NETCONF:Testing

This page contains information about NETCONF (Scale, Performance etc.) testing.

Scale tests

Scale tests for NETCONF in ODL.

NETCONF southbound scale test

Goal of this test is to measure how many NETCONF devices can be mounted by ODL with a set amount of RAM.

Scenario

1. Start netconf-testtool that starts the desired amount of netconf servers
2. Testtool generates initial configuration for odl
3. ODL tries to connect to all of the simulated devices.
4. Measure the amount of devices connected(if all weren't connected)
5. Measure the time until odl connected to all the devices with a certain amount of RAM(2,4,8,16 GB)

How to

1. Make sure the open file limit is set reasonably high to the amount of devices started: https://wiki.opendaylight.org/view/OpenDaylight_Controller:Netconf:Testtool#Too_many_files_open
2. Unpackage a clean odl distribution, our scale utility will take care of feature installation and config generation
3. Download netconf scale-util : https://nexus.opendaylight.org/content/repositories/opendaylight.snapshot/org/opendaylight/netconf/netconf-testtool/1.1.0-SNAPSHOT/netconf-testtool-1.1.0-20160308.161039-64-scale-util.jar
4. Run the scale tool :

```
 java -Xmx8G -jar scale-util-1.1.0-SNAPSHOT-scale-util.jar --distribution-folder ./distribution-karaf-0.4.0-Beryllium --device-count 8000 --ssh false --exi false --generate-configs-batch-size 1000
```

The scale util needs to be pointed to an unpacked distribution (--distribution-folder argument) and handles the karaf start and feature installation. While the test is running the utility is also checking Restconf periodically to see the current status of all netconf devices. After the test completes successfully karaf is stopped, features are cleaned and the whole test is restarted with more devices(currently hardcoded 1000, parameter for this needs to be added). If you are starting with more ram than 2GB you should start with more devices than 8k.right away. The test results are being logged periodically into the scale-results.log that's present in the same location as the util jar. If you are running with even more devices the ram for testtools should be increased aswell.

To run the test with tcp add --ssh false argument when starting the scale-util.

To rerun the test with more ram available to odl you need to edit the /${distribution-location}/bin/setenv script line:

```
 export JAVA_MAX_MEM="4G"
```

NOTE: the fastest way to find out how many devices odl can handle at a given ram is to start the test with a larger amount of devices than it can handle, set the config batch size to 1k and start the test. You can then analyze the result log to see where there seems to be a drop off in the connected devices/or the test timed out.

Results

Beryllium

Environment:

- OS: Ubuntu Linux 4.2.0-30-generic x86_64
- CPU: Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz, 40 cores
- RAM: 128GB
- Network: Single VM for both ODL and simulated NETCONF device
- JVM: Oracle 1.8.0_73

Other configuration:

- No EXI

In this test the simulated devices testtool started only had the base capabilites that testtool has included by default(ietf-netconf-monitoring, ietf-inet-types, ietf-yang-types).

Netconf scale test			
Ram for ODL - 2GB			
transport	devices	time needed	
ssh	8000	3m 40s	4k batches, starts having low memory(most of the time in GC) issues around 8k devices.
ssh	9000	12m 16s	1k batches, times out after 20minutes
tcp	20000	6m 03s	4k batches, reached 20min timeout, maybe can handle a bit more
tcp	21000	18m 54s	1k batches, reached 20min timeout, maybe can handle a bit more
Ram for ODL - 4GB			
ssh	14000	9m 28s	
ssh	15000	17m 20s	timeout after 20minutes, hits the ram limit
tcp	24000	18m 31s	1k batches
tcp	28000	17m 27s	2k batches, timeout after 20min, should be able to get higher but needs more time

With tcp we also noticed that after 15k devices theres a pretty big slowdown with pushing/handling the configs as there starts to be increasing gaps between the individual batches.

