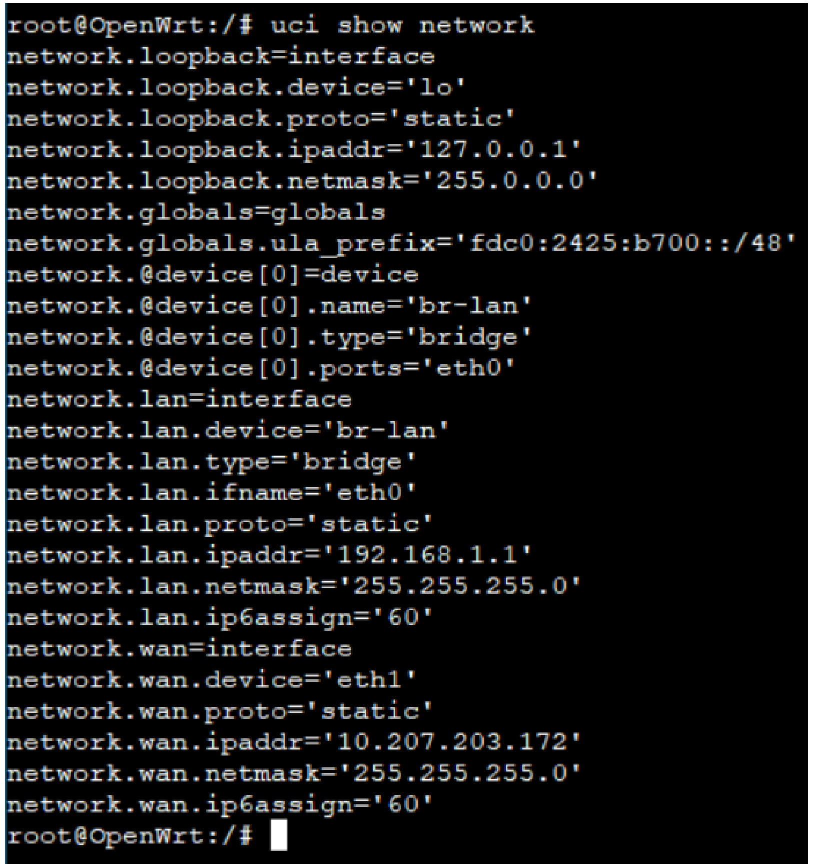
Out of the Box Config

```
config globals 'globals'
        option ula_prefix 'fdea:973b:b559::/48’

config device
        option name 'br-lan'
        option type 'bridge'
        list ports 'eth0’

config interface 'lan'
        option device 'br-lan'
        option proto 'static'
        option ipaddr '192.168.1.1'
        option netmask '255.255.255.0'
        option ip6assign '60

config interface 'wan'
        option device ‘eth1'
        option proto ‘dhcp'

config interface 'wan6’
        option device ‘eth1’
        option proto 'dhcpv6'
```

Benchmark Config

```
config globals 'globals'
        option ula_prefix 'fdea:973b:b559::/48’

config device
        option name 'br-lan'
        option type 'bridge'
        list ports 'eth0’

config interface 'lan'
        option device 'br-lan'
        option proto 'static'
        option ipaddr '192.168.1.1'
        option netmask '255.255.255.0'
        option ip6assign '60

config interface 'wan'
        option device ‘eth1'
        option proto ‘static’

#config interface 'wan6’
#       option device ‘eth1’
#       option proto 'dhcpv6'
```

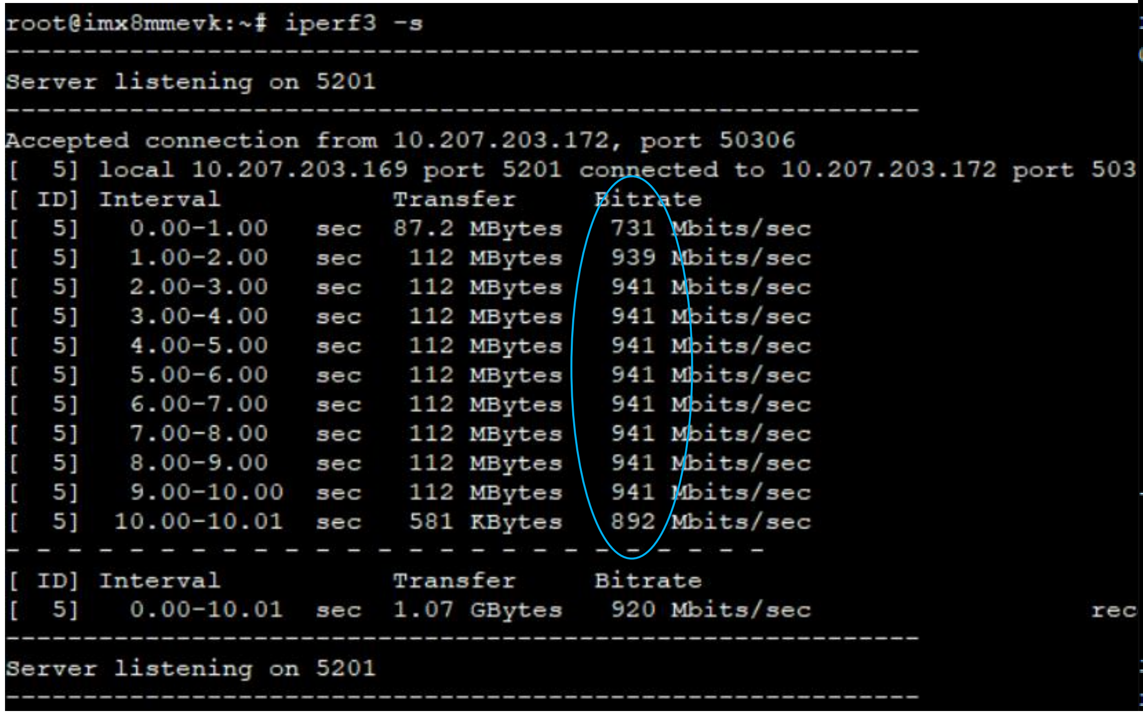
Static WAN IP – WAN interface has static IPv4 address defined here. No IPv6 address is defined/needed.



```
root@OpenWrt:/# uci show network
network.loopback=interface
network.loopback.device='lo'
network.loopback.proto='static'
network.loopback.ipaddr='127.0.0.1'
