Networking Technology#

io-pkt is intended to be a drop in replacement for io-net for those people who are dealing with the stack from an outside application point of view. It includes three stack variants, associated utilities, protocols, libraries and drivers.

The three stack variants are:

- **io-pkt-v4**: IPv4 version of the stack with no encryption or WiFi capability built in. This is a "reduced footprint" version of the stack.
- **io-pkt-v4-hc**: IPv4 version of the stack that has full encryption and WiFi capability built in and includes hardware accelerated crypto capability (aka FastIPSec).
- **io-pkt-v6-hc**: IPv6 version of the stack (includes IPv4 as part of v6) that has full encryption and WiFi capability, also with hardware accelerated crypto.

The full source for QNX's Transparent Distributed Protocol is also included in the repository. This source produces object libraries for both io-net and io-pkt, so those of you who want to build QNET for io-net can also use the source base here.

If you're looking for source, take a look at the Quick Guide to the Networking Source.

Pre-built binaries are available from from the Project Downloads Page. This page is where we'll be placing integration builds (raw builds that might not have undergone much, if any, testing), Milestone builds (which have had, at a minimum, sanity testing performed), Release Candidate builds (which have had full regression and feature testing done on them) as well as the final GA product builds.

Development Plans#

The first release of the Networking product is slated for release later this year. For more details, check out the Networking Roadmap. The initial release will include only those utilities which have been altered to allow correct operation with io-pkt. In many cases the new utilities will **not** work with io-net because they include enhanced features not supported by the old stack. After the initial source has been branched for release, we will start pulling the source for the remainder of the networking utilities (which weren't included as part of the initial release) into the repository. The goal is to have all networking related source code pulled into the public repository for early this year.

Technology Focus#

The following pages cover some of the key areas in technology in the Networking release

- Stack Architecture and Usage
- Packet Filtering, FireWalling and Network Address Translation
- IP Security and Hardware Encryption
- 802.11 a/b/g Wireless
- Transparent Distributed Processing (aka QNET)
- Network Drivers
- Other Stuff (utilities, libraries, protocols)
- Unnumbered interfaces
- Point to point ethernet
- io-net Migration
- Frequently Asked Questions

Tidbits#

The following links describe the status of some works in progress

- net-snmp

For Developers#

A description of how to download and build the source is located in the Source Guide. A good place to look at for information about how the stack does what it does is the NetBSD Documentation web pages (e.g. http://www.netbsd.org/docs/internals/en/chap-networking-core.html). While our stack isn't identical because of the OS changes, the fundamental structure has been kept as close as possible to allow straight forward porting of stack updates and user applications from the open source domain.

Migration from io-net#

For people using the BSD socket interface, no changes are required in your application to work with the new stack. The socket interface is designed to be binary compatible with the original io-net socket library so you don't even have to re-compile your application. For those that are more intimately tied to the stack binary itself (e.g. those with their own protocols, filters or converters or utilities that make use of stack structures directly), then you'll have some re-writing to do to work into the new stack. The good news is that, with the stack upgraded and compatible with the latest rev of NetBSD, you may be able to use more portable interfaces for your application than were available for io-net before (e.g. Berkley Packet Filter interface and the PF interface). Details on migration issues are covered in the io-net migration wiki page.

Driver Development#

There are three "types" of drivers supported. These are "native drivers" (drivers written and optimised for use with io-pkt), "io-net drivers" (drivers written for the io-net networking stack) and "NetBSD ported drivers" (drivers taken from the NetBSD source base and ported over without the addition of io-pkt optimisations / feature updates). Details on these driver types are covered in the Network Drivers. Note that io-pkt drivers are differentiated from io-net drivers via the name (they start with *devnp* rather than *devn*). This lets io-net / io-pkt drivers with the same root name to co-exist on the same system.

As a side note, there are some drivers which will not be available as source code drops due to non-disclosure agreements. Some of these drivers (e.g. WiFi drivers) can't be included as binary drops on this site due to licensing restrictions (since they require specific acceptance of a licensing agreement). These drivers will, however, be included as part of official betas and as part of the GA release of the product.

Debugging#

One of the really cool things about the way that the stack executes as a separate process inside of the operating system is that you can *use* a working version of the stack (including the previous generation io-net stack) to debug a non-working stack (or driver). If you start one stack in the "normal" way and start the second stack with the -ptcpip prefix=/sock2 (for example), you can attach the debugger to the second stack and step through the stack source as expected. You can build a version of the stack with debug information / symbols included by adding a new variant to the appropriate build directory after you've done a build from the trunk directory.

```
# cd trunk/sys
# cp io-pkt-v4.use io-pkt-v4-g.use
# cd target/x86
# addvariant o.v4.g
# cp o.v4/tgt.mk o.v4.g
# cd o.v4.g
# make install
```

This will create an x86/sbin/io-pkt-v4_g binary that you can now start and debug.
Alternatively you can compile as follows which will work from anywhere in the tree (Note: the following is CCOPTS=-<letter O><digit 0>).

```
# make CCOPTS=-O0 DEBUG=-g install
```

Contributing#

- Report Bugs and Contribute Code
- QNX Best Practices
- QNX Coding Standard