Beryllium SR3

Environment:

- OS: Fedora 23 Linux 4.2.3-300.fc23.x86_64
- CPU: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz, 88 cores
- RAM: 64GB
- Network: Both ODL and simulated NETCONF device were on the same test system
- JVM: Oracle 1.8.0_101

Other configurations

- No EXI

Howto

- Make sure the open file limit is set reasonably high to the amount of devices started: https://wiki.opendaylight.org/view/OpenDaylight_Controller:Netconf:Testtool#Too_many_files_open
- Unpackage a clean odl distribution, taken from https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distribution-karaf/0.4.3-Beryllium-SR3/distribution-karaf-0.4.3-Beryllium-SR3.zip
- Download netconf scale-util, place it in the current operating directory:

```
Git clone https://github.com/opendaylight/netconf.git
cd netconf
mvn clean install
cp ~/netconf/netconf/tools/netconf-testtool/target/scale-util-1.2.0-SNAPSHOT.jar ~
```

- Run the scale tool :

```
 java -Xmx8G -jar scale-util-1.2.0-SNAPSHOT-scale-util.jar --distribution-folder ./distribution-karaf-0.4.0-Beryllium --device-count 8000 --ssh false --exi false --generate-configs-batch-size 1000
```

Results:
The Netconf Southbound test would repeatedly fail with a "java.lang.OutOfMemoryError: unable to create a new native thread" error message. The tester used the command set defined above, although the flag -Xmx8G was decreased to 4G, increased to 16G, and 32G -- all with the same result. At the same time this tester increased the value of JAVA_MAX_MEM to 16G and 32G, and still encountered the java.lang.OutOfMemoryError message. Log files of the failure and hs_err_pid log files captured and saved for examination. These are available on request.

Performance tests

Performance tests for NETCONF in ODL.

NETCONF northbound performance test

Goal of this test is to measure is the performance of an external NETCONF client uploading information into ODL (Global Datastore) using just NETCONF northbound server.

Scenario

1. ODL controller starts with simple l2fib models and NETCONF northbound for MD-SAL enabled
2. External fast netconf client writes lots of l2fib entries into MD-SAL's global DataStore using NETCONF northbound interface
3. The client measures time since sending out the 1st request until last response is received
4. After all the l2fibs are in ODL, performance is calculated in terms of requests and l2fibs written per scond

How to

This how to will be split into multiple ordered sections:

ODL and tooling setup

- Build or download beryllium based ncmount distribution of ODL(the distro has MD-SAL NETCONF northbound enabled and contains l2fib models)
  - Download from: //TODO-add-link-to-prebuilt-distro ncmount l2fib distro, unzip and "cd ncmount-karaf-1.1.0-SNAPSHOT/"
  - Or build by:

```
git clone https://git.opendaylight.org/gerrit/coretutorials
cd coretutorials
git checkout stable/beryllium
git fetch https://git.opendaylight.org/gerrit/coretutorials refs/changes/16/35916/1 && git
checkout FETCH_HEAD
cd ncmount
mvn clean install -DskipTests -Dcheckstyle.skip
cd karaf/target/assembly/
```

- Download Beryllium version of NETCONF stress client tool and untar
- Download the rest of testing resources and unzip next to the client
- Start ODL distribution:

```
./bin/karaf
```

- Wait until you see in logs:

```
NETCONF Node: controller-config is fully connected
```

- By default, only the SSH endpoint (port 2830) is opened for MD-SAL, but TCP(will also be tested, port 2831) can be easily enabled by performing REST call:

```
curl -u "admin:admin" -H "Accept: application/xml" -H "Content-type: application/xml" --request
POST 'http://localhost:8181/restconf/config/network-topology:network-topology/topology/topology-
netconf/node/controller-config/yang-ext:mount/config:modules' --data '<module xmlns="urn:
opendaylight:params:xml:ns:yang:controller:config"> \
            <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:netconf:
northbound:tcp">prefix:netconf-northbound-tcp</type> \
            <name>netconf-mdsal-tcp-server</name> \
            <dispatcher xmlns="urn:opendaylight:params:xml:ns:yang:controller:netconf:northbound:
tcp"> \
                <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:config:
netconf:northbound">prefix:netconf-server-dispatcher</type> \
                <name>netconf-mdsal-server-dispatcher</name> \
            </dispatcher> \
        </module>'
```

