HiperSockets in the Round a z/OS, z/VM, zLinux Perspective

Frank J. De Gilio

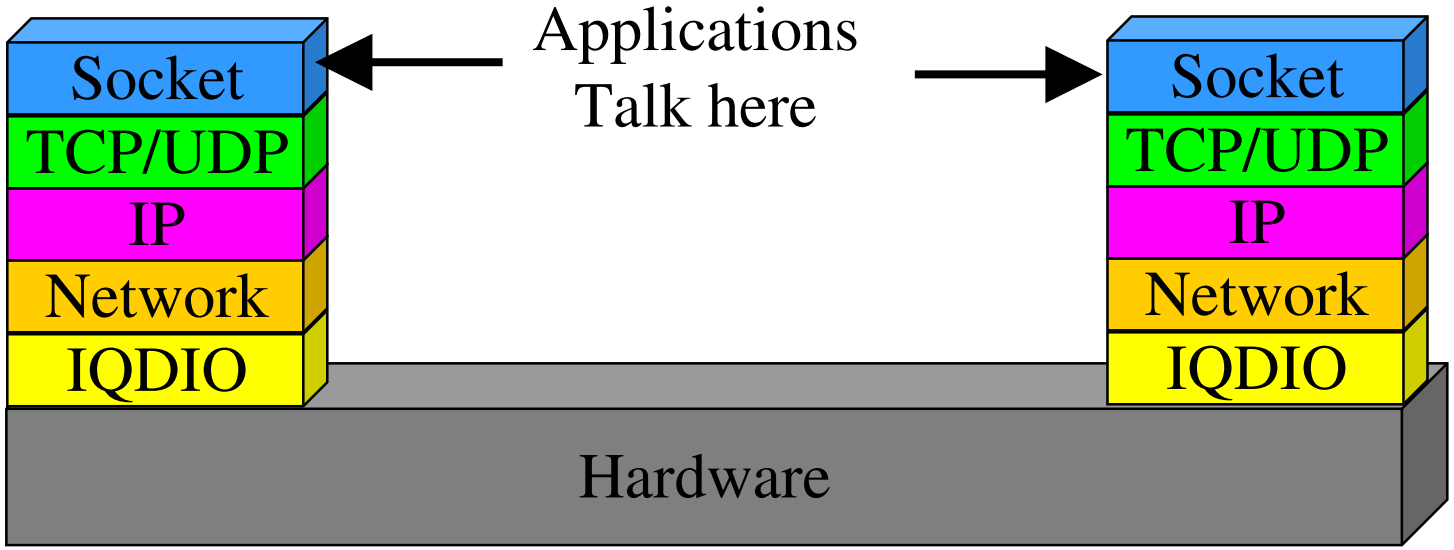
IBM Design Centers

degilio@us.ibm.com

HiperSockets – Just The Facts

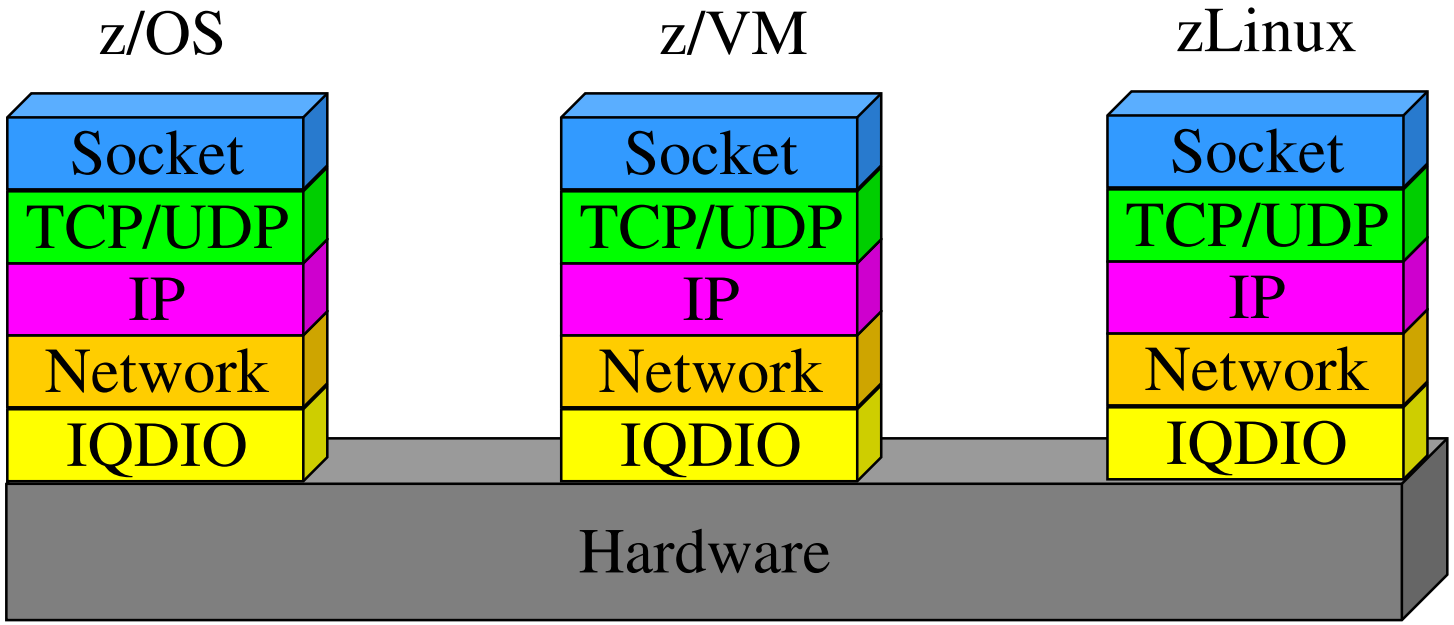
- HiperSockets = Internal Queued Direct IO
- Microcode maintained lookup table
- Three devices for each stack
  - Read Control
  - Write Control
  - Data Exchange
- 1024 Devices across all HiperSockets
- Supports Virtual IP Addressing and Dynamic Virtual IP Addressing

