netlink - netlink library for go

The netlink package provides a simple netlink library for go. Netlink is the interface a user-space program in linux uses to communicate with the kernel. It can be used to add and remove interfaces, set ip addresses and routes, and configure ipsec. Netlink communication requires elevated privileges, so in most cases this code needs to be run as root. Since low-level netlink messages are inscrutable at best, the library attempts to provide an api that is loosely modeled on the CLI provided by iproute2. Actions like `ip link add` will be accomplished via a similarly named function like AddLink(). This library began its life as a fork of the netlink functionality in docker/libcontainer but was heavily rewritten to improve testability, performance, and to add new functionality like ipsec xfrm handling.

Local Build and Test

You can use go get command:

```
go get github.com/vishvananda/netlink
```

Testing dependencies:

```
go get github.com/vishvananda/netns
```

Testing (requires root):

```
sudo -E go test github.com/vishvananda/netlink
```

Examples

Add a new bridge and add eth1 into it:

```
package main

import (
    "fmt"
    "github.com/vishvananda/netlink"
)

func main() {
    la := netlink.NewLinkAttrs()
    la.Name = "foo"
    mybridge := &netlink.Bridge{LinkAttrs: la}
    err := netlink.LinkAdd(mybridge)
    if err != nil  {
        fmt.Printf("could not add %s: %v\n", la.Name, err)
    }
    eth1, _ := netlink.LinkByName("eth1")
    netlink.LinkSetMaster(eth1, mybridge)
}
```

Note `NewLinkAttrs` constructor, it sets default values in structure. For now it sets only `TxQLen` to `-1`, so kernel will set default by itself. If you're using simple initialization(`LinkAttrs{Name: "foo"}`) `TxQLen` will be set to `0` unless you specify it like `LinkAttrs{Name: "foo", TxQLen: 1000}`.

Add a new ip address to loopback:

```
package main

import (
    "github.com/vishvananda/netlink"
)

func main() {
    lo, _ := netlink.LinkByName("lo")
    addr, _ := netlink.ParseAddr("169.254.169.254/32")
    netlink.AddrAdd(lo, addr)
}
```

Future Work

Many pieces of netlink are not yet fully supported in the high-level interface. Aspects of virtually all of the high-level objects don't exist. Many of the underlying primitives are there, so its a matter of putting the right fields into the high-level objects and making sure that they are serialized and deserialized correctly in the Add and List methods.

There are also a few pieces of low level netlink functionality that still need to be implemented. Routing rules are not in place and some of the more advanced link types. Hopefully there is decent structure and testing in place to make these fairly straightforward to add.