- In logs, you should see:

```
Netconf TCP endpoint started successfully at /0.0.0.0:2831
```

Testing over TCP[edit]

- Make sure ODL is up and running according to the previous section
- Go into the folder, where stress client and other test resources have been unpacked. The folder should look like this:

```
edit-l2fib-1000.txt  edit-l2fib-1.txt           netconf-north-perf-test-files.
zip                     stress-client-1.0.0-Beryllium-package
edit-l2fib-100.txt   edit-l2fib-delete-all.txt  netconf-testtool-1.0.0-Beryllium-stress-client.
tar.gz
```

- Execute the client:

```
java -Xmx2G -XX:MaxPermSize=256M -jar stress-client-1.0.0-Beryllium-package/stress-client-1.0.0-Beryllium-stress-client.jar --ip 127.0.0.1 --port 2831 --edits 10000 --exi false --ssh false --username admin --password admin --thread-amount 1 --async false --edit-batch-size 10000 --edit-content edit-l2fib-1.txt
```

- This execution configuration is:
  - 10000 edit-config RPCs (--edits 10000)
  - 1 l2fib entry per edit-config message (see edit-l2fib-1.txt)
    - since there's one l2fib in file and 10k requests will be performed, total amount of l2fibs in ODL will be 10k
    - The input file contains a placeholder for {PHYS_ADDR}, that's replaced by client with real physical address. Each l2fib per each request gets a different physical address, causing all of the l2fibs to be different and really stored in ODL's global DataStore. There are more placeholders available from the client.
  - 1 commit RPC per execution (--edit-batch-size)
  - over TCP (--ssh false)
  - from 1 thread (--thread-amount 1), synchronous (--async false) - async means whether the thread waits for reply after each edit or executes them asynchronously from dedicated thread and handles responses in a different one
  - No EXI (--exi false)
  - Towards ODL listening at 127.0.0.1 2831 (admin:admin) - that's the default config for MD-SAL NETCONF northbound
- The client will be working for a moment and before it exits, it prints results to stdout:

```
FINISHED. Execution time: 5.585 s
Requests per second: 1790.8309455587394
```

- The number: requests per second gives only the performance of edit-config execution. To calculate l2fibs/second, multiply that number by "number of l2fibs in edit-content file", which is 1 in our case so l2fibs/second is equal to requests/second = 1790.
- L2fib count in MD-SAL can be verified with:

```
curl -u "admin:admin" http://localhost:8181/restconf/config/ncmount-l2fib:bridge-domains | grep -o forward | wc -l
```

- Make sure to delete the DataStore for l2fibs by executing the client with input file for delete before executing further tests (Repeat from step "Execute the client"):

```
java -jar stress-client-1.0.0-Beryllium-package/stress-client-1.0.0-Beryllium-stress-client.jar --ip 127.0.0.1 --port 2831 --edits 1 --exi false --ssh false --username admin --password admin --thread-amount 1 --async false --edit-content edit-l2fib-delete-all.txt
```

- It is advised to repeat the test before reading final performance results.

Note: The client has many configuration options. Use -h to see all of them. Note: There are 3 edit-content files in the resources. Each contains a different number of l2-fibs per edit-config: 1, 100 and 1000. Files with different amounts can be produced and used.