Cool HiperSocket Stack Picture



Maximum frame size 64K
(TCPIP MTU 56K)

All The Pretty OSes In A Row

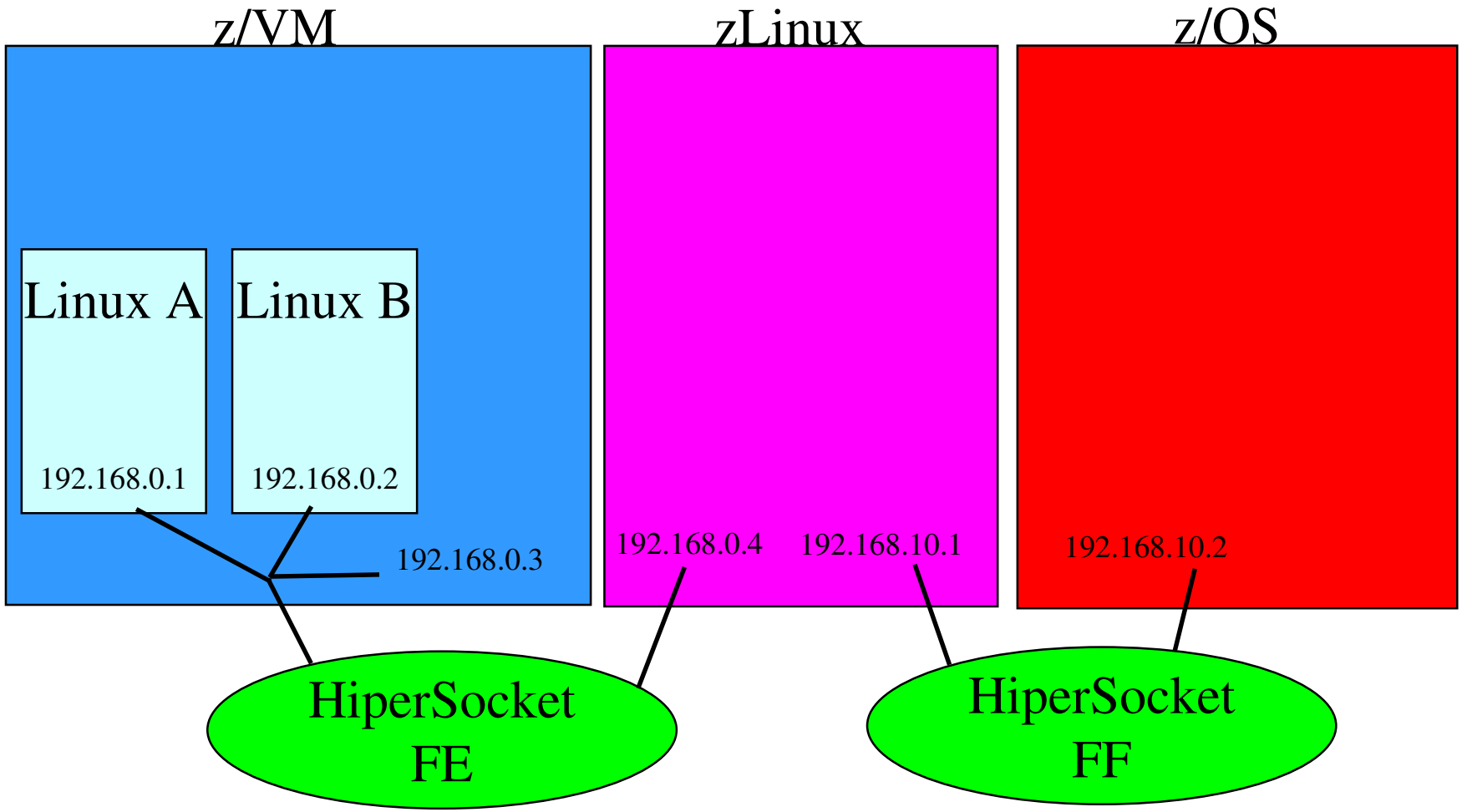


4 Virtual CHPIDs create 4 possible “virtual LANs”

Not bound to a Plex

Point to point routing

HiperSockets – a Network View



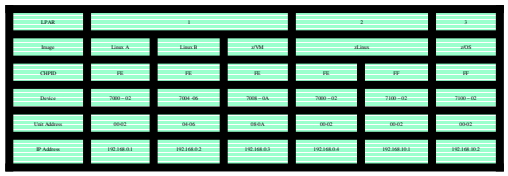
HiperSocket Tables

LPAR	1			2		3
Image	Linux A	Linux B	z/VM	zLinux		z/OS
CHPID	FE	FE	FE	FE	FF	FF
Device	7000 – 02	7004 -06	7008 – 0A	7000 – 02	7100 – 02	7100 – 02
Unit Address	00-02	04-06	08-0A	00-02	00-02	00-02
IP Address	192.168.0.1	192.168.0.2	192.168.0.3	192.168.0.4	192.168.10.1	192.168.10.2

Microcode at Work!!!

