APPENDIX A. UPDATING AND BUILDING THE SOFTWARE

The software consists of two packages built within the Vyatta framework; rohc-lib and rohc.

Rohc-lib contains the librarycode fetched from https://launchpad.net/rohc. The library requires access to a compiled Linux kernel and its corresponding source. The code can be accessed online through bazaar version control management. The code supplied by Thales in the Vyatta framework is using git to track changes locally. The launchpad directory outside the Vyatta framework is conforming to both bazaar and git, to allow fetching updated source from the internet, and providing this code to the Vyatta framework.

Update of rohc-lib code is done this way:

- Go lo launchpad/rohc/trunk directory
- Perform “bzr pull”
- Verify that the library changes are sane, then commit the changes to the git tree.
- Pull this tree from the rohc-lib module inside Vyatta.

Rohc contains the userspace application that support provides the framework around the library.

Note that the library must be installed on the buildmachine in the latest version to get a correct build of the userspace application.

Update and buildprocedure from scratch is as follows:

1. Update: Update rohc-lib as described above
2. Update: Go to build-iso directory, perform
    a. “<path>vc –b=oxnard checkout
    b. cd ../../..
    c. tools/update-git rohc/oxnard/build-iso
    d. cd –
    e. “<path>vc –b=rohc-1.4.0 checkout
3. Update: Perform a merge of all modules i.e. top-level, linux-image and iproute (git merge Oxnard)
4. Update: Login as root, perform
    a. make distclean
5. Update: Exit root, perform
    a. autoreconf –i && ./configure –with-vyatta-build-branch=oxnard
6. Build: Perform
    a. tools/submod-mk –c linux-image iproute rohc-lib
7. Build (IF rohc-lib has changed): Login as root, perform:
    a. dpkg –i pkgs/rohc-lib_999.dev_all.deb
    b. exit root
8. Build: Perform:
    a. tools/submod-mk –c rohc
9. Build: Log in as root, perform:
    a. make iso
10. Finished; iso can be found in build-iso/liveCD

NOTE! Do not perform step 6 and 7 if no update of Linux-image or rohc-lib source has taken place.

APPENDIX B. RUNNING THE LIBRARY

The Rohc implementation is setup by use of the CLI commonly used on the Thales ITR.

The configuration structure is:

```
Service {
        rohc {
                debug-level <0..5>
                interface xxx {
                        execute-script yyyy
                }
        }
}
```

The parameters are used as described below:

debug-level:

Defines the debug-level of the rohc user application. Setting the debug-level when rohc is running will restart the user application.

Debug of the rohc-library can only be set compiletime. Edit the file rohc-lib/debian/rules and change the line

configure += --enable-rohc-debug=0

to a value that is suitable.

interface

The interface to use compression on.