network.loopback.netmask='255.0.0.0'
network.globals=globals
network.globals.ula_prefix='fdc0:2425:b700::/48'
network.@device[0]=device
network.@device[0].name='br-lan'
network.@device[0].type='bridge'
network.@device[0].ports='eth0'
network.lan=interface
network.lan.device='br-lan'
network.lan.type='bridge'
network.lan.ifname='eth0'
network.lan.proto='static'
network.lan.ipaddr='192.168.1.1'
network.lan.netmask='255.255.255.0'
network.lan.ip6assign='60'
network.wan=interface
network.wan.device='eth1'
network.wan.proto='static'
network.wan.ipaddr='10.207.203.172'
network.wan.netmask='255.255.255.0'
network.wan.ip6assign='60'
root@OpenWrt:/# 
```

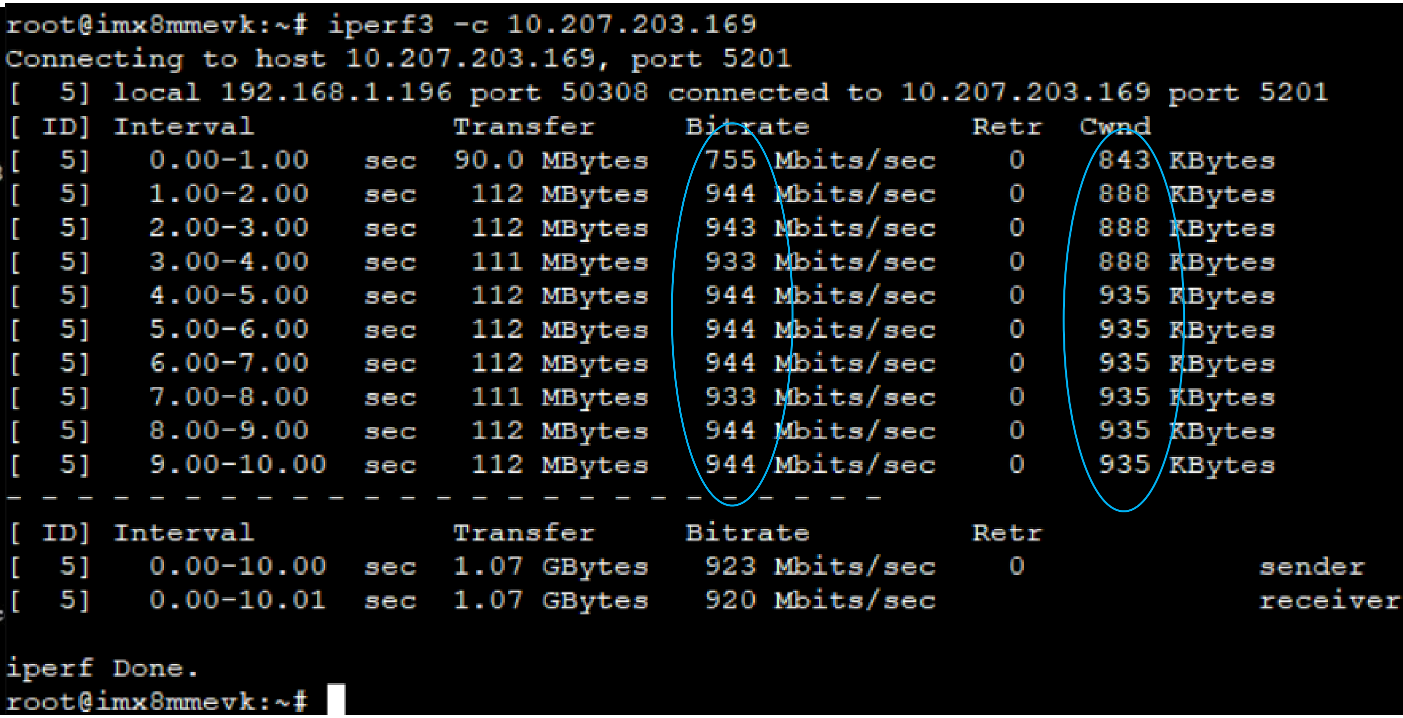
BASIC IPERF3 TEST

IMX8MMini WAN



```
root@imx8mmevk:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 10.207.203.172, port 50306
[  5] local 10.207.203.169 port 5201 connected to 10.207.203.172 port 503
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec  87.2 MBytes   731 Mbits/sec
[  5]   1.00-2.00   sec   112 MBytes   939 Mbits/sec
[  5]   2.00-3.00   sec   112 MBytes   941 Mbits/sec
[  5]   3.00-4.00   sec   112 MBytes   941 Mbits/sec
[  5]   4.00-5.00   sec   112 MBytes   941 Mbits/sec
[  5]   5.00-6.00   sec   112 MBytes   941 Mbits/sec
[  5]   6.00-7.00   sec   112 MBytes   941 Mbits/sec
[  5]   7.00-8.00   sec   112 MBytes   941 Mbits/sec
[  5]   8.00-9.00   sec   112 MBytes   941 Mbits/sec