Linux

192.168.10.1



z/OS

192.168.10.2

Hardware Setup for HiperSockets

- Define CHPIDs (You have to use real device addresses)
- In HCD define:
  - CHPID
  - Type IQD
  - Mode SHR
  - Frame size
  - Control unit number & address range
- Define devices for the control unit
  - Type IQD
- Select operating system for devices

Setting up HiperSockets in z/OS

- Define CHPIDs

```
CHPID PATH=(FF),SHARED,PARTITION=((A1,A2,A3),….),
CNTLUNIT CUNUMBR=7000,PATH=(FF),UNIT=IQD
IODEVICE=(7000,016),CUNUMBR=7000,UNIT=IQD
```

- Modify the TCPIP Profile:

```
DEVICE IUTIQDFF MPCIPA NONRouter AUTORESTART

LINK HIPER IPAQIDIO IUTIQDFF
 HOME
 192.168.10.2 HIPER

GATEWAY
  192.168.10 =    HIPER      24576      0
 START IUTIQDFF
```

Setting Up HiperSockets in z/VM

- Set up CHPIDS

```
VARY ON CHPID FF
VARY ON 7000-704B
ATTACH 7000-7002 TO TCPIP AS 7000-7002
ATTACH 7004-7006 TO LINUXA AS 7004-7006
```

- Setup IP

```
DEVICE HIPERFF HIPERS 7000 PORTNAME HIPERFF
LINK HIPERFF QDIOIP HIPERFF
HOME
   192.168.10.1 HIPERFF
GATEWAY
   192     =    HIPERFF    1500    0.255.255.0   0.168.10.0
START HIPERFF
```

Setting Up HiperSockets on z/VM
(*Continued*)

- Set up Directory Information
- For User TCPIP

```
DEDICATE 7000 7000
DEDICATE 7001 7001
DEDICATE 7002 7002
```

- For User LinuxA

```
DEDICATE 7004 7004
DEDICATE 7005 7005
DEDICATE 7006 7006
```

Setting Up HiperSockets on Linux

- In modules.conf put:

```
Alias hsi1 qeth
```

- In chandev put:

```
qeth1,0x7004,0x7005,0x7006,4096
```

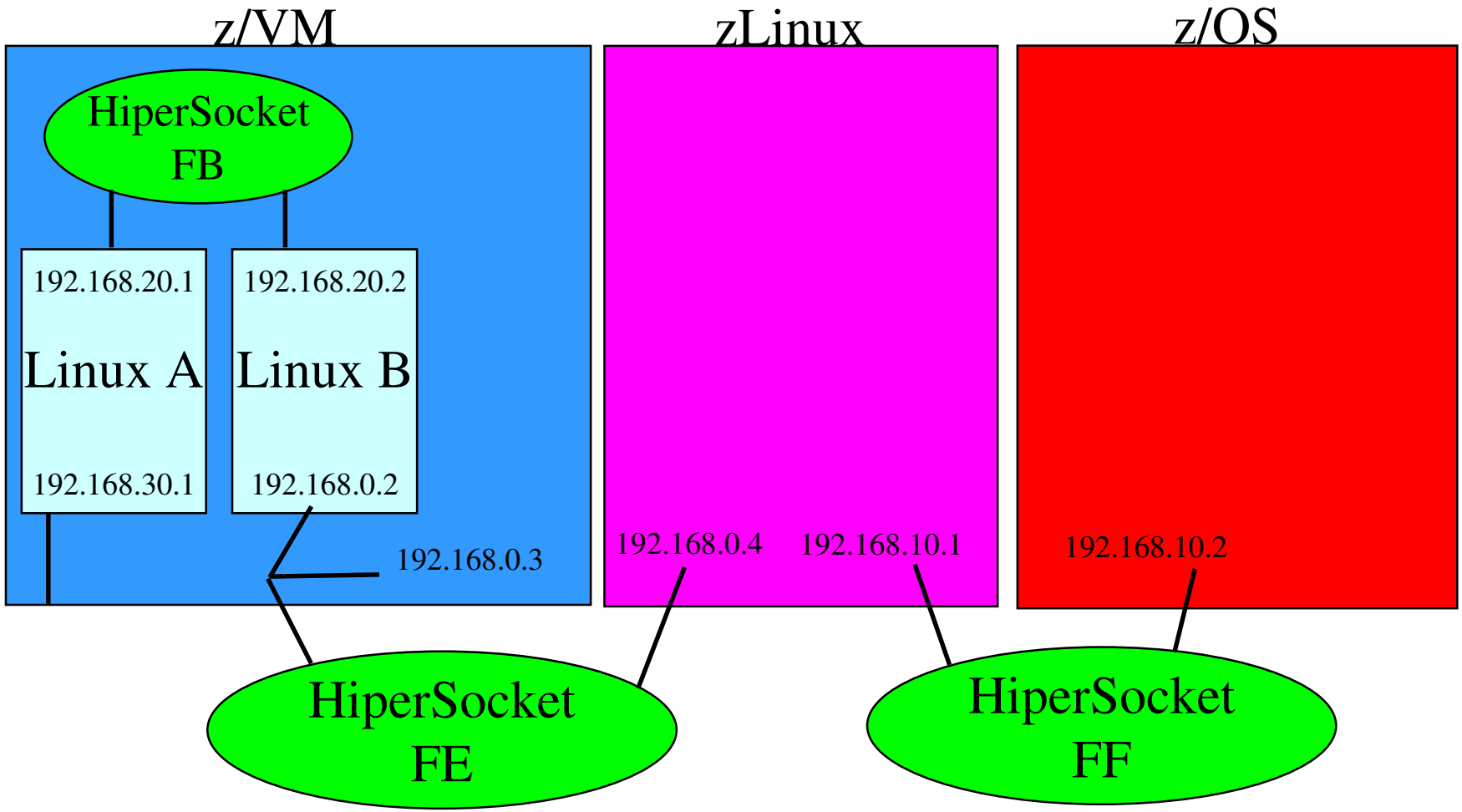
- In rc.config put:

```
NETCONFIG="_0_1"
IPADDR_1="192.168.10.3"
NETDEV_1="hsi1"
IFCONFIG_1="192.168.10.3 broadcast 192.168.10.3
netmask 255.255.255.0 mtu 4096 up"
```

