Yaskawa solectria

Scalability testing for solectria mesh network

testbed description & results

Scalability testing for solectria mesh network

Nov 02, 2016

Test-bed description & RESULTS

# Objective

To analyze the effective improvement in channel utilization offered by the air-time fairness metric kernel patch in alleviating the buffer bloat problem

# Rationale

The target number of nodes to be added to the Solectria Mesh network is 50. However, with the existing mesh network standard, only 32 nodes can be supported with reliable connectivity. When the number of nodes in the mesh network is increased beyond 32, the network experiences dramatic drop in throughput and rise in network latency bringing the network down. This is owing to the increase in the back-off time of individual nodes before the nodes could access the channel resulting in the cumulative network throughput being very less than the network capacity. Consequently, the transmit buffers in the node grow to accumulate the ‘to-be-transmitted’ data, leading to a condition called buffer bloat.

The main cause of buffer-bloat is the variation in transmission rate among the nodes. Currently, when a node gains access to the channel using CSMA technique, it is allowed to transmit a unit volume of data irrespective of its transmission rate. The slower nodes tend to occupy the channel longer, forcing the other nodes to wait and bloat their buffers. The buffer bloat problem can be alleviated by the provision of an ‘Air-Time Fairness Metric (AFM)’ that considers the individual transmission duration thereby accounting the actual channel utilization. AFM patch is available as ath9k-Add-a-per-station-airtime-deficit-scheduler.patch, which is used here to compare the mesh network performance with and without the airtime scheduler

# Test tool

The Realtime Response Under Load (RRUL) test puts a network under worst case conditions, and then measures for fairness, latency, realtime responsiveness, and the relative performance of TCP and UDP streams of varying rates, while under that load. For this test, RRUL test offered by FLENT will be used to study the ICMP Ping latency in milliseconds, and TCP upload and download throughputs in Mbps. The script for installing FLENT is given in appendix-B.

# Test bed for 4 nodes

The 4 node mesh network is configured to have one server (S: 10.15.140.239 – Vericite board) and 3 clients:

C1: 10.15.141.74 (Sabre board)

C2: 10.15.141.82 (Vericite board)

C3: 10.15.141.97 (Sabre board)

The node preparation steps are furnished in Appendix-A.

RRUL tests are conducted to record the node behavior for the following cases:

Case 1: Server with 1 client (2 nodes)

Case 2: Server with 2 clients – synchronized and transmitting in parallel (3 nodes)

Case 3: Server with 3 clients – synchronized and transmitting in parallel (4 nodes)

2 scenarios were considered for the experiment: In the first, nodes C1 and C2 are placed closer to S (as shown in Figure.1), while C3 was separated by around 35 feet from S. In the second scenario, C1 is moved away from S and placed closer to C3.

S

C2

C1 (Scenario 1)

C1 (Scenario 2)

C3

# test procedure

Initially, the kernel image that is not patched with AFM is used to observe the latency and throughput. RRUL is run in server mode at S and in client mode at C1, C2 and C3. The observations are made in the 4 node mesh network, by running the clients one by one, then in pairs and finally all three clients in parallel.

After the initial set of observations without the AFM patch, the kernel image is modified in accordance with the AFM patch and the new images are loaded in the respective nodes. The same set of observations is made to compare with those obtained before applying the AFM patch.

## scheduling

In order to perform the test with varying network load, the nodes are scheduled to transmit individually and in parallel with other nodes. For parallel transmissions, to synchronize the nodes in such a way to begin and end transmission simultaneously, they are remotely scheduled using ssh and crontab. The clock in each node is verified for global synchronization with the FLENT server (in this case 10.15.140.239) using the date –set command.

# Results

Latency Analysis:

Without AFM patch

Figure:1 Latency (in ms) with number of nodes with C1 closer to S (left) and C1 away from S

With AFM patch

Throughput Analysis:

Without AFM patch

With AFM patch

# Conclusions

# Appendix-A

# kernel patching for air-time fairness

# Steps followed in developing the kernel:

1. Get the linux tar ball – current version used is Linux-4.7.5

2. untar in home

3. Patch the kernel with patches:

a. 0001-kernel-4.7.x-imx6-2016-09-21.patch (from SVN trunk/kernel)

b. 0007-ath9k-Add-a-per-station-airtime-deficit-scheduler.patch (airtime fairness patch)

4. Make config file:

sudo CROSS\_COMPILE=arm-linux-gnueabihf- ARCH=arm INSTALL\_PATH=/tmp INSTALL\_MOD\_PATH=/tmp INSTALL\_FW\_PATH=/tmp make solectria\_defconfig

