Libmnl: minimalistic library for Netlink developers

Pablo Neira Ayuso
<pablo@netfilter.org> Netfilter Project

Related work

- Libnetlink: old library for the iproute-tools.
- Libnl
  - Since 2.0, it has been split into several libraries.
  - Object-oriented abstractions.
  - Elaborated callback workflow.
  - Provides list-based caching system.
  - Lots of helpers for rapid development (good for developers not familiar with netlink).
- Libnfnetlink: Not generic enough to be used for other netlink subsystems.

I like simple things!

"Simplify, simplify"

-- Henry David Thoureau. Walden (1854)

Less is better

KISS

Few is sufficient, a lot is too much!

Libmnl: what is it?

- Minimalistic library oriented to netlink-wise developers
  - You have to know how Netlink works: “Communicating between the kernel and user-space in Linux using Netlink sockets”, published in Software: Practise and Experience.
- There are a lot of common tasks in parsing, validating, constructing of both the Netlink header and TLVs that are repetitive and easy to get wrong.
- This library aims to provide simple helpers in common Netlink tasks.
- Available at: http://1984.lsi.us.es/git/ (will move to netfilter.org)

Features

- **Small**: ~30KB on x86
- **Simple**: avoids complexity and elaborated abstractions.
- **Easy to use**: provides helpers for socket handling, message building, validating, parsing, and sequence tracking.
- **Easy to re-use**: you can use the library to build your own abstraction layer on top of this library.
- **Decoupling**: the interdependency of the main bricks (helpers) that compose the library is reduced, eg. the library provides many helpers but the programmer is not forced to use them.
- Doxygen-based documentation.

Four helper sets

- **Socket helpers**: open/close socket, bind, set/get options
- **Netlink message helpers**: getters, putters, iterators and validators.
- **Netlink attribute helpers**: getters, putters, iterators, validators and parsers.
- **Callback helpers** (not attached to socket helpers).

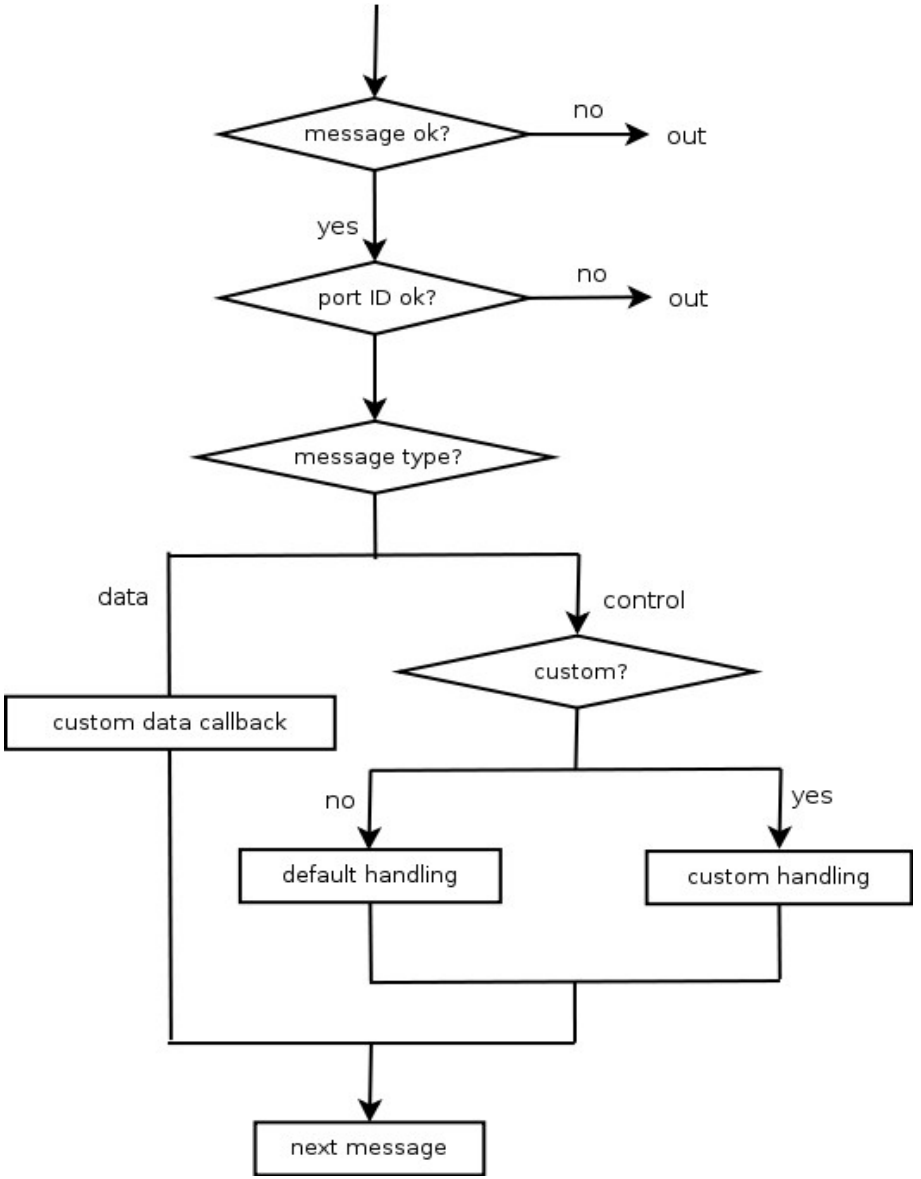
Workflow

1) open Netlink socket
2) bind it
3) while (1) {

        msgs = recv messages
        callback runqueue (msgs, control_cb1, ..., data_cb)
}

- No callback registering: You can re-use the same socket with different callbacks.
- Not forced to use recv() or recvmsg().

Callback workflow



Suggested custom data callback

1) Parse attributes

2) For each attribute

    1.1) check if it is valid (higher that *_MAX)

    1.2) validate attribute

    1.3) store it in some data structure (array?)

- No need to use an array as usual to store a pointer to the attribute.
- *Let's see some examples!*

Libmnl: minimalistic library for Netlink developers

Pablo Neira Ayuso
<pablo@netfilter.org> Netfilter Project