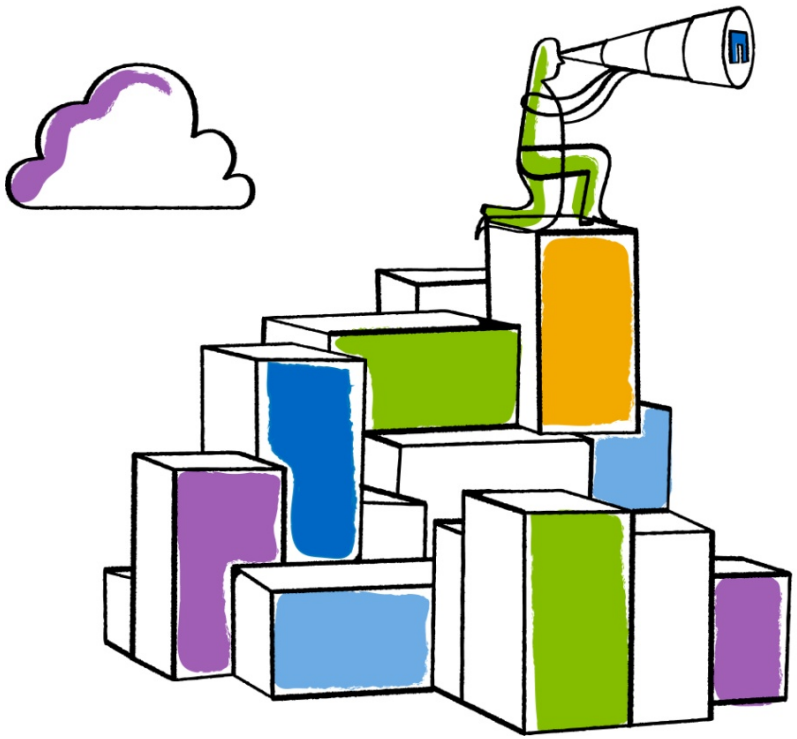
NetApp

Go further, faster®

NFStest



Jorge Mora

mora@netapp.com

February 27, 2013

Outline

- Packet trace module
- Test utilities module
- Tests
- Installation and setup
- Future work

Packet trace module

- Python module
- Takes a trace file created by tcpdump as input
- Decodes one packet at a time
- Search for specific packets
- Helps create completely automated tests

Packet trace module

```
from packet.pktt import Pktt

x = Pktt("/traces/tracefile.cap")

# Iterate over all packets in the trace file
for pkt in x:
    print pkt
```

Packet trace module

```
from packet.pktt import Pktt

x = Pktt("/traces/tracefile.cap")

# Iterate over all packets in the trace file
for pkt in x:
    print pkt

x.next()
```

Packet trace module

```
from packet.pktt import Pktt

x = Pktt("/traces/tracefile.cap")

# Iterate over all packets in the trace file
for pkt in x:
    print pkt

x.next()

x.rewind([index])
```

Packet trace module

```
from packet.pktt import Pktt

x = Pktt("/traces/tracefile.cap")

# Iterate over all packets in the trace file
for pkt in x:
    print pkt

x.next()

x.rewind([index])

x.match(match_str [, maxindex])
```

Packet trace module

```
x.match(match_str [, maxindex])

Examples:

# Find the packet with both the ACK and SYN TCP flags
# set to 1
pkt = x.match("TCP.flags.ACK == 1 and TCP.flags.SYN == 1")

# Find the next NFS EXCHANGE_ID request
pkt = x.match("NFS.argop == 42")

# Find the next NFS OPEN request or reply
pkt = x.match("NFS.op == 18")
```

Packet trace module

Examples:

```
# Find all packets coming from subnet 192.168.1.0/24 using
# a regular expression
while x.match(r"IP.src == re('192\.168\.1\.\d*')"):
    print x.pkt

# Find GETATTR asking for FATTR4_FS_LAYOUT_TYPE(bit 62)
call = x.match("NFS.attr_request & (1<<62) != 0")
if call:
    # Find GETATTR reply
    xid = call.rpc.xid
    # Find reply where the number 62 is in the array
    # NFS.obj_attributes
    mstr = "RPC.xid == %d and 62 in NFS.obj_attributes" % xid
    reply = x.match(mstr)
```

Packet trace module

Examples:

```
# Find the next WRITE request
pkt = x.match("NFS.argop == 38”)
if pkt:
    print pkt.nfs
    print pkt.nfs.NFSop

# Same as above, but using membership test operator instead
if ("NFS.argop == 38" in x):
    print x.pkt.nfs
    print x.pkt.nfs.NFSop
```

Packet trace module

Packet layers supported:

- ETHERNET II (RFC 894)
- IP layer (only supports v4)
- TCP layer
- RPC layer
- NFS v4.0
- NFS v4.1 including pNFS file layouts

