



# Linux Netlink as an IP Services Protocol

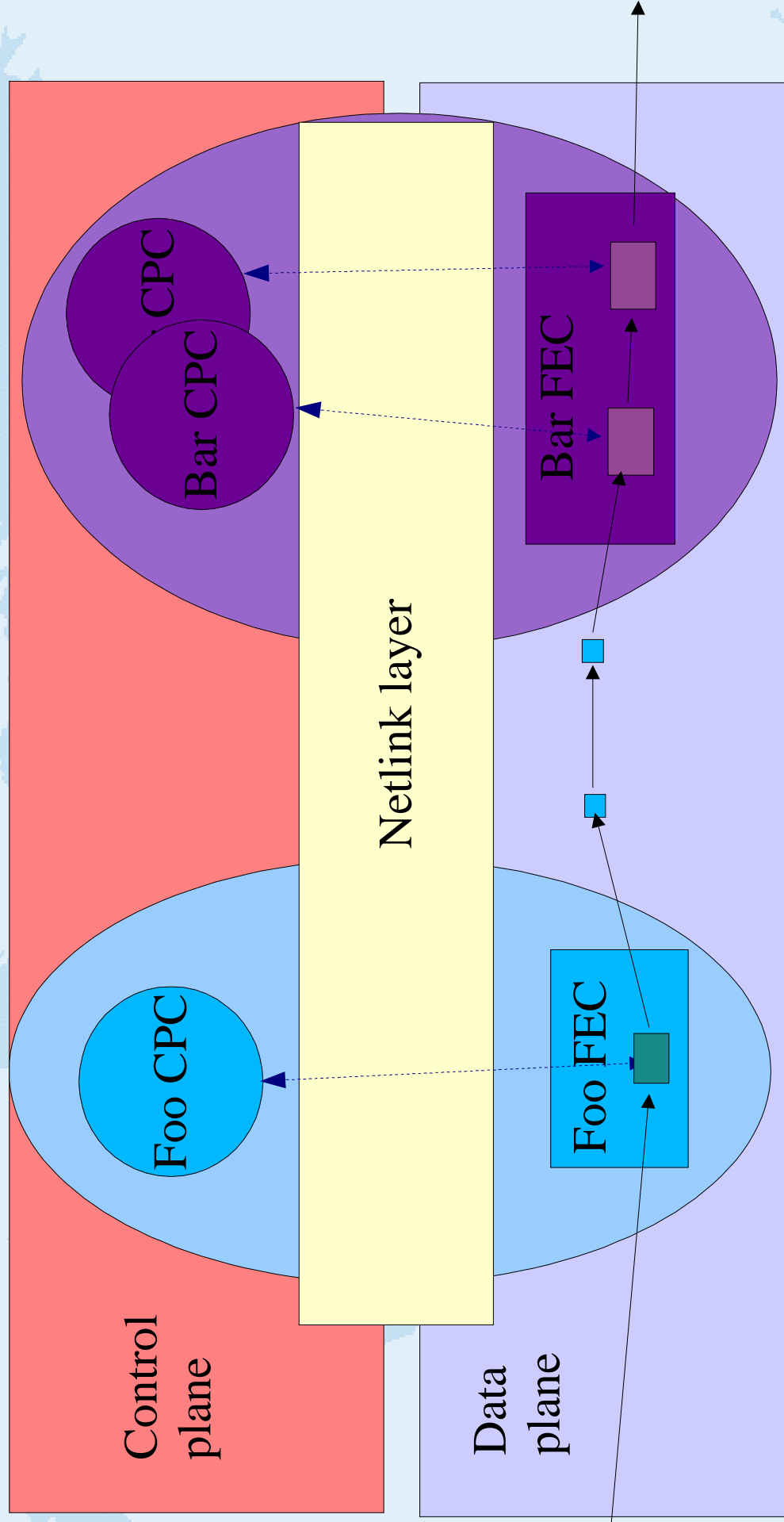
Jamal Hadi Salim --- Znyx Networks  
Hormuzd Khosravi --- Intel

IETF51, London, UK

# Background

- Initial base was BSD route socket API
- Evolution of Linux netlink (A. Kuznetsov, kernel 2.1)
  - Well defined *wire protocol* instead of just socket API
  - Non exclusive to IPV4 forwarding
  - Infact extended usage beyond IP
    - IPCs, Inter-kernel messaging, etc
    - Non-IP protocols (eg decnet)
    - Focus is on IP (to fit ForCES criteria)

