The Road to IPv6

Simon Wilkinson (Your File System Ltd)

Contact Solutions
Service Line x25656

University of Southampton

PR_B32_C3049_M01

⚠ CAUTION

Legacy IP Only

This product does not support the current generation of Internet Protocol, IPv6.

PLIES

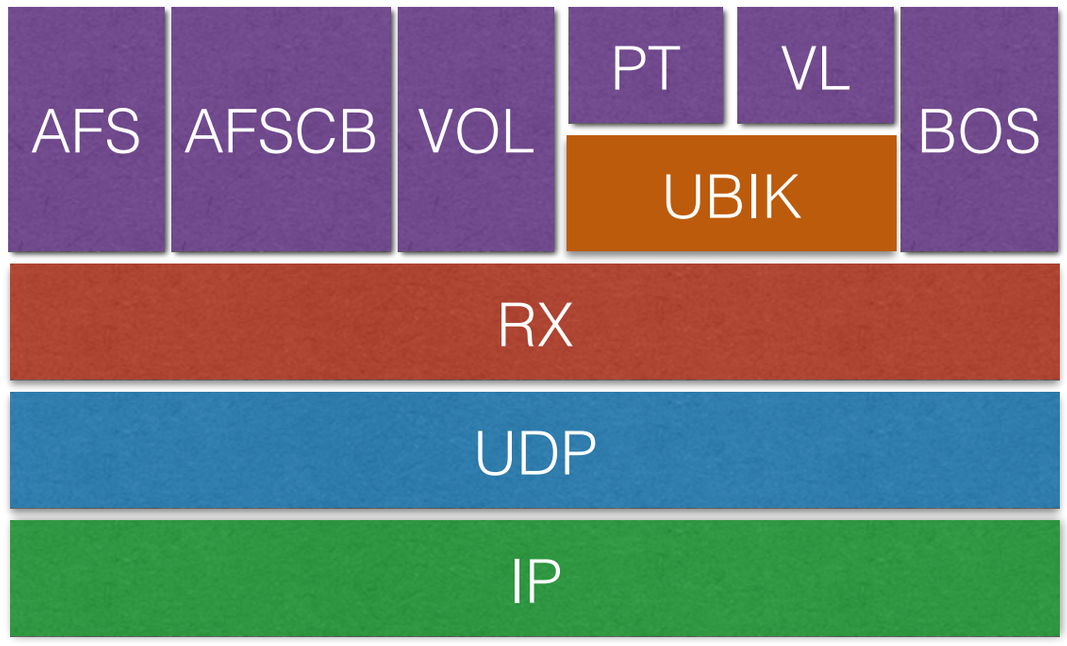
ta.co.uk

bizhub C360

The Problem

- Protocol issues
- Configuration Issues
- Implementation issues

Protocol Stack



Here be dragons …

- kaserver (kauth)
- NFS translator (rmtsys)
- Backup Service (budb, backmon, bumon, butc)