Testing over SSH

- Testing over SSH is almost identical to TCP, just make sure to change the port and ssh flag when executing the client:

```
java -Xmx2G -XX:MaxPermSize=256M -jar stress-client-1.0.0-Beryllium-package/stress-client-1.0.0-Beryllium-stress-client.jar --ip 127.0.0.1 --port 2830 --edits 10000 --exi false --ssh true --username admin --password admin --thread-amount 1 --async false --edit-batch-size 10000 --edit-content edit-l2fib-1.txt
```

- The results should be most-likely worse compared to TCP:

```
FINISHED. Execution time: 11.64 s
Requests per second: 859.106529209622
```

- Delete can be performed same as with TCP, since it doesn't matter if delete is executed over TCP or SSH

Results

Beryllium

Environment:

- OS: Ubuntu Linux 4.2.0-30-generic x86_64
- CPU: Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz, 40 cores
- RAM: 128GB
- Network: Single VM for both ODL and simulated NETCONF device
- JVM: Oracle 1.8.0_73

Base configuration:

- Single client thread
- Synchronous client thread
- No SSH
- No EXI
- ODL HEAP: 8 Gb (edit in bin/setenv before executing karaf)

Measured numbers with a single client:

Netconf northbound single client performance				
Client type	l2fib per request	**TCP performance**	**SSH performance**	Total l2fibs
Sync	1	**1 730 edits/s 1 730 l2fibs/s**	**1 474 edits/s 1 474 l2fibs/s**	100k
Async	1	**7 063 edits/s 7 063 l2fibs/s**	**6 600 edits/s 6 600 l2fibs/s**	100k
Sync	100	**233 edits/s 23 372 l2fibs/s**	**148 edits/s 14 850 l2fibs/s**	500k
Async	100	**421 edits/s 42 179 l2fibs/s**	**386 edits/s 38 600 l2fibs/s**	500k
Sync	500	**61 edits/s 30 935 l2fibs/s**	**13 edits/s 6 590 l2fibs/s**	1M
Async	500	**81 edits/s 40 894 l2fibs/s**	**69 edits/s 34 500 l2fibs/s**	1M
Sync	1000	**35 edits/s 35 365 l2fibs/s**	**13 edits/s 13 248 l2fibs/s**	1M
Async	1000	**38 edits/s 38 099 l2fibs/s**	**19 edits/s 19 898 l2fibs/s**	1M

Multiple clients:

Netconf northbound mutliplce client performance					
Clients	Client type	l2fib per request	**TCP performance**	**SSH performance**	Total l2fibs
8	Sync	1	**23 010 edits/s 23 010 l2fibs/s**	**13 847 edits/s 13 847 l2fibs/s**	400k
8	Async	1	**41 114 edits/s 41 114 l2fibs/s**	**12 527 edits/s 12 527 l2fibs/s**	400k
16	Sync	1	**31 743 edits/s 31 743 l2fibs/s**	**15 879 edits/s 15 879 l2fibs/s**	400k
16	Async	1	**43 252 edits/s 43 252 l2fibs/s**	**12 496 edits/s 12 496 l2fibs/s**	400k
8	Sync	100	**852 edits/s 85 215 l2fibs/s**	**769 edits/s 76 989 l2fibs/s**	1,6M
8	Async	100	**984 edits/s 98 419 l2fibs/s**	**869 edits/s 86 923 l2fibs/s**	1,6M
16	Sync	100	**808 edits/s 80 885 l2fibs/s**	**723 edits/s 72 345 l2fibs/s**	1,6M
16	Async	100	**852 edits/s 85 224 l2fibs/s**	**749 edits/s 74 962 l2fibs/s**	1,6M
8	Sync	500			

8	Async	500			
16	Sync	500			
16	Async	500			
8	Sync	1000			
8	Async	1000			
16	Sync	1000			
16	Async	1000			

Beryllium SR3

Environment

- OS: Fedora 23 Linux 4.2.3-300.fc23.x86_64
- CPU: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz, 88 cores
- RAM: 64GB
- Network: Both ODL and simulated NETCONF device on same system
- JVM: Oracle 1.8.0_60