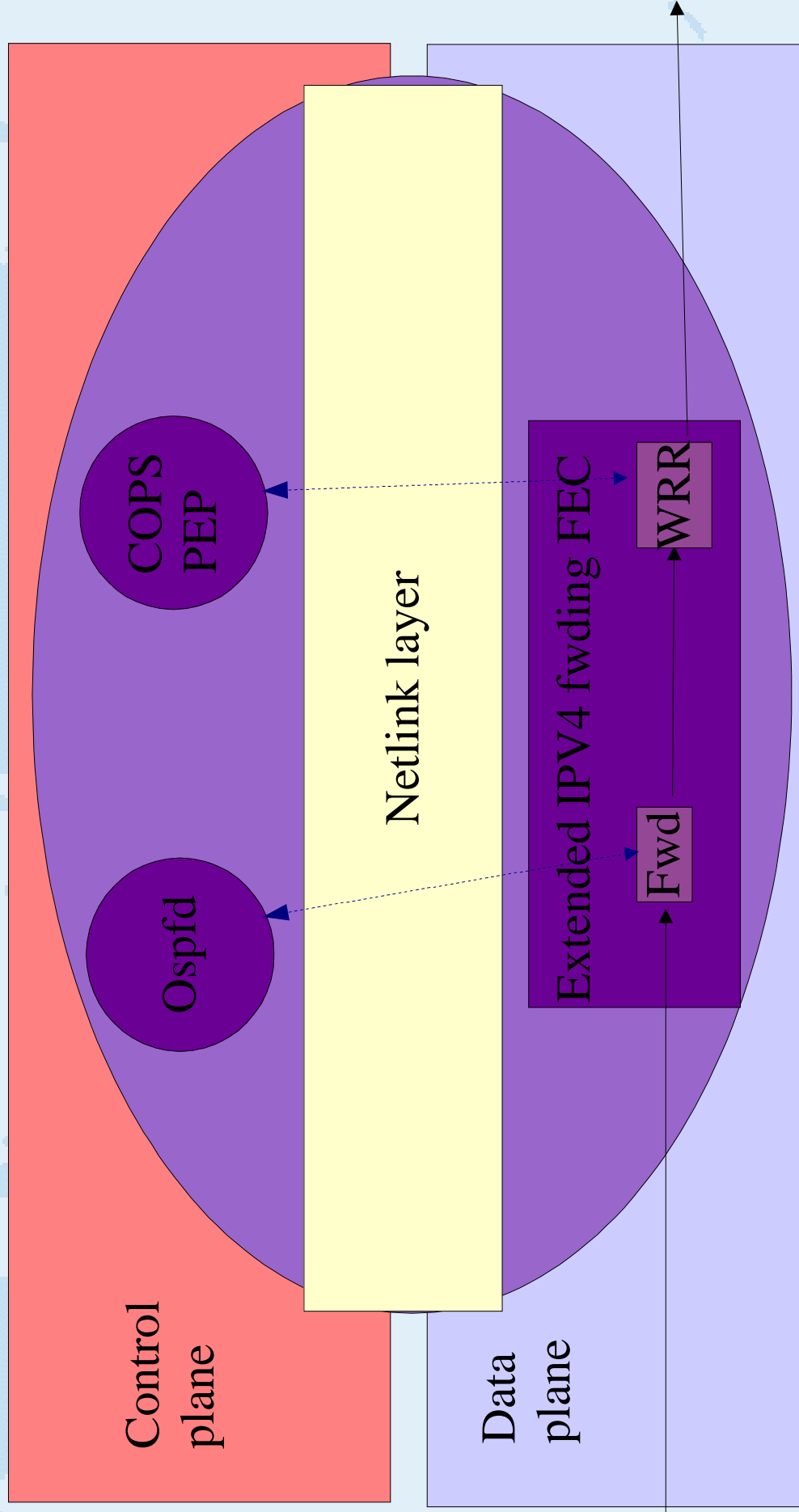
# IP Service Model



Legend:

- Service foo
- Service bar

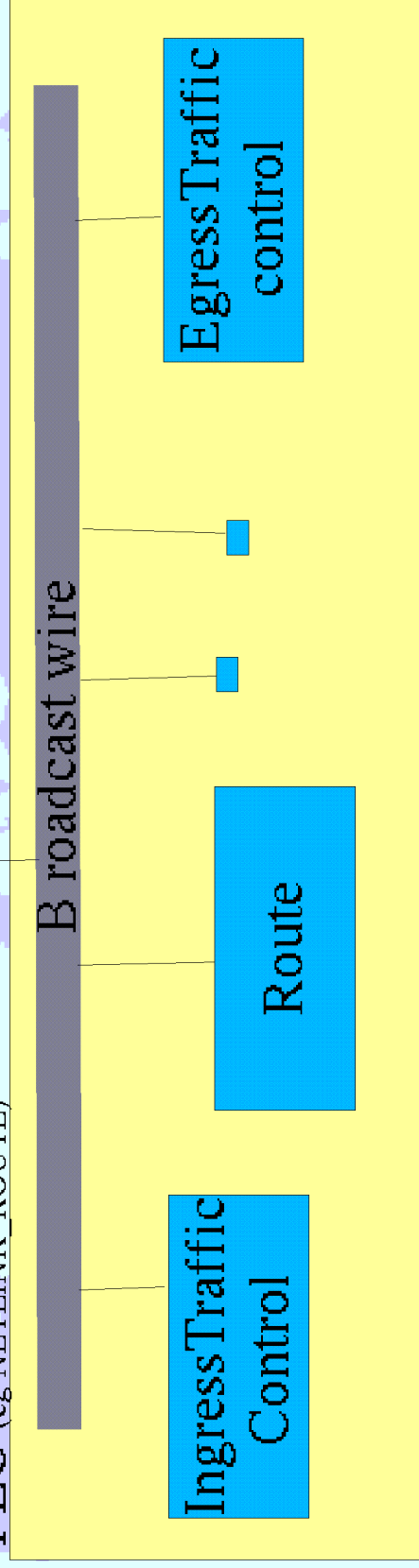
# Example: Extending IPv4 forwarding service



# CP ↔ FEC communication internals

Local CP or local proxy for remote CP

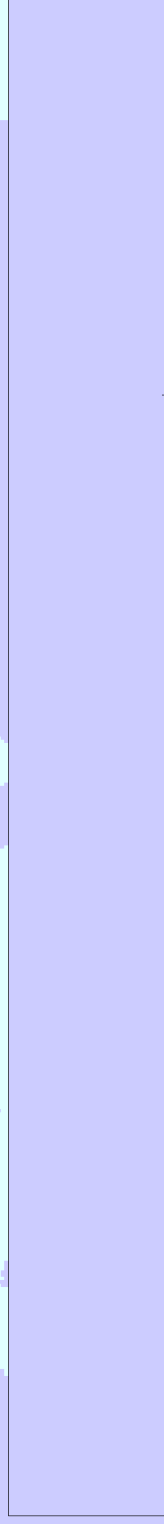
FEC (eg NETLINK\_ROUTE)



- *Connect* to FEC if not attached on the wire already
- *Send* packet on wire
  - Message could be discarded by wire if malformed etc
  - Message on the wire could be broadcast, multicast or unicast
  - Interested FEC or CP nodes pick specific message of interest

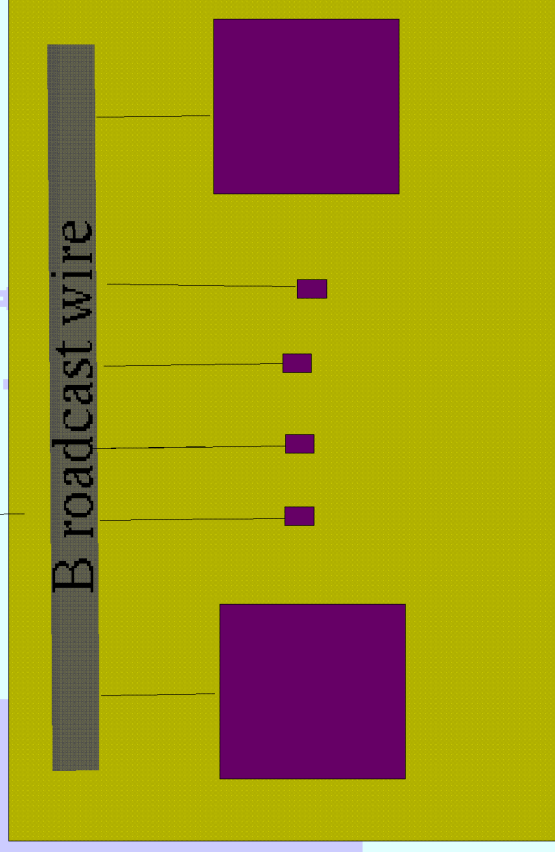
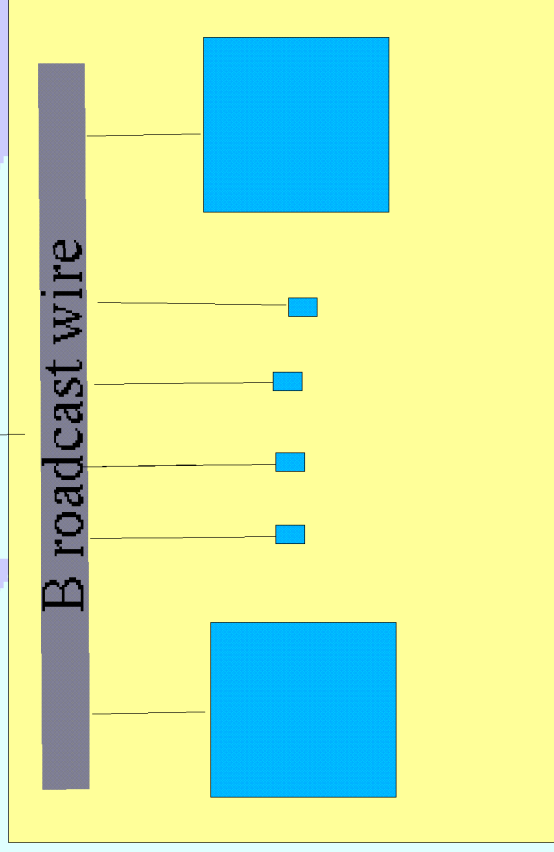
# CP $\leftrightarrow$ FEC communication internals: two services