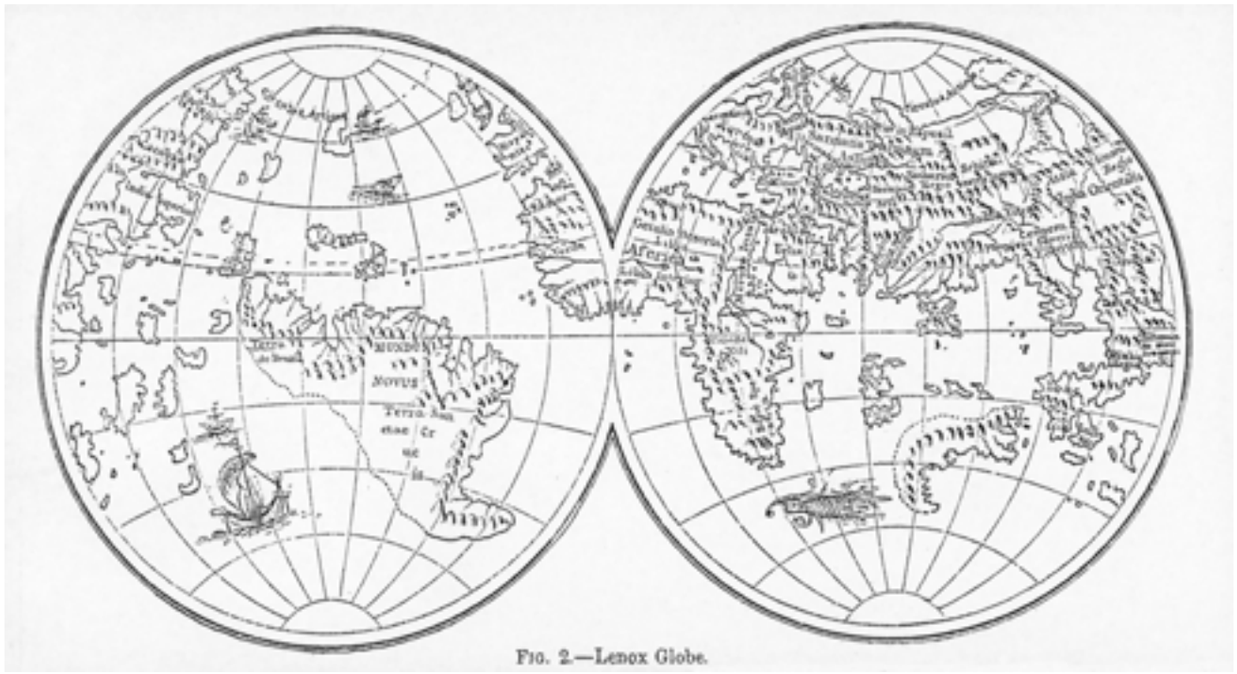
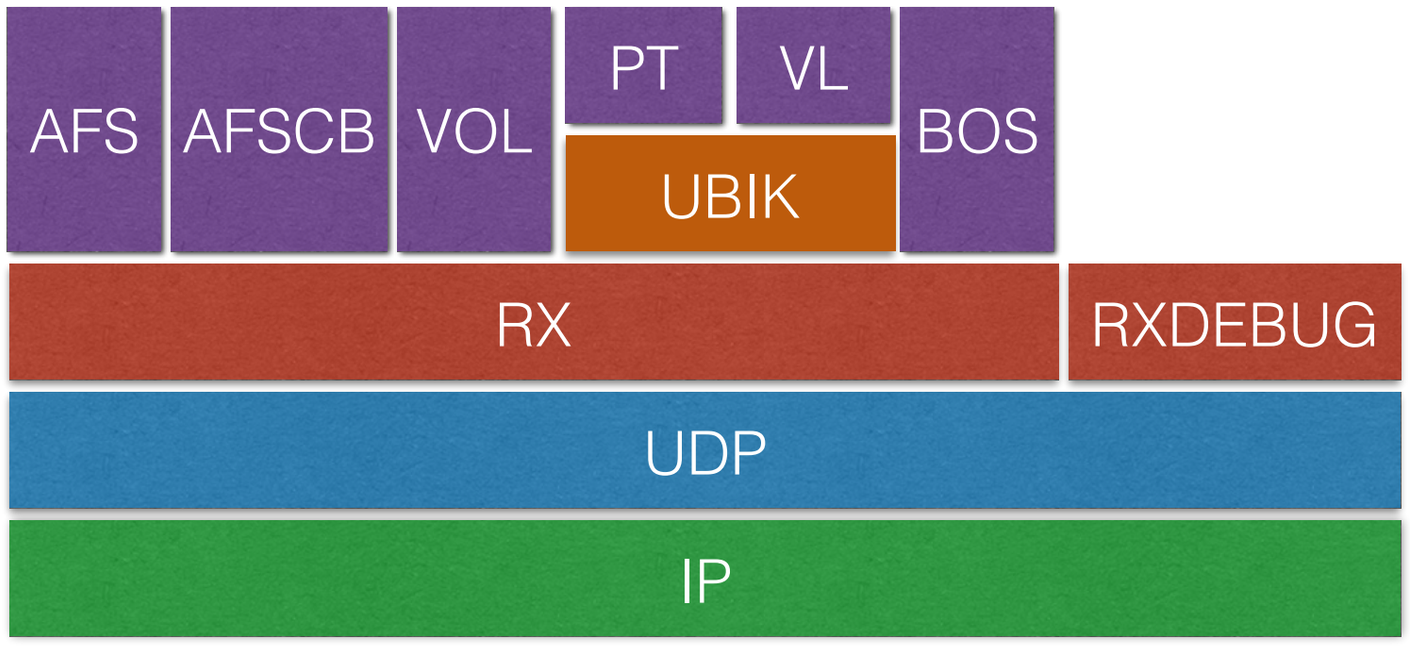


FIG. 2.—Lenox Globe.