Other configurations

N/A

Steps to recreate

- Make sure the open file limit is set reasonably high to the amount of devices started: https://wiki.opendaylight.org/view/OpenDaylight_Controller:Netconf:Testtool#Too_many_files_open
- Build beryllium based ncmount distribution of ODL as follows (the prebuilt version could not be found on the OpenDaylight website):

```
git clone https://git.opendaylight.org/gerrit/coretutorials
cd coretutorials
git checkout master
cd ncmount
mvn clean install –DskipTests –Dcheckstyle.skip
cd karaf/target/assembly
```

(When the other set of instructions were followed, when the tester executed "mvn clean install –DskipTests –Dcheckstyle.skip", maven failed due to "Non-resolvable parent POM for org.opendaylight.coretutorials:ncmount-aggregator:1.1.0-SNAPSHOT". Attempting to hand-edit all of the relevant pom.xml files proved impractical -- too many files to investigate & fix -- so the tester used this work around. At least maven completed the command.)

- Git clone https://github.com/opendaylight/netconf.git

```
mvn clean install
```

(Trying to download the Beryllium version of the NETCONF stress tool from the URL in the Wiki failed. When the tester used the link, he encountered the error “404 Not found: repository with ID: “autorelease-1074” not found”.)

- Download the rest of testing resources (per link above) and unzip next to the client
- Start ODL distribution:

```
./bin/karaf
```

At this point, the expected message "NETCONF Node: controller-config is fully connected" was not found in the logs. The tester was not confident he understood these instructions, & stopped here, waiting on further instructions.

NETCONF southbound performance test

Goal of this test is to measure is the performance of a NETCONF device uploading information into ODL using just NETCONF southbound notifications.

Scenario

1. ODL controller mounts a simulated NETCONF device (simulates Cisco IOS XR thanks to some of its routing models)
2. Small ODL application triggers a NETCONF notification stream for new mountpoint

3. Simulated device immediately starts sending routes with a certain number of prefixes into ODL
4. Application waits until all notifications have been processed and measures the execution/receiving time
5. Application outputs the performance numbers: notifications/second and prefixes/second to the log

Notes:

- Application is Binding Aware
- Application does not perform any additional processing of the routes/prefixes, just counts it in and throws away.

How to

- Create or use empty folder and keep all test related resources in that folder
- Download and unzip latest Opendaylight ncmount distribution
  - Ncmount distribution is required since it contains the performance testing code
- Download Beryllium NETCONF Testtool
- Download and unzip additional testing resources:
  - Testing yang schema set - Cisco IOS XR routing models
  - NETCONF notification inputs - XML rendered NETCONF notifications according to XR routing models
- Start testtool:

```
 java -jar netconf-testtool-1.0.0-Beryllium-executable.jar --schemas-dir ./xrSchemas/ --ssh false
--exi false --notification-file i2rs-notifs-perf100k.xml
```

- Start ODL distribution:

```
./ncmount-karaf-1.1.0-SNAPSHOT/bin/karaf
```

and wait until you there's following message in the log:

```
NETCONF Node: controller-config is fully connected
```

- Mount the testtool simulated device in ODL using following REST call:

```
 curl -u "admin:admin" -H "Accept: application/xml" -H "Content-type: application/xml" --request
POST 'http://localhost:8181/restconf/config/network-topology:network-topology/topology/topology-
netconf/node/controller-config/yang-ext:mount/config:modules' --data '<module xmlns="urn:
opendaylight:params:xml:ns:yang:controller:config"> \
     <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">
prefix:sal-netconf-connector</type> \
     <name>controller-notif-100000</name> \
     <address xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">127.
0.0.1</address> \
     <port xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">17830<
/port> \
     <username xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">adm
in</username> \
     <password xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">adm
in</password> \
     <tcp-only xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">tru
e</tcp-only> \
     <keepalive-delay xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:
netconf">0</keepalive-delay> \
     <event-executor xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:
netconf"> \
       <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:netty">prefix:netty-
event-executor</type> \
       <name>global-event-executor</name> \
     </event-executor> \
     <binding-registry xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:
netconf"> \
       <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:md:sal:binding">prefix:
binding-broker-osgi-registry</type> \
       <name>binding-osgi-broker</name> \
     </binding-registry> \
     <dom-registry xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf"
> \
       <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:md:sal:dom">prefix:dom-
broker-osgi-registry</type> \
       <name>dom-broker</name> \
     </dom-registry> \
     <client-dispatcher xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:
netconf"> \
       <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:config:netconf">prefix:
netconf-client-dispatcher</type> \
       <name>global-netconf-dispatcher</name> \
     </client-dispatcher> \
     <processing-executor xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:
netconf"> \
       <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:threadpool">prefix:
threadpool</type> \
       <name>global-netconf-processing-executor</name> \
     </processing-executor> \
   </module>'
```