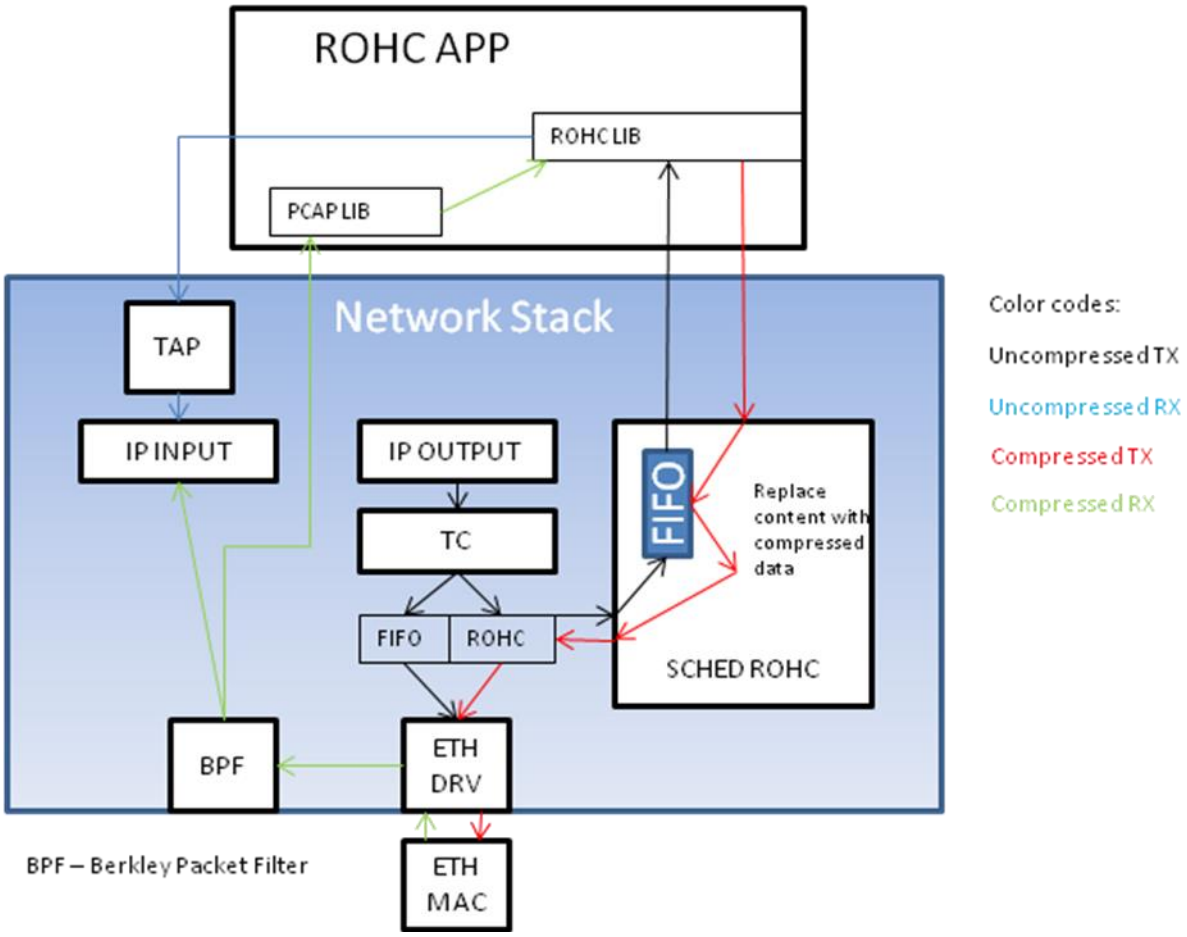
execute-script:

The parameter should point to a script that initializes qdisc in order to get the packets that shall be compressed into the correct rohc queue. Scripts are located in directory /opt/Vyatta/etc/rohc.

APPENDIX C. PACKET FLOW FOR ROHC

This is a short description of how packets to be compressed and decompressed by ROHC travel trough Linux network stack and user space application ROHC.

The information for packet flow is partly taken from “FFI-notat 2010/02528 Integration of Robust Header Compression (ROHC) in a Linux based tactical router – an extension to Linux traffic control”, chapter 3.4 and partly based on code reading.



Output of compressed packets:

- Packet arrives in IP OUTPUT from a local process or because of forwarding from another interface
- TC (Traffic Control) decides what queue to use for packets, default is FIFO qdisc. Here ROHC queue is available, routed by TOS value (depending on configuration)
- If packet matches a TC rule for ROHC, packet is sent to ROHC queue that forward the packet to a kernel module called “SCHED ROHC”. Here the packet is stored in a FIFO queue and user space app ROHC is informed that a packet is waiting
- ROHC APP read packet and send contents to ROHC LIBRARY for compression. The compressed payload is returned to the SCHED ROHC module
- SCHED ROHC fetches original packet from FIFO queue and replaces contents with compressed content before the packet is submitted to the Ethernet driver