Local CP or local proxy for remote CP



FEC for service 1

FEC for service 2



# Netlink Message

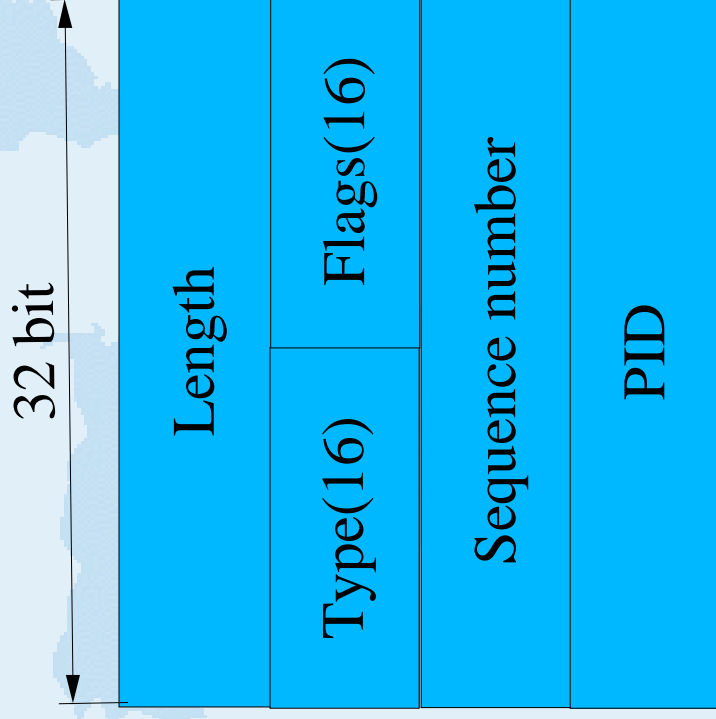
32 bits

Netlink message header

IP service template

IP service specific data (TLVs)

# Netlink header



Flags: (16 bits)

- further extrapolates msg content
- service command specific or generic

**[read draft]**

- generic flags include:

NLM\_F\_REQUEST "this is a request"

NLM\_F\_ACK: "I need an ACK to this msg"

NLM\_F\_ECHO: "echo this request back"

NLM\_F\_MULTI: "multipart message; one of"

Length: includes the header  
type: describes the contents

- service specific msg or one of generic netlink messages:

NLMSG\_NOOP,

NLMSG\_ERROR,

NLMSG\_OVERRUN,

and NLMSG\_DONE

seq number: a 32 bit seq number

PID: a process identifier.