(/home/ubu14/yellowstone-kernel-tmp/linux-4.7.5/arch/arm/configs/solectria\_defconfig)

5. Clean before compilation:

sudo CROSS\_COMPILE=arm-linux-gnueabihf- ARCH=arm INSTALL\_PATH=/tmp INSTALL\_MOD\_PATH=/tmp INSTALL\_FW\_PATH=/tmp make clean

6. Compile:

sudo CROSS\_COMPILE=arm-linux-gnueabihf- ARCH=arm INSTALL\_PATH=/tmp INSTALL\_MOD\_PATH=/tmp INSTALL\_FW\_PATH=/tmp make zImage modules dtbs

7. Install modules:

sudo CROSS\_COMPILE=arm-linux-gnueabihf- ARCH=arm INSTALL\_PATH=/tmp INSTALL\_MOD\_PATH=/tmp INSTALL\_FW\_PATH=/tmp make modules\_install

8. Cryptodev patch:

cd cryptodev-linux-1.8/

patch -p1 --dry-run -i ~/0001-cryptodev-1.8\_linux-4.7.x\_2016-06-27.patch

9. sudo CROSS\_COMPILE=arm-linux-gnueabihf- ARCH=arm INSTALL\_PATH=/tmp INSTALL\_MOD\_PATH=/tmp INSTALL\_FW\_PATH=/tmp KERNEL\_DIR=~/linux-4.7.5 make clean

11. sudo CROSS\_COMPILE=arm-linux-gnueabihf- ARCH=arm INSTALL\_PATH=/tmp INSTALL\_MOD\_PATH=/tmp INSTALL\_FW\_PATH=/tmp KERNEL\_DIR=/home/ubu14/linux-4.7.5 make

12. sudo CROSS\_COMPILE=arm-linux-gnueabihf- ARCH=arm INSTALL\_PATH=/tmp INSTALL\_MOD\_PATH=/tmp INSTALL\_FW\_PATH=/tmp KERNEL\_DIR=/home/ubu14/linux-4.7.5 make install

13. Loading the required files to SD card:

sudo cp ~/linux-4.7.5/arch/arm/boot/zImage /mnt/rootfs/boot

sudo cp ~/linux-4.7.5/arch/arm/boot/dts/imx6\*var\*.dtb /mnt/rootfs/boot/dtbs/

sudo mkdir /mnt/rootfs/lib/modules/4.7.5-IMX6.VAR\_R17

sudo cp -R /tmp/lib/modules/4.7.5-IMX6.VAR\_R17/\* /mnt/rootfs//lib/modules/4.7.5-IMX6.VAR\_R17/

16. To see the process in Yosemite board, using minicom (connected through serial port):

sudo minicom -D /dev/ttyUSB0

# Appendix-B

# Installation of erquired components

Steps for installing Python and other required components:

Edit the sources.list file to add the following content:

deb http://ftp.us.debian.org/debian/ jessie maiap n contrib non-free

#deb-src http://ftp.us.debian.org/debian/ jessie main contrib non-free

deb http://ftp.us.debian.org/debian/ jessie-updates main contrib non-free

#deb-src http://ftp.us.debian.org/debian/ jessie-updates main contrib non-free

deb http://security.debian.org/ jessie/updates main contrib non-free

#deb-src http://security.debian.org/ jessie/updates main contrib non-free

#deb http://ftp.debian.org/debian jessie-backports main contrib non-free

##deb-src http://ftp.debian.org/debian jessie-backports main contrib non-free

sudo apt-get update

sudo apt-get install python fping

sudo apt-get install python-setuptools

sudo wget ftp://ftp.netperf.org/netperf/netperf-2.7.0.tar.gz

gunzip netperf-2.7.0.tar.gz

tar –xvf netperf-2.7.0.tar

cd netperf-2.7.0

./configure --enable-demo

cd ~

sudo install ~/netperf-2.7.0/Makefile.in ~

sudo netstat -lpn |grep 12865

sudo kill <process number)

sudo netserver & (Starting netserver of netperf)

Steps for installing FLENT:

wget http://download.opensuse.org/repositories/home:tohojo:flent/xUbuntu\_12.04/Release.key

sudo apt-key add - < Release.key

sudo sh -c "echo 'deb http://download.opensuse.org/repositories/home:/tohojo:/flent/xUbuntu\_15.04/ /' > /etc/apt/sources.list.d/flent.list"

sudo apt-get update

sudo apt-get install flent

# Flent commands

sudo flent rrul -f stats -l 10 -H 10.15.140.239 -t results -o file1

Using rrul for a combination of TCP upload and download, UDP upload and download and Ping.

Other file formats:

-f org\_table - for table format

-p all\_scaled – for plots in .png format