- After the POST, you should see following message in log (expressing the device is mounted):

```
 NETCONF Node: controller-notif-100000 is fully connected
```

- After a while, measurements should appear in the log:

```
Elapsed ms for 100000 notifications: 9092
Performance (notifications/second): 10998.680158380994
Performance (prefixes/second): 10998.680158380994
```

- Test can be repeated by deleting the mountpoint and then continuing from step "Mount the testtool simulated device in ODL" :

```
   curl -u "admin:admin" -H "Accept: application/xml" -H "Content-type: application/xml" --
request DELETE  http://localhost:8181/restconf/config/network-topology:network-topology/topology
/topology-netconf/node/controller-config/yang-ext:mount/config:modules/module/odl-sal-netconf-
connector-cfg:sal-netconf-connector/controller-notif-100000
```

- It is advised to repeat the test a couple times, before reading the results so JVMs are able to heat up to proper operating temperature

Notes:

- There are 3 different input notification files. They differ in the number of prefixes within a single notification: 1, 10, 100. Each file is set to send 100k notifications
- The numbers can be changed at will

- The name of the mountpoint must by suffixed with "-100000". The number indicates the count of notifications to expect in ODL and must be the same number as number of notifications configured in the input file. Otherwise the test will be inaccurate or never finish
- 100k Notifications might be too low for measuring performance in fast environments. So to increase the count to 1M you need to change the property "times" in the input files and also update the "name" attribute in POST request by changing the number from 100 000 to 1 000 000.

Results

Beryllium

Environment:

- OS: Ubuntu Linux 4.2.0-30-generic x86_64
- CPU: Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz, 40 cores
- RAM: 128GB
- Network: Single VM for both ODL and simulated NETCONF device
- JVM: Oracle 1.8.0_73

Configuration:

- Single simulated device
- ODL with a single active NETCONF session
- ODL HEAP: 8 Gb
- No SSH
- No EXI

Measured performance send/receive of 1M Notifications:

Netconf southbound notification performance				
Total notifications	Prefixes per notification	**TCP performance**	**SSH performance**	**TCP+EXI performance**
100k	1	**10716 notifications/s 10716 prefixes/s**	**9828 notifications/s 9828 prefixes/s**	
100k	2	**7112 notifications/s 14224 prefixes/s**	**5496 notifications/s 10992 prefixes/s**	
100k	10	**1996 notifications/s 19965 prefixes/s**	**1635 notifications/s 16356 prefixes/s**	

* SSH test performance is worse over time (with test reruns on the same ODL) and more memory is used but not freed. Looks like a memory leak. https://bugs.opendaylight.org/show_bug.cgi?id=5488

Beryllium-SR-3

Test failed to run.