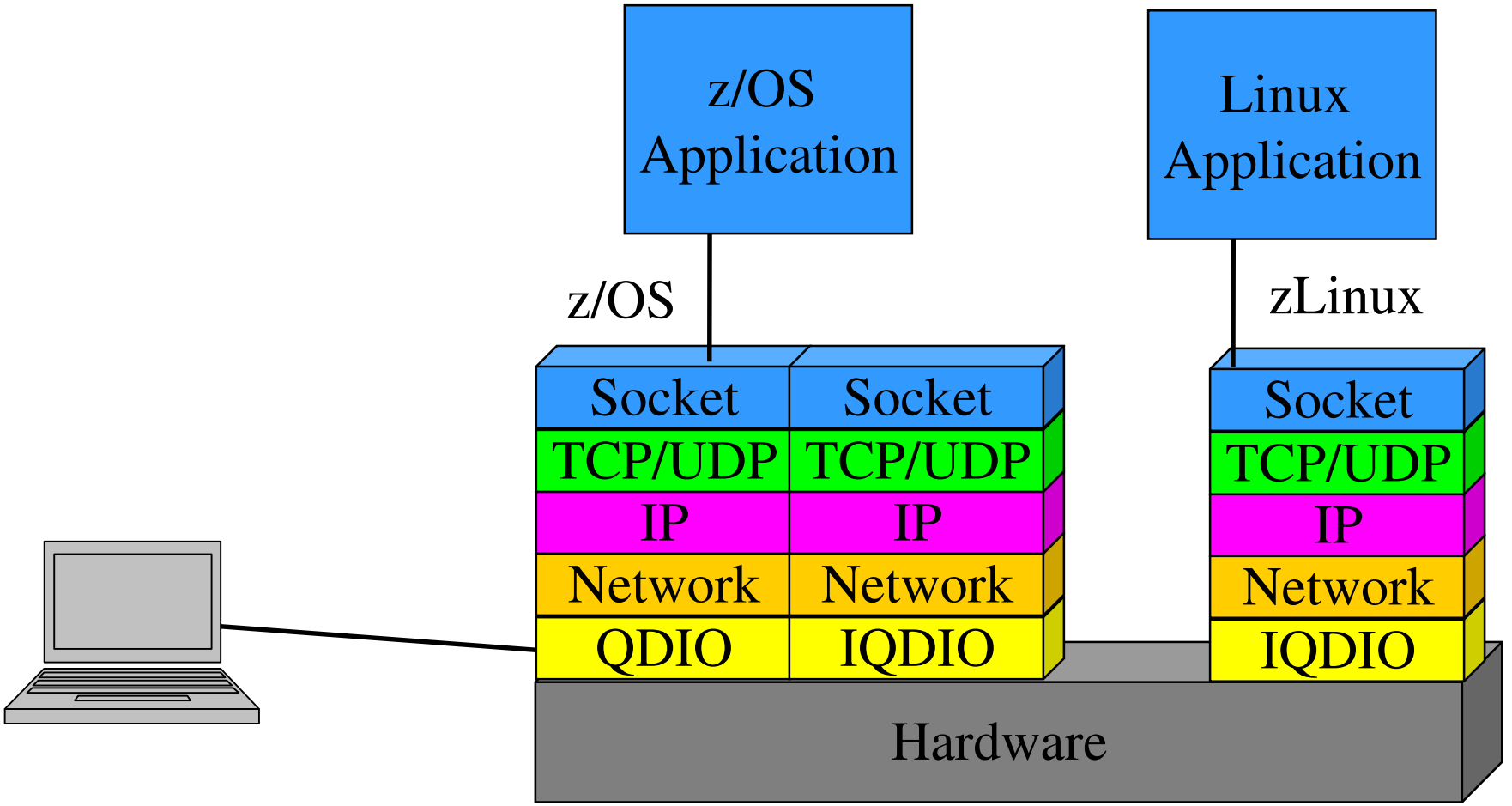
VM Guest LAN Support

- Virtual HiperSockets (Virtual Virtual sockets?!?)
- Emulates HiperSockets within a VM image
- Maximum number of unused CHPIDs -1
- 3072 I/O devices per guest LAN
- 1024 guests (TCP/IP stacks)
- Faster communication between Linux images than HiperSockets

Wheels Within Wheels



HiperSocket Accelerator



Rsockets – Beyond the Hipe

- Fast – If you like that kind of thing
- VM Guest LANs faster for inter Linux communication
- Don't expect IIOP flows to be faster…
- More secure communication
  - Unsniffable traffic between connections
  - Reduces the need for SSL
  - Lessens the dependency on encryption
    - Real performance benefits
- Less mercurial configuration than the wire stuff