Rx Protocol

- No changes required to core protocol
- All address handling performed within UDP and IP

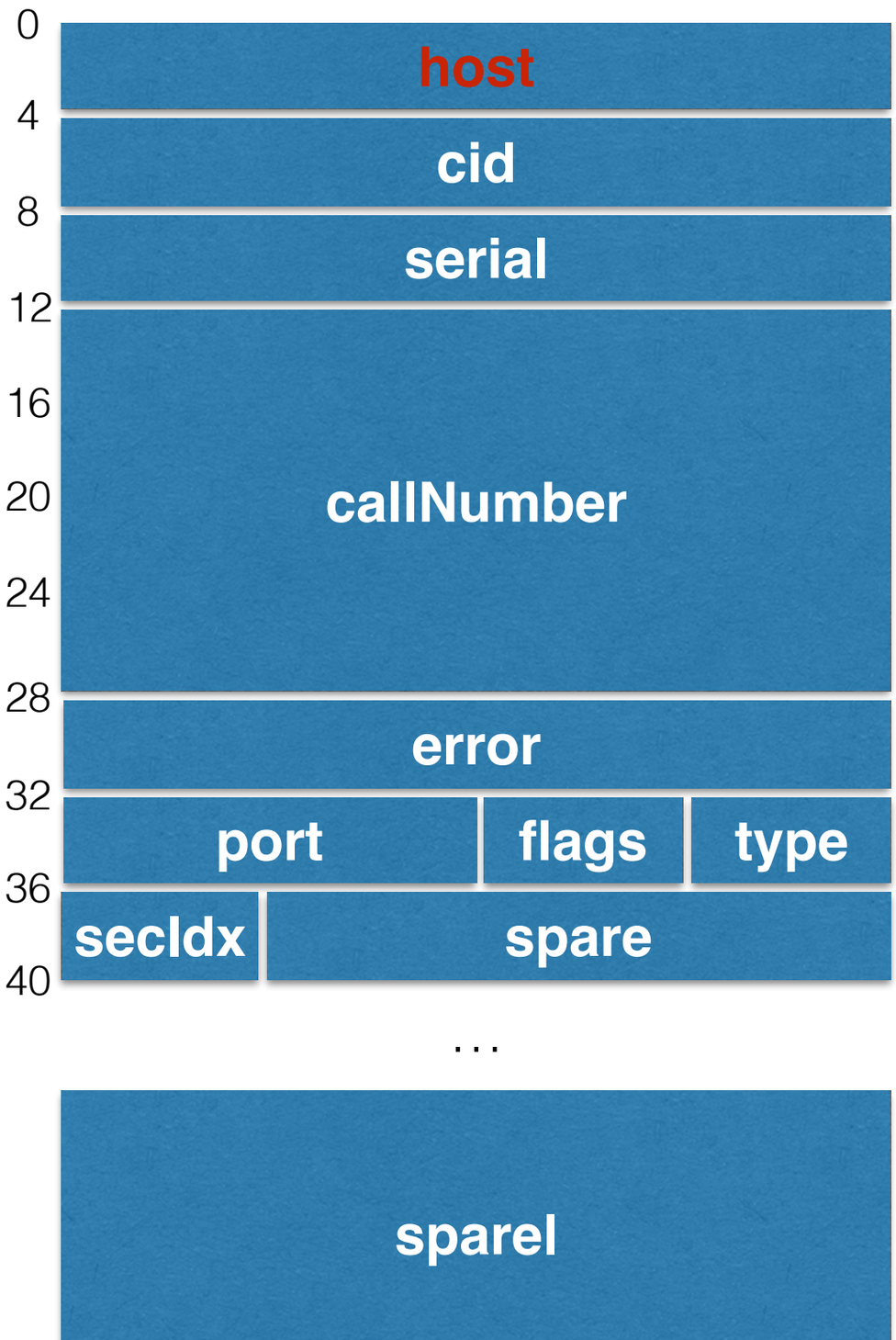
RX Debug Packets