[  5]   9.00-10.00  sec   112 MBytes   941 Mbits/sec
[  5]  10.00-10.01  sec   581 KBytes   892 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-10.01  sec  1.07 GBytes   920 Mbits/sec                  rec
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

IMX8MMini LAN



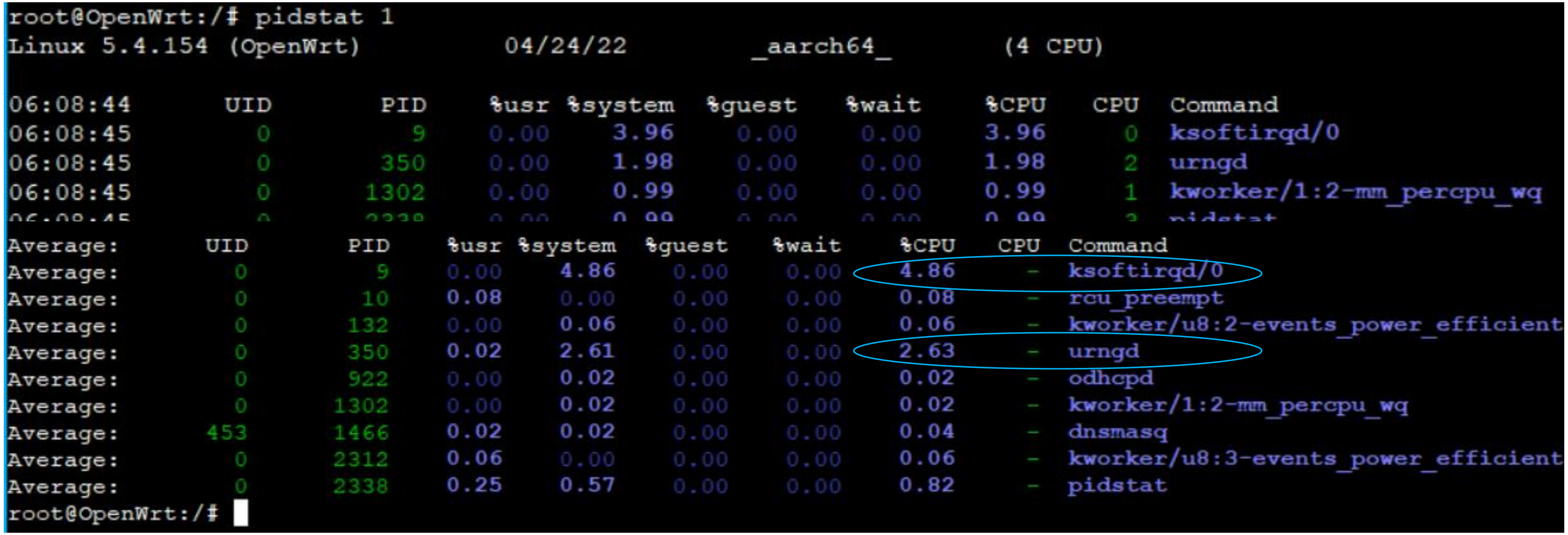
```
root@imx8mmevk:~# iperf3 -c 10.207.203.169
Connecting to host 10.207.203.169, port 5201
[  5] local 192.168.1.196 port 50308 connected to 10.207.203.169 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  90.0 MBytes   755 Mbits/sec    0    843 KBytes
[  5]   1.00-2.00   sec   112 MBytes   944 Mbits/sec    0    888 KBytes
[  5]   2.00-3.00   sec   112 MBytes   943 Mbits/sec    0    888 KBytes
[  5]   3.00-4.00   sec   111 MBytes   933 Mbits/sec    0    888 KBytes
[  5]   4.00-5.00   sec   112 MBytes   944 Mbits/sec    0    935 KBytes
[  5]   5.00-6.00   sec   112 MBytes   944 Mbits/sec    0    935 KBytes
[  5]   6.00-7.00   sec   112 MBytes   944 Mbits/sec    0    935 KBytes
[  5]   7.00-8.00   sec   111 MBytes   933 Mbits/sec    0    935 KBytes
[  5]   8.00-9.00   sec   112 MBytes   944 Mbits/sec    0    935 KBytes
[  5]   9.00-10.00  sec   112 MBytes   944 Mbits/sec    0    935 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  1.07 GBytes   923 Mbits/sec    0             sender
[  5]   0.00-10.01  sec  1.07 GBytes   920 Mbits/sec                  receiver

iperf Done.
root@imx8mmevk:~#
```