Some Basic Performance Test Info

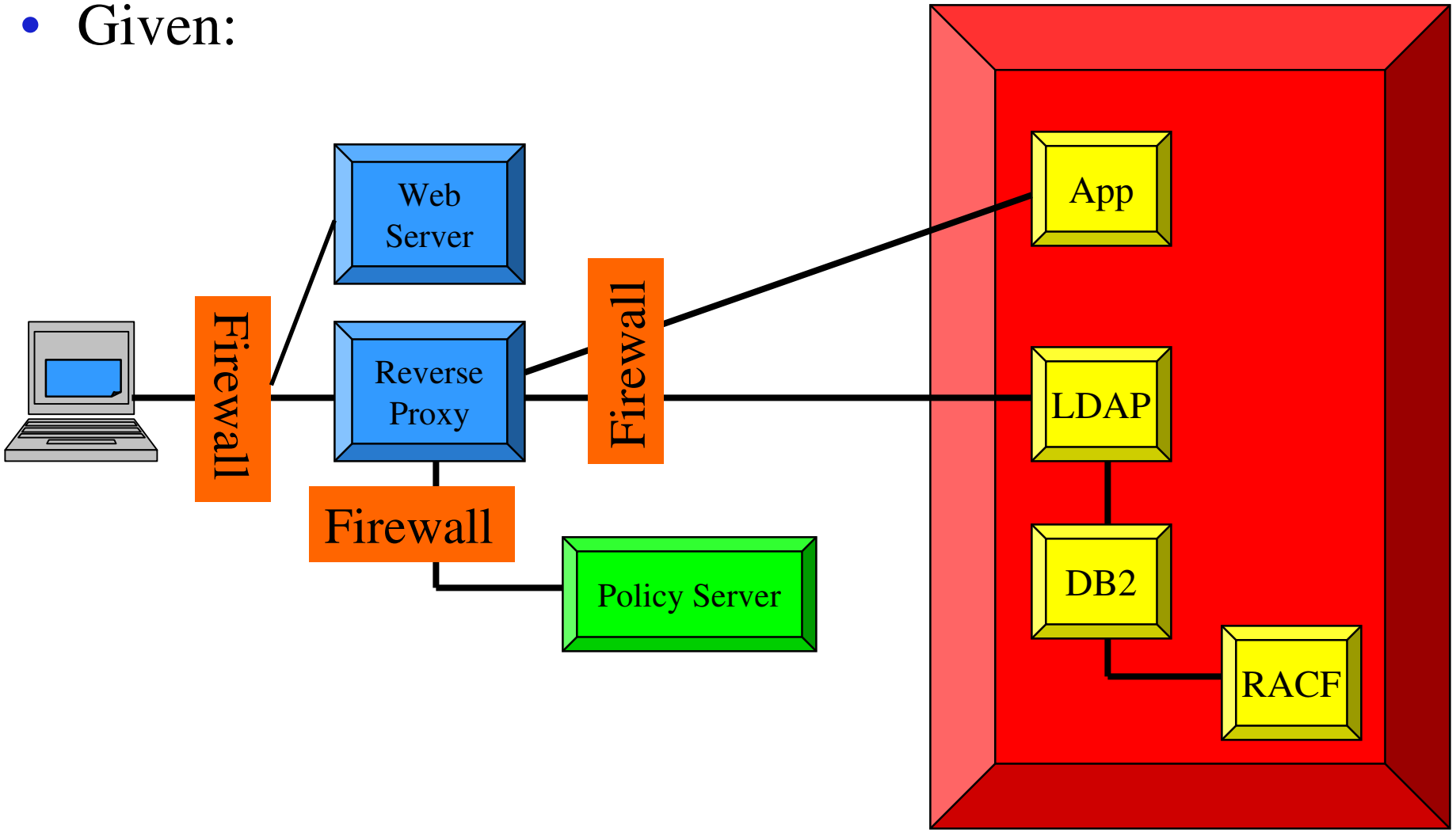
- HiperSockets consistently outperformed a single Gigabit Ethernet and provided reduced response times.
- For Streams traffic profiles, HiperSockets increased performance as the Message Frame Size (MFS or blocksize) was increased.
- MFS had minimal effect on interactive and Web-type traffic.
- Increasing the number of client-server sessions typically increased throughput.
- Changing the size of TCP/IP send and receive buffers could lead to increased throughput.
- Increasing the number of client sessions for Streams traffic typically increased the data transfer rate and CPU utilization.
- The difference in the number of memory busses in the zSeries models can impact performance.
- Adding CPUs to the client and server can increase performance, assuming the client can generate additional traffic load.
- Each operating system environment and its associated TCP/IP stack have particular parameter settings that can be used to optimize HiperSockets performance.
- Intermixing operating system environments requires attention to parameter tuning.