Test utilities module

- Process command line arguments
- Functionality for PASS/FAIL
- Test grouping functionality
- Mechanism to run individual tests
- Multiple client support
- Logging mechanism
- Debug info control
- Mount/Unmount control
- Create files/directories
- Mechanism to start a packet trace
- Mechanism to capture NFS debugging messages
- Support for pNFS testing

Test utilities module

```
from nfstest.test_util import TestUtil

x = TestUtil()
x.scan_options()
x.test_group(msg)
x.test(expr, msg [, subtest][, failmsg])
x.run_tests(msg)
x.create_host(host)
x.dprint(level, msg)
x.mount()
x.umount()
x.trace_start()
x.trace_stop()
x.trace_open()
x.nfs_debug_enable(nfsdebug=flags|rpcdebug=flags)
x.nfs_debug_reset()
x.exit()
```

Test utilities module

--help

Display usage information and options available

$ nfstest_posix --help

```
Options:
  --version             Show program's version number and exit
  --help                Show this help message and exit
  --file=<file>         Options file
  --server=<server>     Server name or IP address
  --port=<port>         NFS server port [default: 2049]
  --nfsversion=3|4      NFS version [default: 4]
  --minorversion=0|1    Minor version [default: 1]
  --export=<export>     Exported file system to mount
                        [default: '/']
  --mtpoint=<mntpoint> Mount point [default: '/mnt/t']
  --verbose=none|opts|debug|all|bitmask
                        Verbose level [default: 'none']
```

Test utilities module

--verbose <none|opts|info|debug|all|intbitmask>
Verbose level for info/debug messages

$ nfstest_posix --server 192.168.0.11 --verbose all
$ nfstest_posix --server 192.168.0.11 --verbose 0xFF

DBG2: Mount volume: mount -t nfs4 -o minorversion=1,hard,intr …
DBG7: Allocated aligned buffer of 8192 bytes @ 0x2349000
DBG3: Open file /mnt/t/test_dio_2557_20121011204501_f_1 for reading
DBG3: Read file 4096@258048
PASS: READ right before end of file should return correct read count (4096)
DBG3: Read file 4096@262144
PASS: READ at end of file should return read count = 0
DBG7: Freeing allocated buffers

Test utilities module

--tverbose <group|normal|verbose>
Verbose level for test messages (default: normal)

$ nfstest_posix --server 192.168.0.11
*** Verify POSIX API access() on NFSv4
PASS: access - file access allowed with mode F_OK
PASS: access - file access not allowed with mode F_OK for a non-existent file
PASS: access - file access allowed with mode R_OK for file with permissions 0777
PASS: access - file access allowed with mode W_OK for file with permissions 0777
PASS: access - file access allowed with mode X_OK for file with permissions 0777
...

$ nfstest_posix --server 192.168.0.11 --tverbose group
PASS: Verify POSIX API access() on NFSv4 (58 passed, 0 failed)

Test utilities module

--createlog
Create log file when specified

--keeptraces
Do not remove any trace files at the end of execution

--nfsdebug <flags>
Set NFS kernel debug flags and save log messages

--rpcdebug <flags>
Set RPC kernel debug flags and save log messages

Test utilities module

--bugmsgs <file>
File containing test messages to mark as bugs if they failed

--ignore
Ignore all bugs given by bugmsgs

--nfsversion 2|3|4
NFS version [default: 4]

--minorversion 0|1
NFS minorversion [default: 1]

Tests

- nfstest_pnfs
- nfstest_posix
- nfstest_delegation
- nfstest_dio
- nfstest_cache

Tests

$ nfstest_pnfs --server 192.168.0.11

*** Verify traffic for file using pNFS - READ
PASS: EXCHANGE_ID should be sent to MDS
PASS: EXCHGID4_FLAG_USE_PNFS_MDS should be set
PASS: EXCHGID4_FLAG_USE_NON_PNFS should not be set
PASS: CREATE_SESSION should be sent to MDS
PASS: SEQUENCE request should start with a sequence id of 1
PASS: RECLAIM_COMPLETE should be sent to MDS
PASS: GETATTR should be sent to MDS asking for FATTR4_LEASE_TIME
PASS: NFS server should return lease time(30) > 0
PASS: GETATTR should be sent to MDS asking for FATTR4_SUPPORTED_ATTRS
PASS: NFS server should support file type layouts (LAYOUT4_NFSV4_1_FILES)
PASS: GETATTR should be sent to MDS asking for FATTR4_FS_LAYOUT_TYPE
PASS: NFS server should return LAYOUT4_NFSV4_1_FILES in fs_layout_types
PASS: OPEN should be sent
PASS: LAYOUTGET layout type should be LAYOUT4_NFSV4_1_FILES
PASS: LAYOUTGET iomode should be LAYOUTIOMODE4_READ
PASS: LAYOUTGET should ask for full file layout
PASS: LAYOUTGET reply layout type should be LAYOUT4_NFSV4_1_FILES
...