Sender Bitrate: Maximum/944 Mbits/sec; Average/923 Mbits/sec

Receiver Bitrate: Maximum/941 Mbits/sec; Average/920 Mbits/sec

Cwnd Range: 843 Kbytes ~ 935 Kbytes

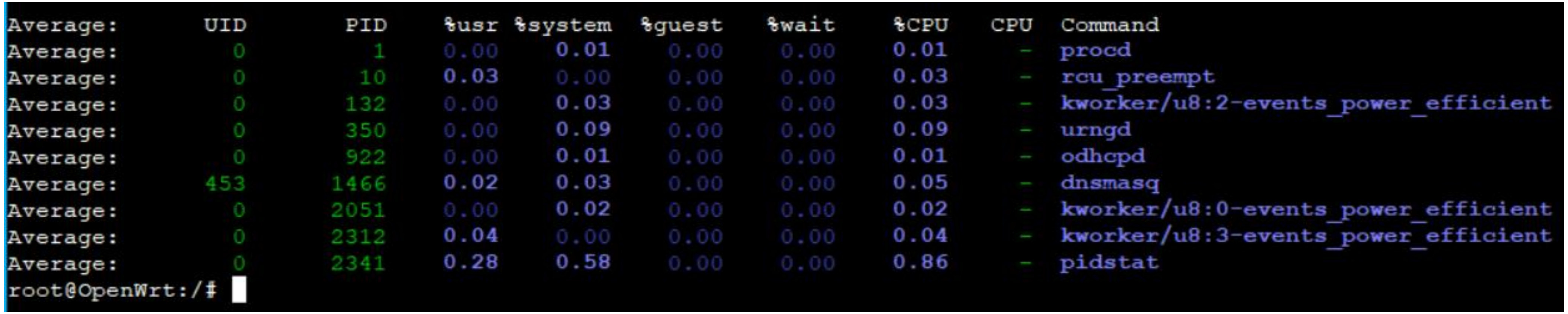
IMX8MPLUS_OPENWRT CPU LOAD DURING IPERF3 TEST



```
root@OpenWrt:/# pidstat 1
Linux 5.4.154 (OpenWrt)         04/24/22        _aarch64_       (4 CPU)

06:08:44        UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
06:08:45          0         9    0.00    3.96    0.00    0.00    3.96     0  ksoftirqd/0
06:08:45          0       350    0.00    1.98    0.00    0.00    1.98     2  urngd
06:08:45          0      1302    0.00    0.99    0.00    0.00    0.99     1  kworker/1:2-mm_percpu_wq
06:08:45          0      2338    0.00    0.99    0.00    0.00    0.99     3  pidstat
Average:        UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
Average:          0         9    0.00    4.86    0.00    0.00    4.86     -  ksoftirqd/0
Average:          0        10    0.08    0.00    0.00    0.00    0.08     -  rcu_preempt
Average:          0       132    0.00    0.06    0.00    0.00    0.06     -  kworker/u8:2-events_power_efficient
Average:          0       350    0.02    2.61    0.00    0.00    2.63     -  urngd
Average:          0       922    0.00    0.02    0.00    0.00    0.02     -  odhcpd
Average:          0      1302    0.00    0.02    0.00    0.00    0.02     -  kworker/1:2-mm_percpu_wq