Factors Affecting HiperSocket Performance

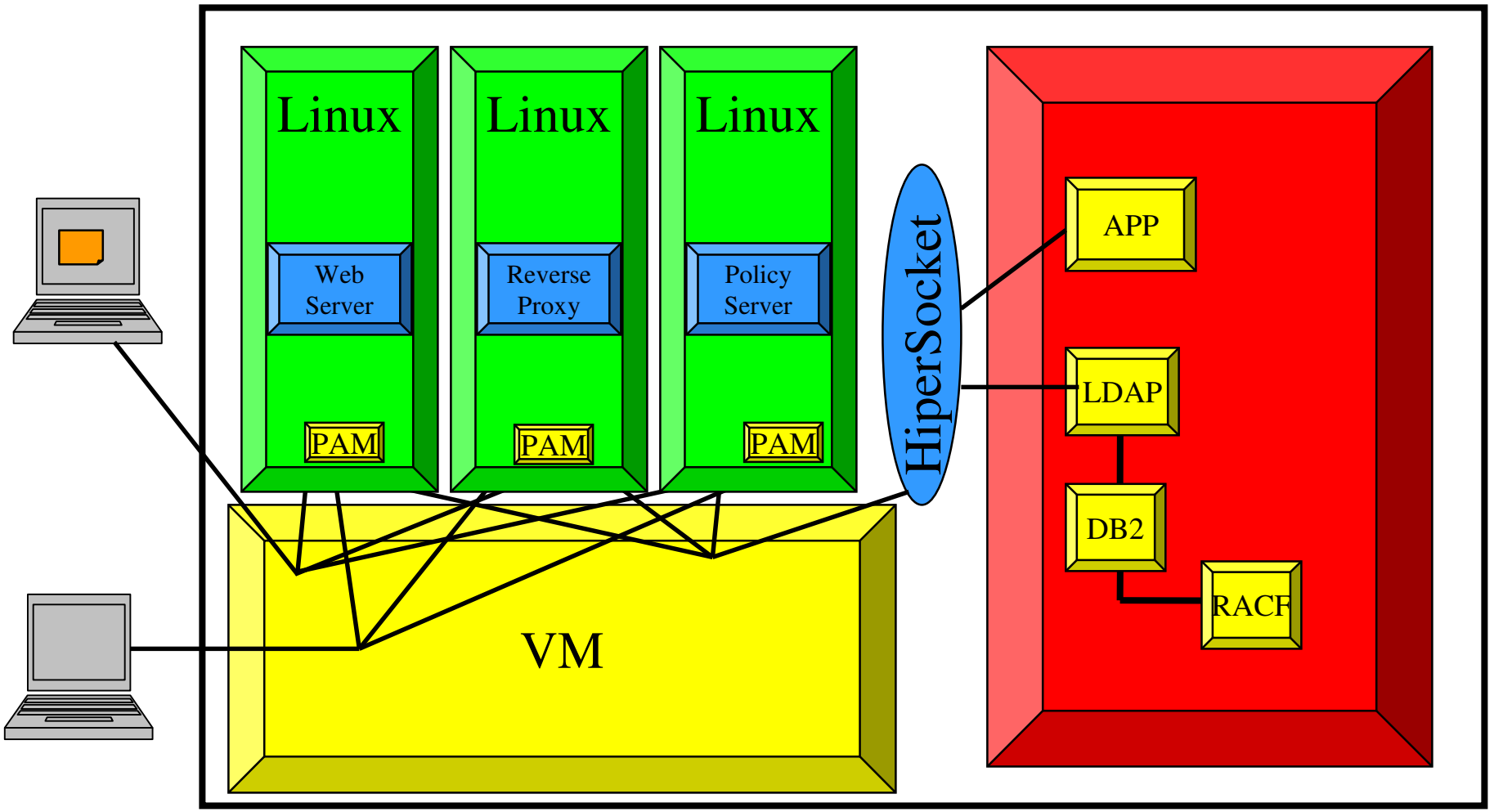
- MTU Size (Depending on traffic)
- Number of CPUs
- CPU Speed
- TCP/IP stack of OS
- Number of clients
- Network load
- Real memory
- Application Characteristics

A Real World Use of HiperSockets

- Given:



