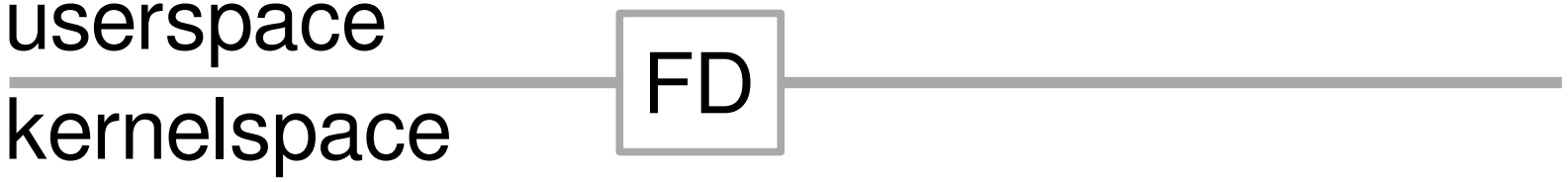
Golang, Linux kernel and the netlink family in between

golab.io 2018

Florian Lehner
github.com/florianl

Simple webserver

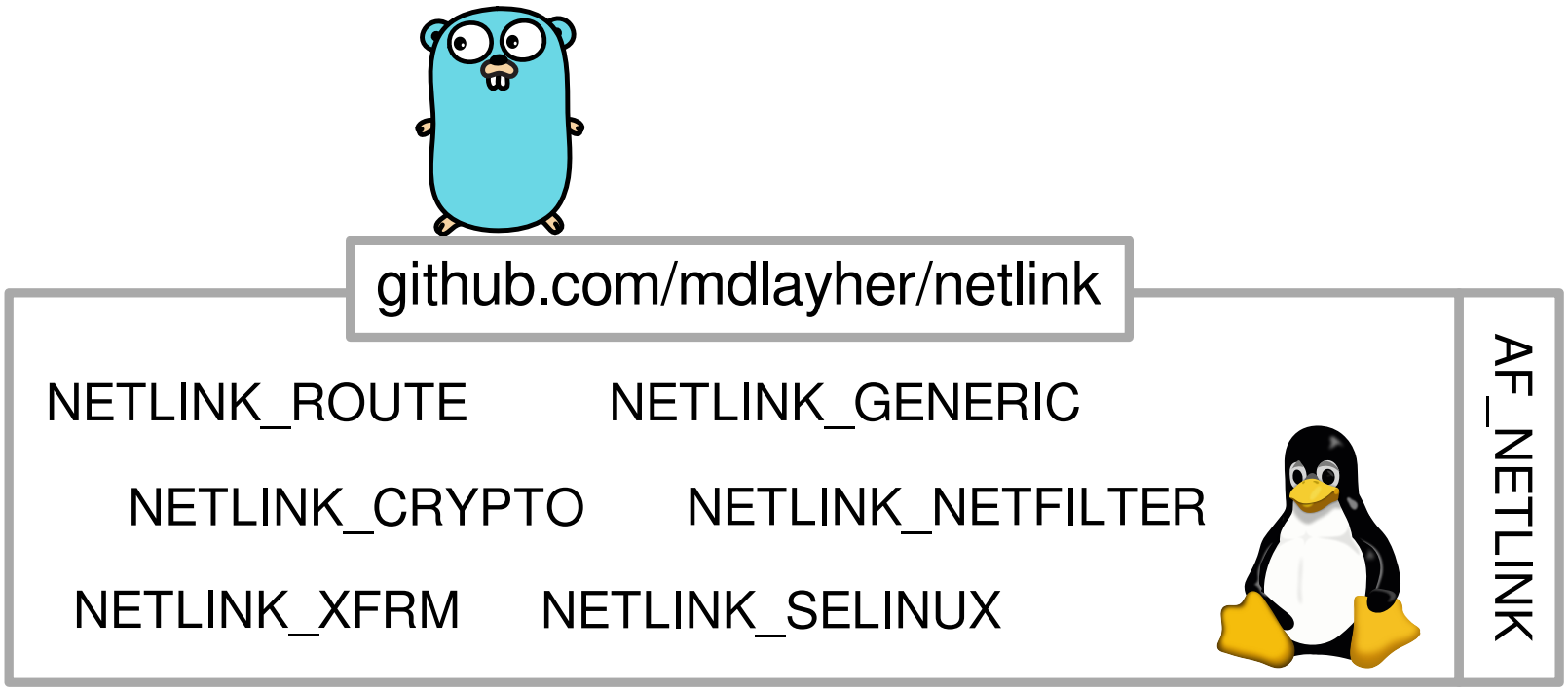
```
package main

import (
    "fmt"
    "log"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "We ♥ golang")
}

func main() {
    http.HandleFunc("/", handler)
    log.Fatal(http.ListenAndServe(":8080", nil))
}
```



Netlink

```
Netlink is used to transfer information between the kernel and userspace processes.
It consists of a standard sockets-based interface for user space processes.
```

by NETLINK(7)



Gopher by Takuya Ueda

Tux by Larry Ewing

Netlink - Header

Linux kernel implementation

```
struct nlmsghdr {
    __u32 nlmsg_len;    /* Length of message including header */
    __u16 nlmsg_type;   /* Type of message content */
    __u16 nlmsg_flags;  /* Additional flags */
    __u32 nlmsg_seq;    /* Sequence number */
    __u32 nlmsg_pid;    /* Sender port ID */
};
```

by include/uapi/linux/netlink.h

Golang representation

```
type Header struct {
    Length uint32
    Type HeaderType
    Flags HeaderFlags
    Sequence uint32
    PID uint32
}
```

by github.com/mdlayher/netlink

Netlink - Attribute

Linux kernel implementation

```
struct nlattr {
    __u16           nla_len;
    __u16           nla_type;
};
```

by include/uapi/linux/netlink.h

Golang representation

```
type Attribute struct {
    Length uint16
    Type uint16
    Data []byte
}
```

by github.com/mdlayher/netlink

Userspace implementations in golang

- show / manipulate traffic control settings

github.com/ema/qdisc

- taskstats interface, for per-task, per-process, and cgroup statistics

github.com/mdlayher/taskstats

- IEEE 802.11 WiFi device actions and statistics

github.com/mdlayher/wifi

- WireGuard generic netlink interface

github.com/mdlayher/wireguardctrl

- nftables

github.com/google/nftables

NETLINK_NFLOG example

```
package main
import (
    "context"
    "fmt"
    "time"
    "github.com/florianl/go-nflog"
    "golang.org/x/sys/unix"
)
func monitor(m nflog.Msg) int {
    fmt.Printf("%v\n", m)
    return 0
}
// Send outgoing pings to nflog group 100
// # sudo iptables -I OUTPUT -p icmp -j NFLOG --nflog-group 100
func main() {
    nf, err := nflog.Open()
    if err != nil {
        fmt.Println("could not open nflog socket:", err); return
    }
    defer nf.Close()
    ctx, cancel := context.WithTimeout(context.Background(), 5*time.Second); defer cancel()
    nf.Register(ctx, unix.AF_INET, 100, nflog.NfUlnlCopyPacket, monitor)
    select {case <-ctx.Done():}
}
```

Conclusion

- no Cgo required to communicate with the Linux kernel
- get data direct from the Linux kernel
- help to implement more netlink family protocols

Thank you

Florian Lehner
github.com/florianl