Average:        453      1466    0.02    0.02    0.00    0.00    0.04     -  dnsmasq
Average:          0      2312    0.06    0.00    0.00    0.00    0.06     -  kworker/u8:3-events_power_efficient
Average:          0      2338    0.25    0.57    0.00    0.00    0.82     -  pidstat
root@OpenWrt:/# 
```

CPU load

in iperf3 test



```
Average:        UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
Average:          0         1    0.00    0.01    0.00    0.00    0.01     -  procd
Average:          0        10    0.03    0.00    0.00    0.00    0.03     -  rcu_preempt
Average:          0       132    0.00    0.03    0.00    0.00    0.03     -  kworker/u8:2-events_power_efficient
Average:          0       350    0.00    0.09    0.00    0.00    0.09     -  urngd
Average:          0       922    0.00    0.01    0.00    0.00    0.01     -  odhcpd
Average:        453      1466    0.02    0.03    0.00    0.00    0.05     -  dnsmasq
Average:          0      2051    0.00    0.02    0.00    0.00    0.02     -  kworker/u8:0-events_power_efficient
Average:          0      2312    0.04    0.00    0.00    0.00    0.04     -  kworker/u8:3-events_power_efficient
Average:          0      2341    0.28    0.58    0.00    0.00    0.86     -  pidstat
root@OpenWrt:/# 
```

CPU load
in idle

During iperf3 test, Average CPU load on IMX8MPlus_OpenWrt < 5%

i.MX 8M Network Benchmarks

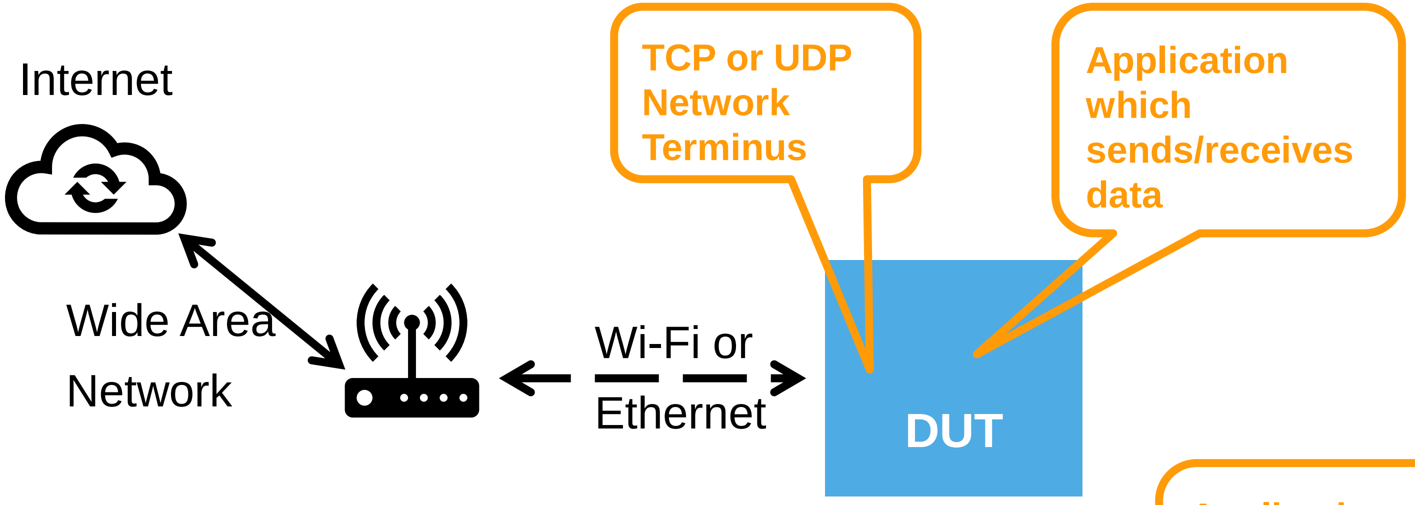
SECURE CONNECTIONS
FOR A SMARTER WORLD

PUBLIC

NETWORKING USE-CASES