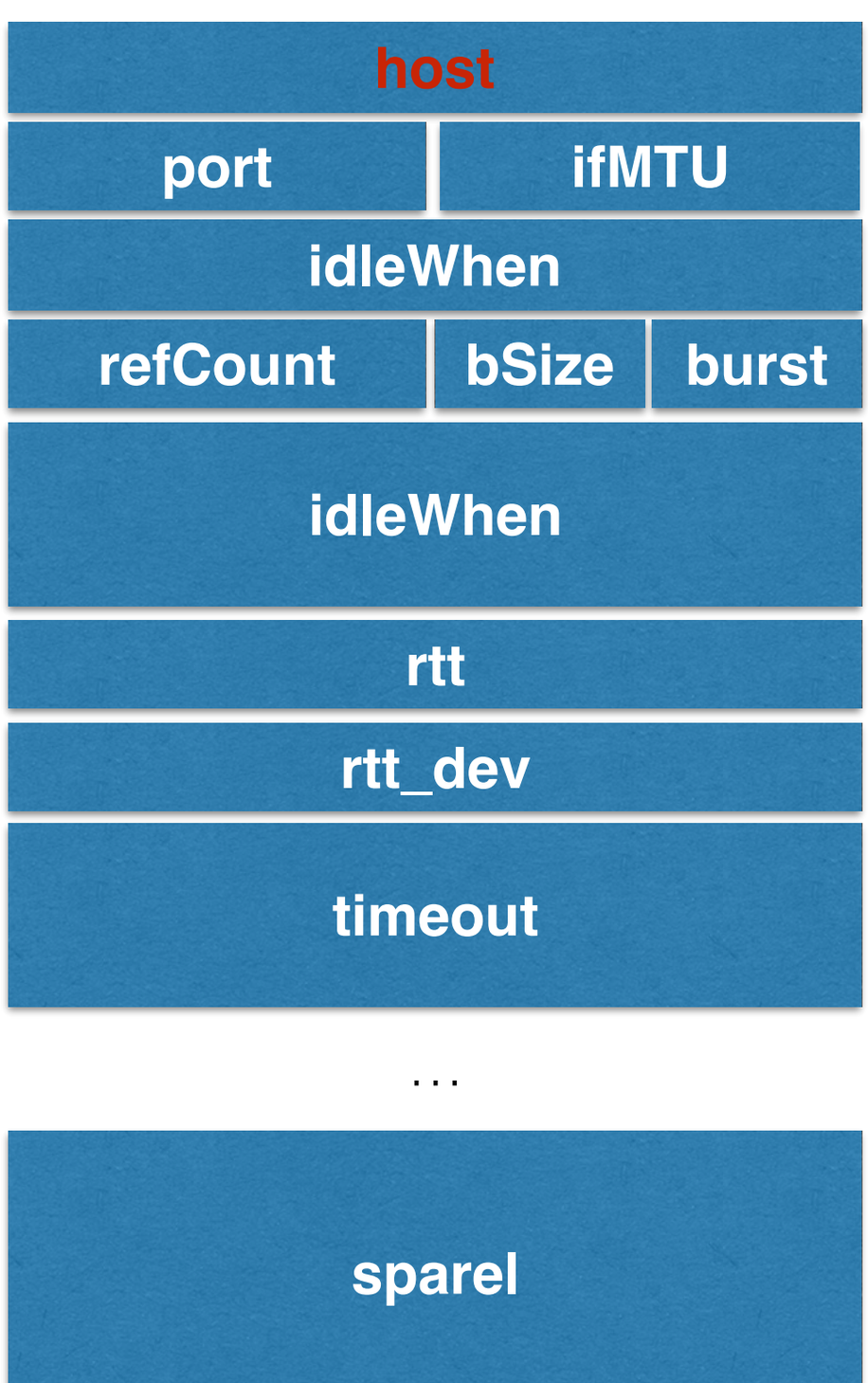


RX DEBUG

- Four packet types can be requested
  - 0x01 - Debug Statistics
  - 0x02 - Connections
  - 0x03 - Peers
  - 0x04 - RX Statistics