A HiperSocket Implementation



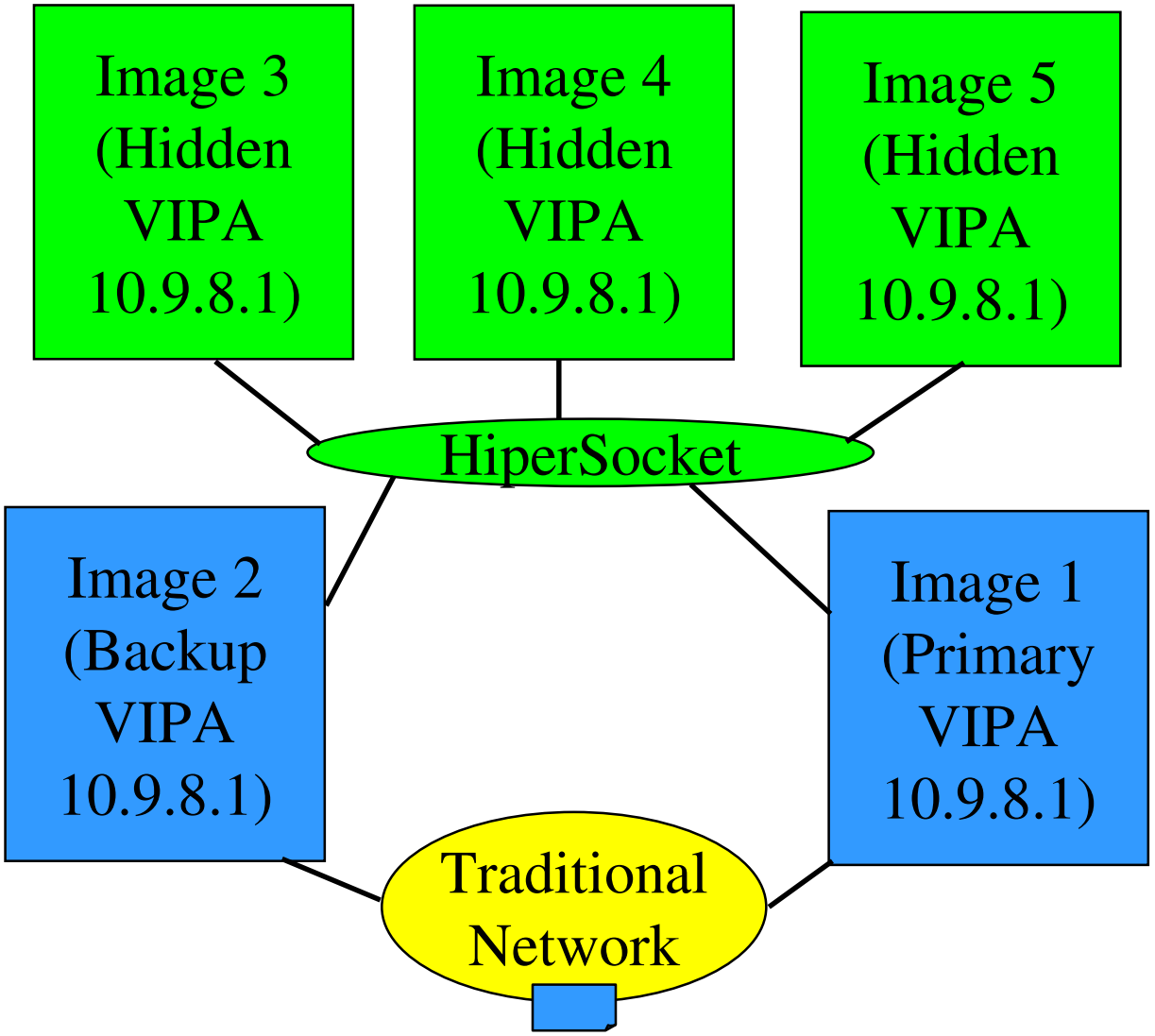
HiperSockets Implementation is Cool

- "Look Ma No Wires"
  - No packets to sniff
  - No need to encrypt
- All management on the box
  - RACF (or similar) holds all the rules for all the servers
  - Definitions can be server exclusive
- Physically separate Internet flows and Intranet flows
- Missing firewalls can be Linux firewalls using virtual HiperSockets

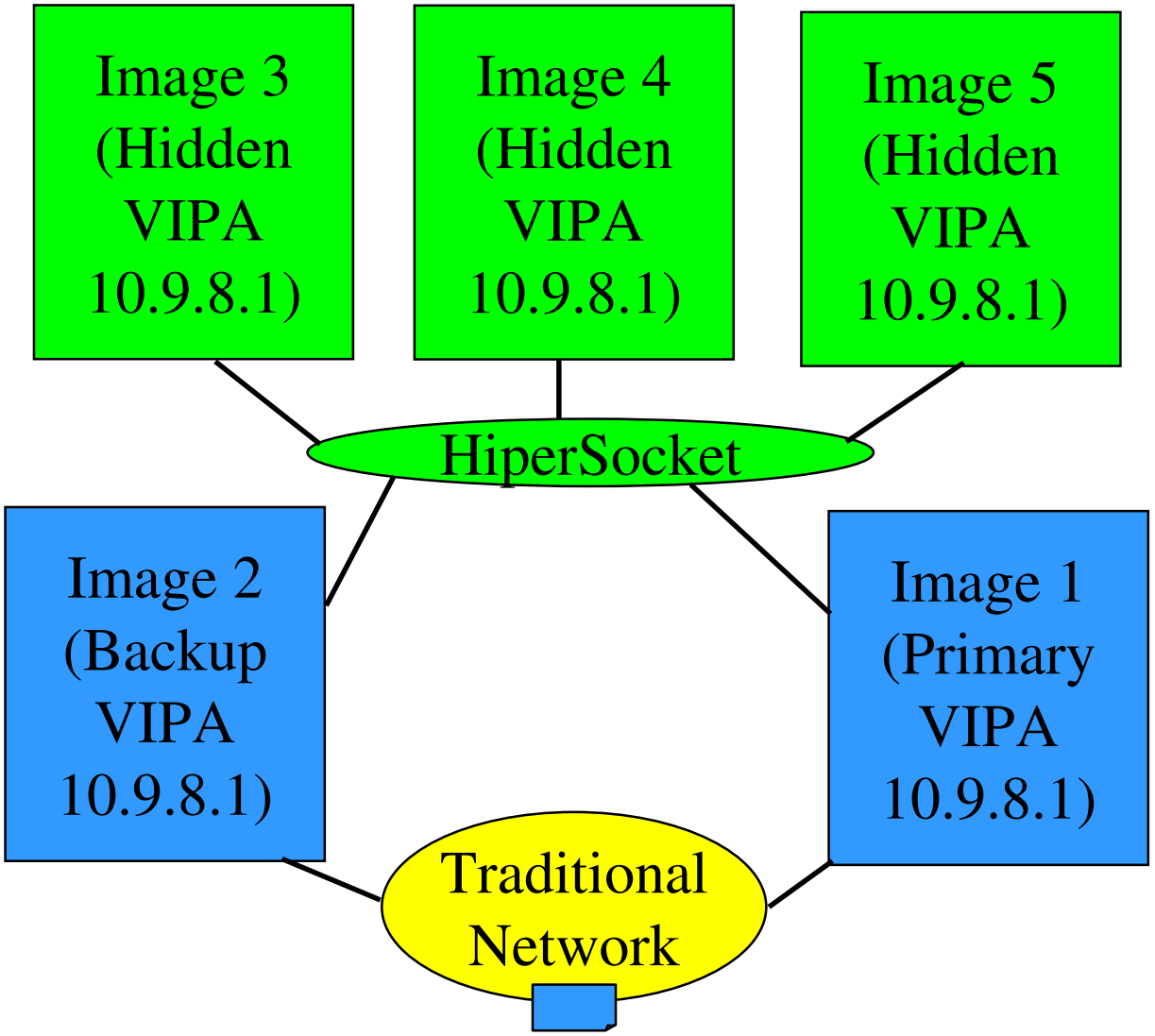
HiperSockets and z/OS are Hipercool!!