Environment

- OS: Fedora 23 Linux 4.2.3-300.fc23.x86_64
- CPU: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz, 88 cores
- RAM: 64GB
- Network: Both ODL and simulated NETCONF device were on the same test system
- JVM: Oracle 1.8.0_101

Steps performed

- Download and unzip latest Opendaylight ncmount distribution
- Followed the instructions under the “NETCONF northbound performance test” to build the netconf test tool, since the link to the excutable above was not working. Maven built the package without any issues, although the step “git fetch https://git.opendaylight.org/gerrit/coretutorials refs /changes/16/35916/1 && git checkout FETCH_HEAD” was skipped due to problems with the pom.xml files maven would not build the package.
- The file netconf-testtool-1.0.0-Beryllium-executable.jar was copied from its location under $HOME/netconf into $HOME
- Downloaded & unzipped the additional testing resourced into the home directory
- The testtool was started with “java -jar netconf-testtool-1.0.0-Beryllium-executable.jar --schemas-dir ./xrSchemas/ --ssh false --exi false --notification-file i2rs-notifs-perf100k.xml”
- Then the ODL executable was from the home directory with “ncmount-karaf-1.1.0-SNAPSHOT/bin/karaf” (NB -- both are on the same machine; different virtual ttys were used.
- The wiki does not state where to look for the log message – i.e., which log to monitor – so ncmount-karaf-1.1.0-SNAPSHOT/data/log/karaf.log was monitored. However, the message “NETCONF Node: controller-config is fully connected” never appeared; instead there was a lengthy error message stating “Unable to setup SSH session on demand”, & some further details. (Output captured.)
- Nevertheless, I executed the lengthy curl command, just to see what happened. Instead of the expected message in the log (again, I assumed the author of the Wiki page meant the karaf.log file already monitored), the log showed a lengthy error message beginning with “java.net. ConnectException: Connection timed out”, & listing a series of Java objects.

NETCONF end-to-end performance test

Goal of this test is to measure is the performance the end-to-end (external REST client -> REST north -> MD-SAL -> NETCONF south -> NETCONF device) performance of ODL using both NETCONF and RESTCONF.

Scenario

1. ODL controller mounts a simulated NETCONF device (simulates Cisco IOS XR thanks to some of its routing models)
2. External REST client starts sending prefixes via RESTCONF
3. ODL application handles the calls, transforms the request into device specific models and writes to the device
4. The client waits until all of its requests were handled in RESTCONF and calculates the rate

How to

- Build or download beryllium based ncmount distribution of ODL(the distro has MD-SAL NETCONF northbound enabled and contains l2fib models)
  - Download from: //TODO-add-link-to-official-ncmount-distro-after-fix-is-merged ncmount l2fib distro, unzip and "cd ncmount-karaf-1.1.0-SNAPSHOT/"
  - Or build by:

```
 git clone https://git.opendaylight.org/gerrit/coretutorials
 cd coretutorials
 git fetch https://git.opendaylight.org/gerrit/coretutorials refs/changes/54/36054/1 && git
checkout FETCH_HEAD
 cd ncmount
 mvn clean install -DskipTests -Dcheckstyle.skip
 cd karaf/target/assembly/
```

- Download Beryllium NETCONF Testtool
- Download the rest of testing resources and unzip
- Start testtool:

```
 java -jar netconf-testtool-1.0.0-Beryllium-executable.jar --schemas-dir ./xrSchemas/ --ssh false
--exi false --distribution-folder /ncmount-karaf-1.1.0-SNAPSHOT
```

- Start ODL distribution:

```
 ./ncmount-karaf-1.1.0-SNAPSHOT/bin/karaf
```

- Wait until ODL distribution is fully up and running
- Execute rest-stress-client to perform a test run

```
 java -jar rest-stress-client.jar --ip localhost --port 8181 --destination /restconf/operations
/ncmount:write-routes --edits 100 --edit-content json_routes_10.json --async-requests true --
throttle 1000 --auth admin admin
```

- This writes a batch of 10 prefixes into the mounted device 100 times.
- See rest-stress-client help for further information/options

Multiple devices

To run this test with multiple devices/clients these changes apply:

- Start testtool with more devices

```
 --device-count 16
```

- After odl is up and running run the rest perf client with these additional arguments:

```
 --same-device false  --thread-amount 16
```

--thread-amount is the number of clients. Each client will be mapped to a single device.