Connection and Peers





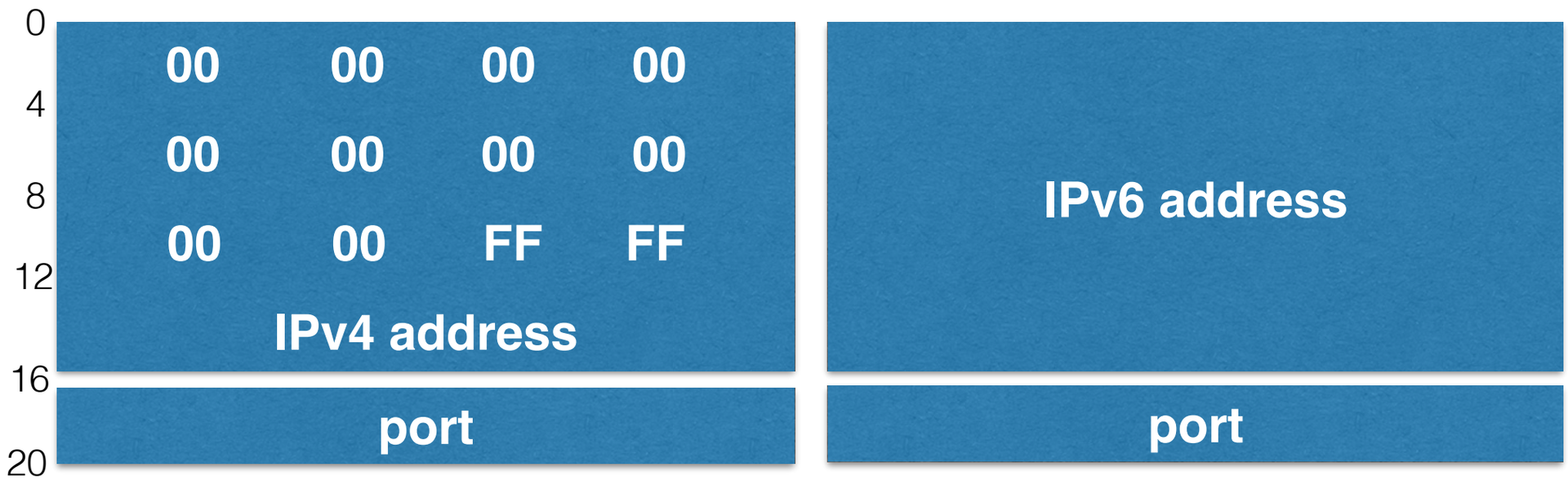
What’s an endpoint?