- HiperSockets can be used to offload XCF traffic
- HiperSockets implementing Sysplex sockets…
  - Smaller stack in addition to faster routing…
- HiperSockets and Dynamic VIPA
- HiperSockets and Sysplex Distributor
  - Takes advantage of XCF and WLM

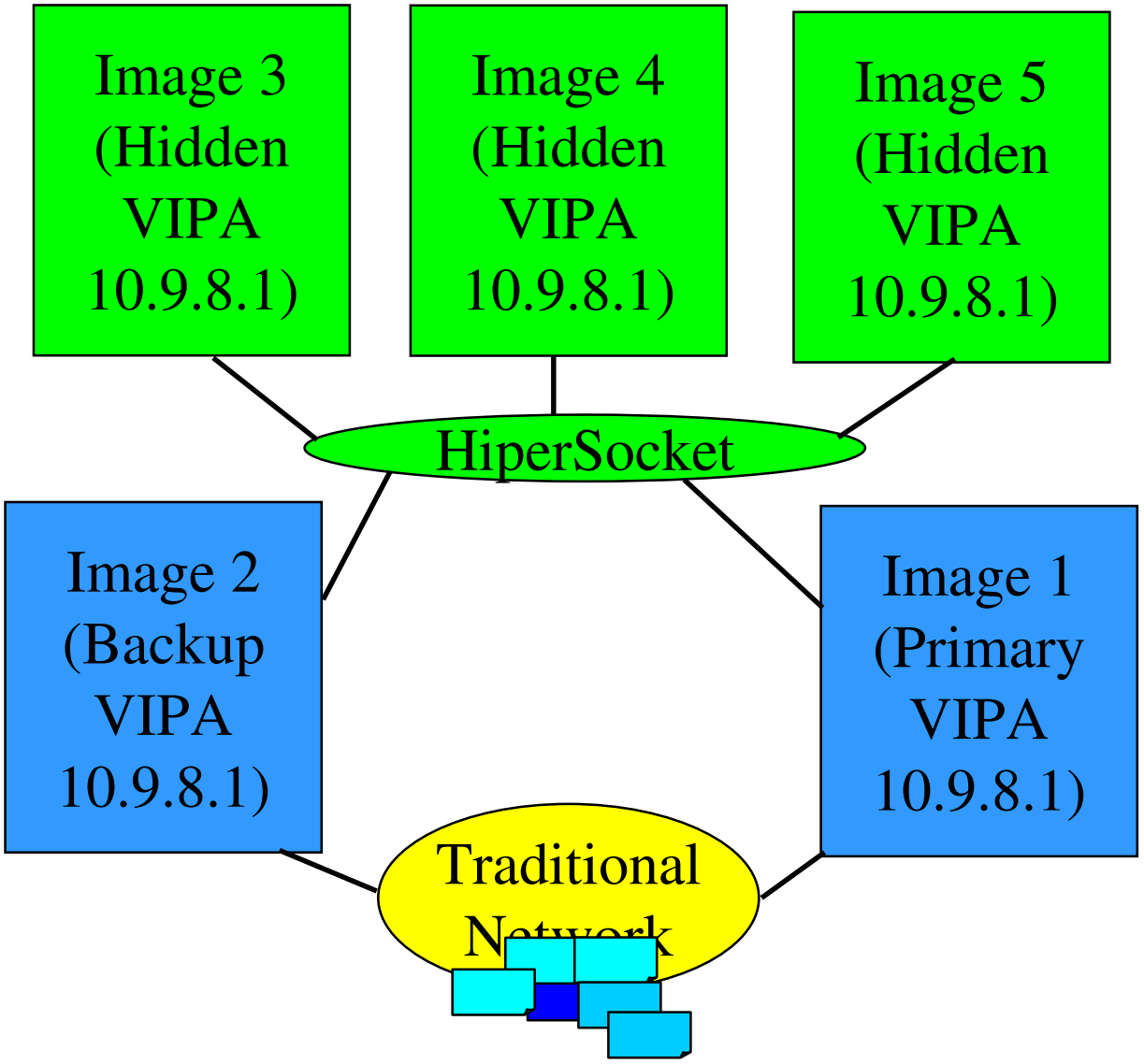
HiperSockets and Sysplex Distributor



HiperSockets and Sysplex Distributor



HiperSockets Sysplex Distributor WLM



HiperSockets in the Round

- Simple to set up
- Can be tuned for OS and applications
  - Simple straightforward knobs
- Better Performance than the wire
- Secure data transfer
  - Added performance benefit by removing SSL layers
- Has some special advantages in z/OS
  - HiperSocket Accelerator
  - Dynamic XCF
  - Sysplex Sockets
  - Sysplex Distributor