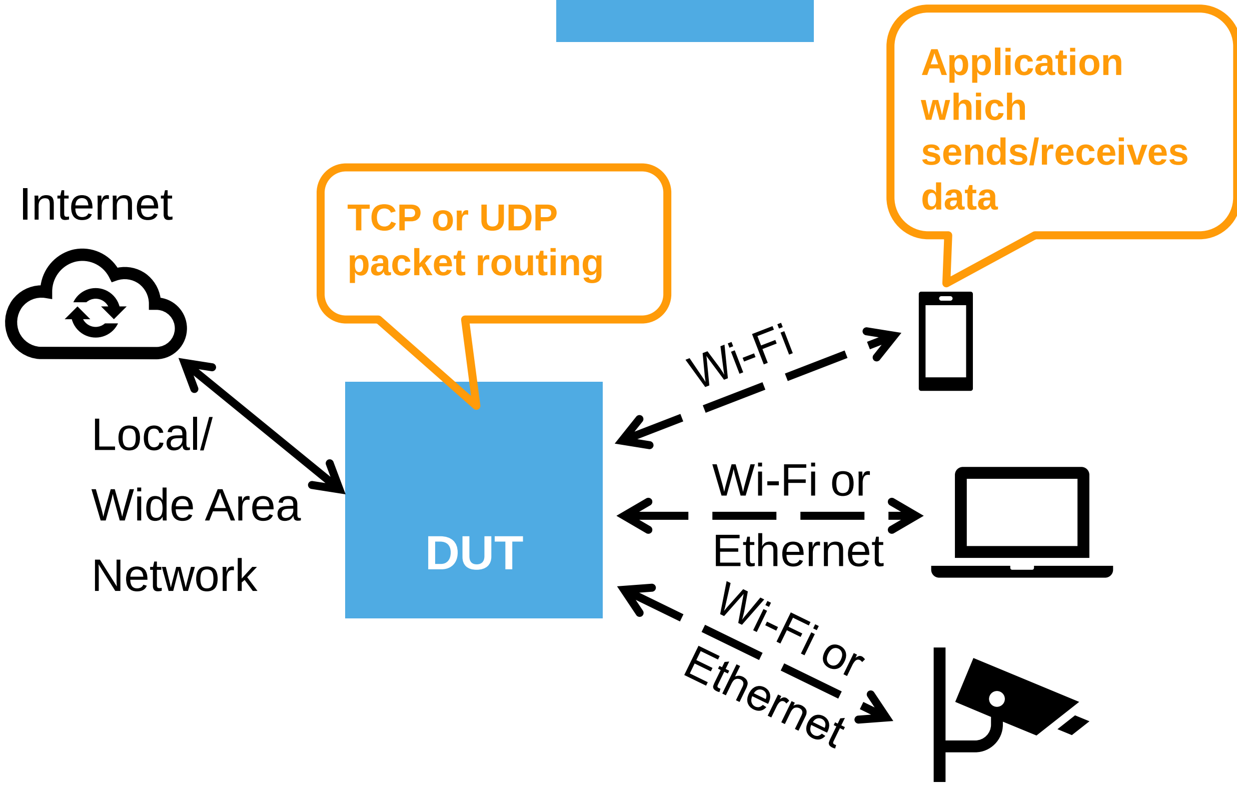
1. Network Termination

- TCP or UDP handshaking
- Reciving/Sending packets
- Acknowledging, resending, discarding, etc.
- Aggregating/disaggregating data
- Packets consumed/originated by Device-Under-Test


Internet
Wide Area
Network
Wi-Fi or
Ethernet
DUT
TCP or UDP Network Terminus
Application which sends/receives data

2. Network Routing

- TCP or UDP routing
- DUT examines packet headers and forwards packets to the appropriate route
- DUT may perform additional functions e.g. Firewall, IPSEC encapsulation, Deep Packet Inspection
- Most packets pass through DUT



KEY TAKE-AWAYS FROM i.MX 8M NETWORKING BENCHMARKS

- For **network termination**, i.MX 8M has good performance, and there is no significant difference in results compared to LS1043A
  - This is the most common use-case for i.MX – where it is the terminus of the network

- For **network routing**, i.MX 8M can support ~500Mbps for mixed packet sizes (IMIX), or ~1.5Gbps for large packets with standard Linux networking stack
  - Demo implementation of DPDK on i.MX 8M Mini shows 2-4x improvement
    - Other options such as XDP software fastpath could also be explored
  - LS1043A is significantly faster for routing use-cases, (up to 10Gbps with DPDK)

DUT SYSTEM DESCRIPTION :

Board	CPUs	Maximum CPU Frequency	L2 Cache Size	Memory size	Peak Raw DDR bandwidth
i.Mx8MQ EVK (Rev 1.0)	4x Cortex-A53	1.5GHz	1MB	DDR4 DRAM- 3GB	12.8GB/s