- Currently - IPv4 address

- Sometimes with port

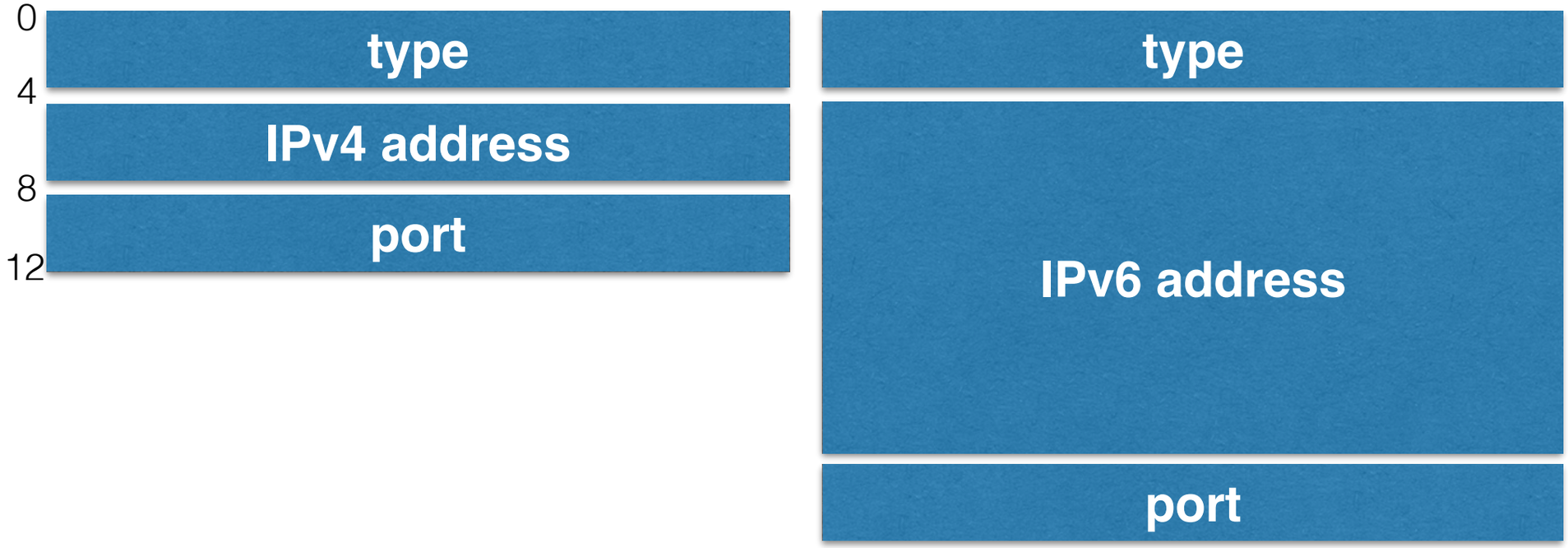
IPv6/v4 endpoints

- Use v4 mapped addresses



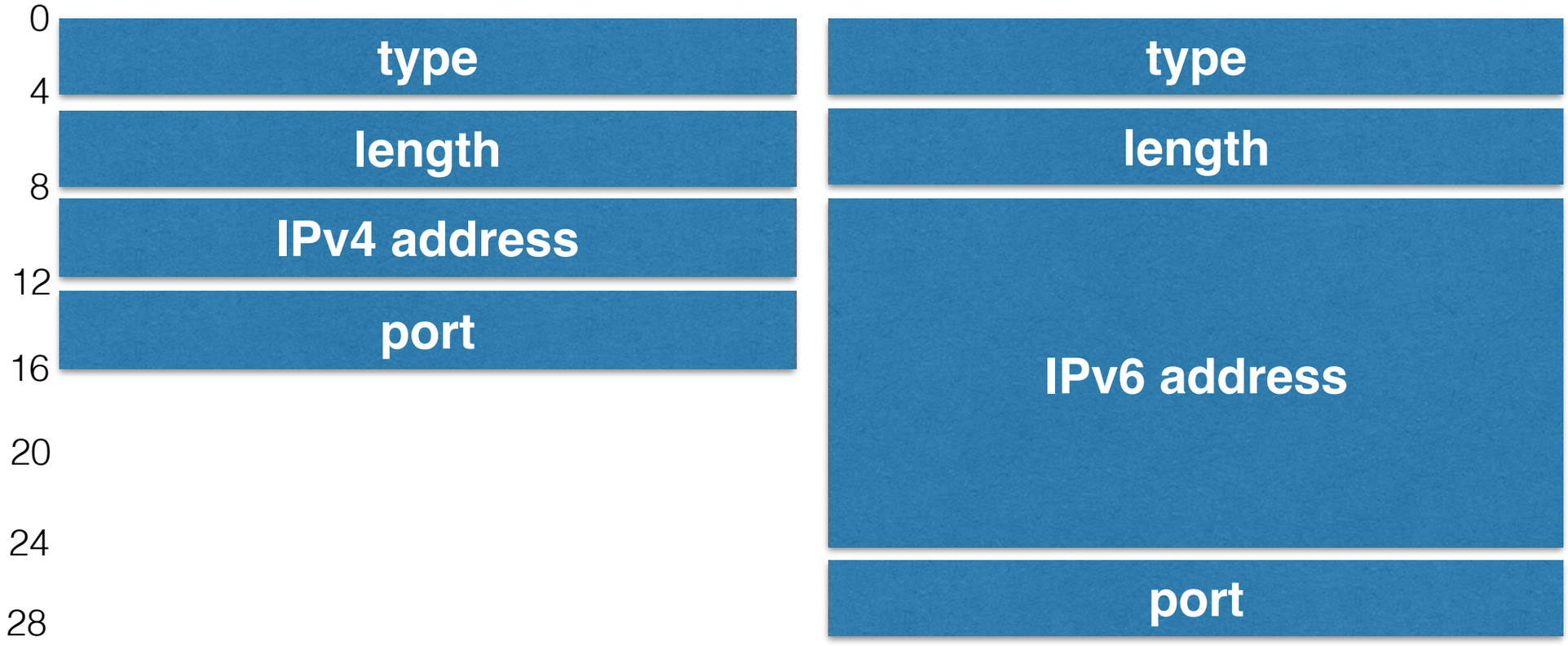
IPv6 / v4 endpoints

- Use a discriminator



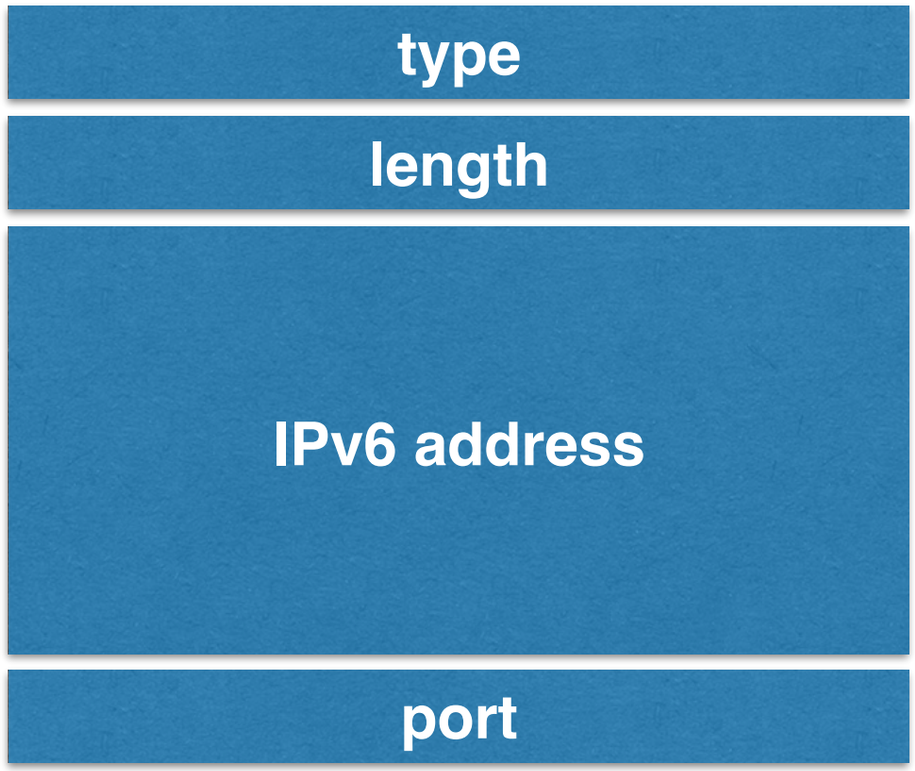
IPv6 / v4 endpoints

- Use an extended union



What is an endpoint?

```
ext-union endpoint
    switch (afs_int32 type) {
      case ENDPOINT_UDP_IPV4:
        afs_int32 host;
        short port;
      case ENDPOINT_UDP_IPV6:
        opaque addr[16];
        short port;
    }
```



Codepoint allocation

- Who allocates new RX debug packet code points?

- Can existing hosts deal with receiving requests for new debug packet types?

AFS

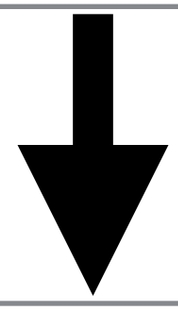
- Much of the AFS fileserver protocol is endpoint agnostic

- Only RXAFS_FlushCPS is affected

RXAFS_FlushCPS

```
typedef afs_int32 IPAddrs<FLUSHMAX>;

RXAFS_FlushCPS(IN  ViceIds *IdsArray, IPAddrs *AddrsArray, afs_int32 spare1,
               OUT afs_int32 *spare2, afs_int32 *spare3) = 162;
```



```
typedef endpoint Endpoints<FLUSHMAX>;

RXAFS_FlushCPS6(IN  ViceIds *IdsArray, Endpoints *AddrsArray) = XXX;
```

AFSCB

- Clients report all interface addresses to the server
- Multiple RPCs provide this functionality

```
RXAFSCB_TellMeAboutYourself
RXAFSCB_WhoAreYou
RXAFSCB_InitCallBackState2
```

- Addresses are irrelevant in today's NAT'd world

AFSCB - Debugging

- Multiple RPCs provide debugging output containing IPv4 addresses

```
RXAFSCB_GetServerPrefs
RXAFSCB_GetCellServDB
RXAFSCB_GetCellByNum
```

- Not necessary for normal cache manager operation

VOL

- VOL is mainly address agnostic, but forwarding is an issue

```
AFSVol_SetForwarding
AFSVol_Forward
AFSVol_ForwardMultiple
```