Tests

$ nfstest_posix --server 192.168.0.62 --tverbose group

PASS: Verify POSIX API access() on NFSv3 (58 passed, 0 failed)
PASS: Verify POSIX API chdir() on NFSv3 (3 passed, 0 failed)
PASS: Verify POSIX API creat() on NFSv3 (6 passed, 0 failed)
PASS: Verify POSIX API fcntl() on NFSv3 (34 passed, 0 failed)
PASS: Verify POSIX API fdatasync() on NFSv3 (2 passed, 0 failed)
PASS: Verify POSIX API fstat() on NFSv3 (44 passed, 0 failed)
PASS: Verify POSIX API fstatvfs() on NFSv3 (20 passed, 0 failed)
PASS: Verify POSIX API fsync() on NFSv3 (2 passed, 0 failed)
PASS: Verify POSIX API link() on NFSv3 (5 passed, 0 failed)
PASS: Verify POSIX API lseek() on NFSv3 (10 passed, 0 failed)
PASS: Verify POSIX API lstat() on NFSv3 (44 passed, 0 failed)
PASS: Verify POSIX API mkdir() on NFSv3 (7 passed, 0 failed)
PASS: Verify POSIX API opendir() on NFSv3 (2 passed, 0 failed)
PASS: Verify POSIX API read() on NFSv3 (3 passed, 0 failed)
PASS: Verify POSIX API readdir() on NFSv3 (3 passed, 0 failed)
PASS: Verify POSIX API readlink() on NFSv3 (2 passed, 0 failed)
PASS: Verify POSIX API rename() on NFSv3 (10 passed, 0 failed)
PASS: Verify POSIX API rewinddir() on NFSv3 (4 passed, 0 failed)
...

Tests

$ nfstest_delegation --server 192.168.0.11 --client 192.168.0.18

*** Basic READ delegation tests
PASS: READ delegation should be granted
PASS: OPEN's should not be sent for the same file
PASS: READ stateid should be the DELEG stateid
PASS: READ's should not be sent when reading delegated file from a different process

*** Basic WRITE delegation tests
PASS: WRITE delegation should be granted
PASS: OPEN's should not be sent for the same file
PASS: WRITE stateid should be the DELEG stateid

*** Basic READ delegation tests with file lock
PASS: READ delegation should be granted
PASS: OPEN's should not be sent for the same file
PASS: READ stateid should be the DELEG stateid
PASS: READ's should not be sent when reading delegated file from a different process
PASS: LOCK should not be sent to the server
...

Tests

$ nfstest_dio --server 192.168.0.11

*** Verify eof marker is handled correctly when reading eof using aligned buffer
PASS: READ right before end of file should return correct read count (4096)
PASS: READ at end of file should return read count = 0

*** Verify data correctness when reading/writing using direct I/O
PASS: File created with buffered I/O is read correctly with direct I/O
PASS: File created with direct I/O is read correctly with buffered I/O

*** Verify fstat() gets correct file size after writing
PASS: The fstat() should get correct file size after every write
PASS: The fstat() should get correct file size after writing at offset = 10G

*** Verify READ is sent after writing when the file is open for both read & write
PASS: READ data should be correct
PASS: READ should be sent to the server
PASS: READ should be sent with correct offset (0) and count (4096)
...

Tests

$ nfstest_cache --server 192.168.0.11 --client 192.168.0.18 --nfsversion 3

*** Verify consistency of attribute caching with NFSv3 on a file acregmin = 10
PASS: File size should have not changed at t=0
PASS: File size should have not changed just before acregmin
PASS: File size should have changed just after acregmin

*** Verify consistency of attribute caching with NFSv3 on a file acregmin = 10 acregmax = 20
PASS: File size should have not changed at t=0
PASS: File size should have not changed just before acregmax
PASS: File size should have changed just after acregmax
PASS: File size should have not changed just before acregmin
PASS: File size should have changed just after acregmin

*** Verify consistency of attribute caching with NFSv3 on a directory acdirmin = 10
PASS: Hard link count should have not changed at t=0
PASS: Hard link count should have not changed just before acdirmin
PASS: Hard link count should have changed just after acdirmin
...

Installation and setup

Web page:
http://wiki.linux-nfs.org/wiki/index.php/NFStest

Download and setup:
$ git clone git://git.linux-nfs.org/projects/mora/nfstest.git
$ cd nfstest
$ sudo python setup.py install
$ man nfstest

Future work

- Add option to create html output of test results
- Add Web/GUI interface
- Fix rewind() so it does not go back to the start of file
- Fix TCP retransmission issue
- Fix TCP packet ordering issue
- Add IPv6 support
- Add NFSv3 support
- Add NFSv2 support
- Create regression tests for packet trace module

Thank you