i.Mx8M Plus (Rev 1.0)	4x Cortex-A53	1.8GHz	512KB	DRAM- 6GB	16GB/s
i.Mx8M Mini EVK (Rev 1.0)	4x Cortex-A53	1.8GHz	512KB	LPDDR4 DRAM- 2GB	12GB/s
i.Mx8M Mini EVK (Rev 1.0)	4x Cortex-A53	1.8GHz	512KB	DDR4 DRAM- 2GB	9.6GB/s
i.Mx8M Nano	4x Cortex-A53	1.5GHZ	512KB	LPDDR4 DRAM- 2GB	6.4GB/s
LS1043ARDB (Rev 1.1)	4x Cortex-A53	1.6GHz	1MB	DDR4 SDRAM- 2GB	6.4GB/s

TCP TERMINATION TEST RESULTS

- All i.MX 8M achieved line rate with a single A53 core (940Mbps payload on 1G unidirectional connection)
- Some small variations in CPU loads due to CPU frequency, cache, memory bandwidth
- Results are similar to Layerscape LS1043A
- This is the most common use-case for i.MX devices – the terminus of a network connection

	Single core CPU Utilization @ 940Mbps, large packets
i.MX 8M Plus	50%
i.MX 8M Quad	36%
i.MX 8M Mini	41%
i.MX 8M Nano	46%
LS1043A	44%

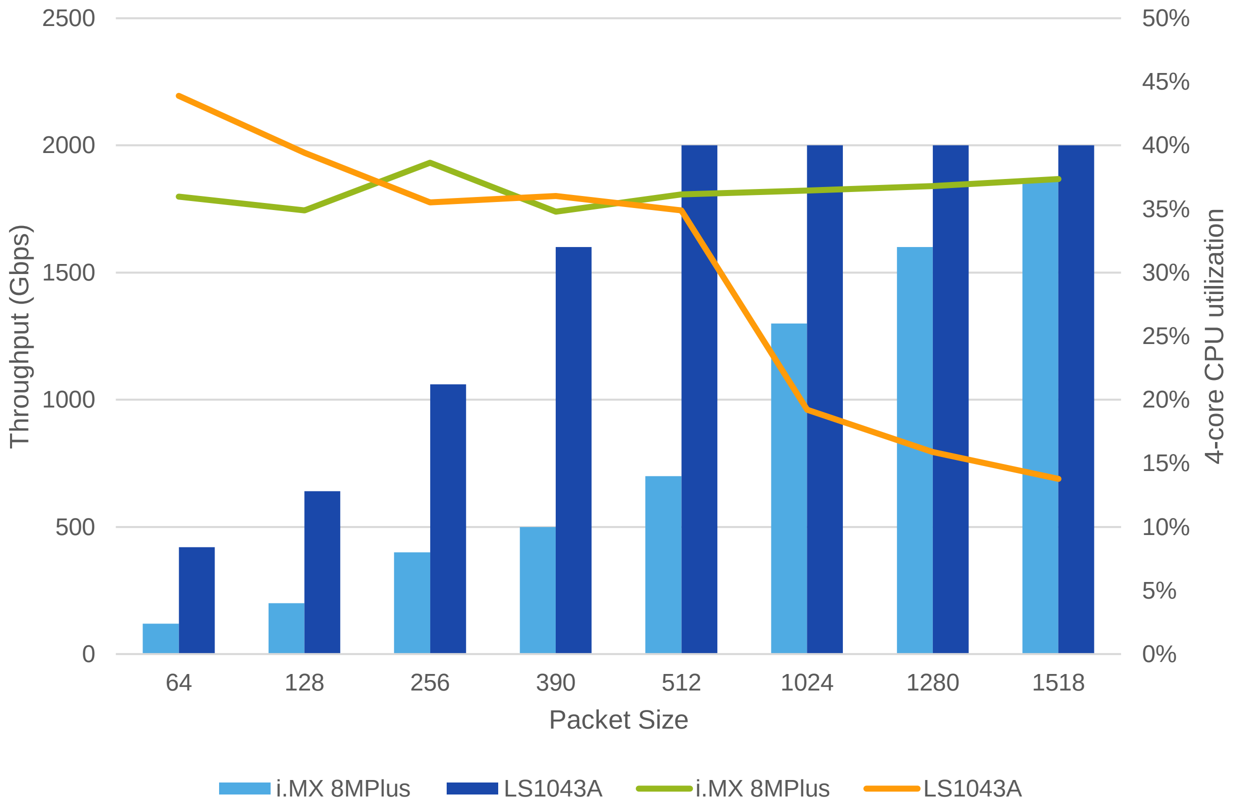
ETHERNET<->ETHERNET L3FWD RESULTS FOR LINUX NETWORKING STACK ON i.MX 8MPLUS, COMPARISON WITH LS1043A

- i.MX 8M Plus achieves ~500Mbps for 390B packet sizes (equivalent to IMIX)
- LS1043A delivers superior throughput and CPU utilization due to:
  - Hardware acceleration of buffering/queueing
  - Packet interrupt distribution across all cores
  - I/O coherent interconnect

Notes

- No significant difference was observed when testing with multiple flows
- Linux networking stack is not optimized for throughput, contains numerous memcopies, etc.