- While AFSVol_SetForwarding is still called, it has no effect on the server

Vol - Forwarding

- Currently the client tells the server the IPv4 address to move the volume to
- Does the client know best?
- What if the client and server have different views of the world?
- Not as simple as just swapping IPv4 addresses for v6 ones

ubik restrictions

- All ubik servers must be able to contact all other ubik servers

- All clients must be able to contact all ubik servers which can become sync sites

- The ubik server with the lowest address gets the deciding vote at tied elections

- Need a form of ranking that works across v4 and v6

ubik - VOTE

- Only one service RPC with address dependencies

```
VOTE_GetSyncSite
```

- A number of debugging RPCs to consider

```
VOTE_Debug
VOTE_SDebug
VOTE_XDebug
VOTE_XSDebug
```

ubik - DISK

- One RPC which is used to find out all of the addresses for a given Ubik server

```
DISK_UpdateInterfaceAddr
```

PT

- In general, the PT is address agnostic
- Exception to this is IP-ACLs which cause a number of issues

```
PT_GetHostCPS
```

PT - IP ACLs

- An IP ACL is just a user with a special name (192.168.0.1 rather than sxw)
- IPv6 addresses can be handled in similar ways
- We'd have to canonicalise
- And deal with wildcards

VL

- Protocol-wise, VL is by far the most affected

- Not helped by incomplete conversion between different RPC families O -> N -> U

VL - O to N

GetEntryById	GetEntryByIdN
GetEntryByName	GetEntryByNameN
CreateEntry	CreateEntryN
ReplaceEntry	ReplaceEntryN
ListEntry	ListEntryN
LinkedList	LinkedListN
ListAttributes	ListAttributesN
ChangeAddr	
UpdateEntry	ListAttributesN2
UpdateEntryByName	

VL - N to U

GetEntryByIdN
GetEntryByNameN
CreateEntryN
ReplaceEntryN
ListEntryN
LinkedListN
ListAttributesN
ListAttributesN2

GetEntryByNameU

GetAddrsU
RegisterAddrs

vldb

- Storing large numbers of IPv6 addresses requires a new vldb format
- Small number (2) can be stored in existing format
- Don’t register temporary addresses!

Migration Issues

- How do you expose IPv6-only volumes to old clients?
- How do you safely migrate volumes to IPv6 only hosts?

Configuration

- CellServDB

- NetRestrict

CellServDB

- New CellServDB format designed at Pittsburgh Hackathon

```
[core]
thiscell = andrew.cmu.edu
use_dns = yes

[cells]
andrew.edu = {
    description = "Project Andrew - CMU"
    vlserver = tcp/128.2.10.2
    ptserver = udp/128.2.10.11
    dbserver = 128.1.10.7
    dbserver = FF00::128.2.10.28
    dbserver = db3.andrew.cmu.edu
    use_dns = yes
}

[ptserver]
servers = {
    vice2 = {
        address = 128.2.10.2:7002
        priority = 2
    }
}
```

```
[fileserver]
dbservers = {
    vlserver = vice7
}

[rank]
# syntax for host addresses:
# [proto/]host[/mask][:port]
128.2.10.2 = 2000
tcp/128.2.10.11 = 9000
128.2.10.2 = 2000
128.2.10.12:7003 = 1500
128.2.172/22 = 100
```

Implementation

- Anywhere there is an int for an address needs rewritten
- All hash functions taking addresses have to be reworked
- Everywhere we print / log / audit an IP address needs updated

Implementation

- RX implementation needs to be dual stack
- ICMP handling needs to work for IPv6
- The afsconf package needs to handle IPv6 cell information
- New fs pioctls (for cell manipulation) have to be created
- The vldb ubik database must store IPv6 addresses

Implementation

- The host package needs to track IPv6 clients
- vos needs to be able to handle servers with multiple addresses
- ubik's multi-home support (and configuration) needs work

Happy Eyeballs

- If we have both IPv4 and IPv6 addresses for a service - which should we use?

- RFC6555 describes a solution for TCP

DNS6to4

- Allows v6-only clients to talk to v4-only servers
- But requires a name lookup
- What do we lookup?

PMTU Discovery

- IPv6 allows robust, platform independent, PMTU discovery

- How do we utilise that within RX?

Where YFS is today

- RX fully v6 capable
- Servers contactable over v6 addresses
- v6 registration for fileservers in testing
- v6 ubik voting still outstanding