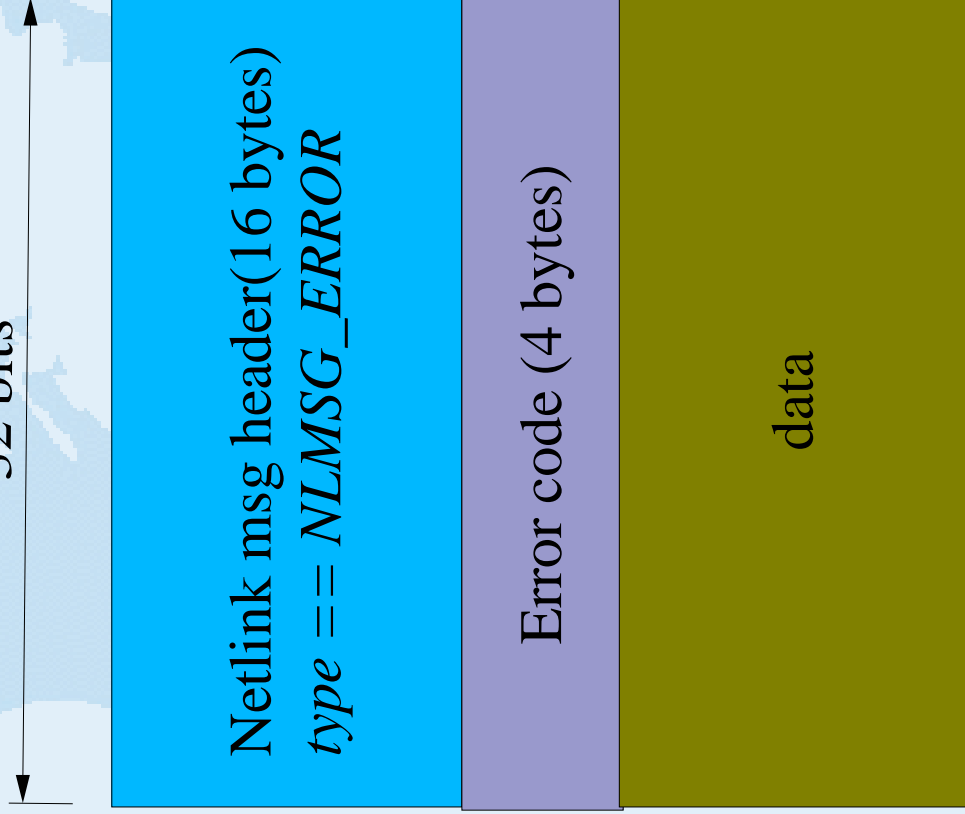
\* PID 0 is for CP to FE;

\*Any other identifies user space PID.



# Netlink ACKs and NACKs

32 bits



This is in response to a request to ACK message back

Error code of 0 indicates ACK  
data contains original netlink msghdr  
(useful for comparing seq numbers etc)

Error code of !0 indicates NACK

- data contains original netlink msghdr + original data sent (as received by NACKing end)

- Error is the E\* value (eg EINVAL)

# Sample Service template: IPv4 forwarding

Family(8)	Src length(8)	Dst length(8)	TOS(8)
TableID(8)	Protocol(8)	Scope(8)	Type(8)
Flags(32)			

Family: AF\_INET for IPv4 and AF\_INET6 for IPv6

src length: mask length of src IP address

dst length: mask for src IP address

TOS: the TOS field

TableID: the id for routing table being datafilled

protocol: "who" filled this entry (eg gated, zebra etc)

scope: validity of route in terms of hops (eg LOCAL, HOST, LINK etc)

type: the route type (eg UNICAST, MULTICAST, ANYCAST etc)

flags: additional descriptors (eg multipath equalizer route etc)

# IPV4 TLVS

RTA\_DST: Dst IP address

RTA\_SRC: SRC IP address

RTA\_OIF: egress interface (where nh is located)

RTA\_IIF: ingress interface

RTA\_GATEWAY: This is a gateway route

etc (look at `linux_src/include/linux/rtnetlink.h`)

# FEC ↔ CPC protocol building

- Mechanisms provided for either reliable, unreliable and/or HA services
- Reliable protocol
  - Make use of *ACKing*
  - Add timer for retransmits
- *echos*
  - create heartbeat protocol to specific FEC or FEC node

# Building sample service: basic

## IPv4 routing/forwarding

- \* Connect to service FEC NETLINK\_ROUTE by socket() addressing
- \* Bind to listen to events related to any mods to the IPV4 FIB via joining multicast group RTMGRP\_IPV4\_ROUTE
- \* Bind to listen to port events (multicast group RTMGRP\_LINK)
- \* Query state of FIB and ports and synchronize
- \* Register to FE to receive service specific packets
- \* Start sending service specific commands (ADD/DEL/GET etc)
- \* receive responses (if needed)
- \* Synchronizing on any asynchronous events (eg link down/up) etc

# References

- ▶ Linux kernel 2.4.7 source
- ▶ Netlink man pages (incomplete) A kleen
- ▶ BSD Route Sockets API W Richard Stevens  
vol2