- i.MX 8M Plus is the only Mscale part with more than one ethernet port.

L3fwd Eth<->Eth throughput comparison, 2 flows & 4 cores



5G INDUSTRIAL IOT GATEWAY SOLUTION

Brand	LS1028A	LS1046A	LS1043A	IMX8Mplus	IMX8Mmini	IMX8MQ
Quectel(QCA)	RM500Q-CN/RM502Q-AE	RM500Q-CN/RM502Q-AE		RM500Q-CN/RM502Q-AE	RM500Q-CN/RM502Q-AE	
Quectel(UniSoC)	RG200U-CN		RG200U-CN			
Quectel(MTK)						
Fibocom(QCA)	FM150-AE-01	FM150-AE-01	FM150-AE-01			
Fibocom(MTK)						
Gosuncn(QCA)	GM800	GM800				
MeigSmart(QCA)						

- Official site: https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/layerscape-processors/layerscape-1046a-and-1026a-processors:LS1046A?tab=Documentation_Tab
- Community: https://community.nxp.com/t5/Layerscape/Enabling-5G-Module-on-Layerscape-Platforms/td-p/1372545

SUMMARY

- OpenWRT provides an out-of-the-box Residential Gateway for demo, eval, and more
- Nice-looking GUI, good networking performance
- Simple to add new features via OpenWRT menuconfig
- Single generated image – easy to deploy
- Wi-Fi/5G module can be configured and deployed easily

TECHNOLOGY SHOWROOM

JOURNEYS BY DESIRED ENGAGEMENT

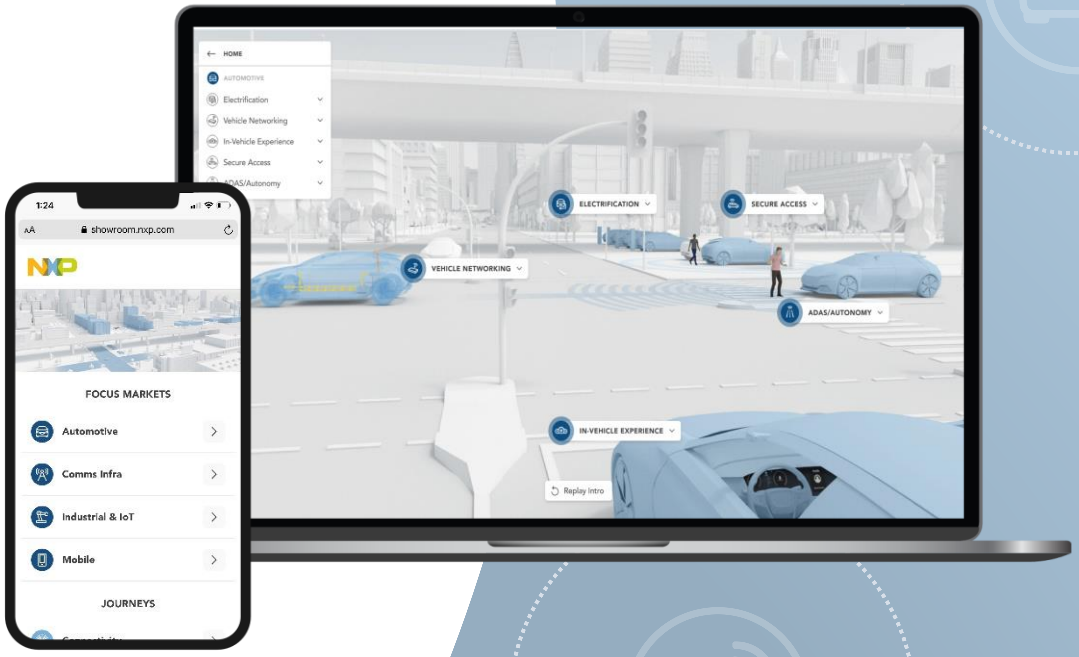
Self-guided tour
Live-streaming at set times
Guided tours

40+ VIRTUAL DEMOS

Focus on system solutions
Set up along NXP verticals

JOURNEYS BY DESIRED FOCUS

Edge & AI/ML
Safety & Security
Connectivity
Analog



SHOWROOM.NXP.COM

WELCOME TO FOLLOW NXP AT SOCIAL PLATFORMS

欢迎您关注「恩智浦微招聘」公众号
及时获取恩智浦“芯”职位及员工
活动相关资讯

关注NXP客栈公众号，查看恩
智浦最新官方资讯及技术材料

关注恩智浦B站官方账号，观
看恩智浦最新技术视频

Q&A

SECURE CONNECTIONS
FOR A SMARTER WORLD

SHOWROOM.NXP.COM