

Linux Virtual Server Tutorial

Horms (Simon Horman) – horms@valinux.co.jp
VA Linux Systems Japan, K.K. – www.valinux.co.jp
with assistance from
NTT Comware Corporation – www.nttcom.co.jp

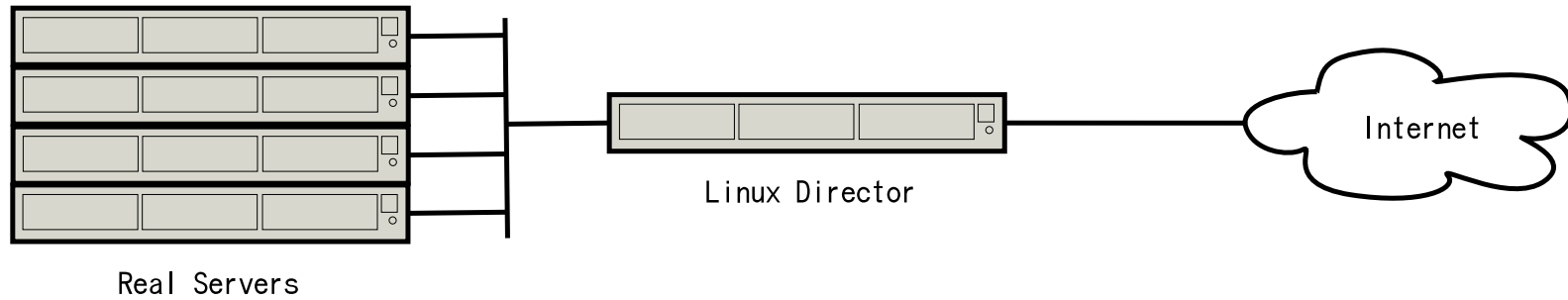
July 2003

<http://www.ultramonkey.org/>

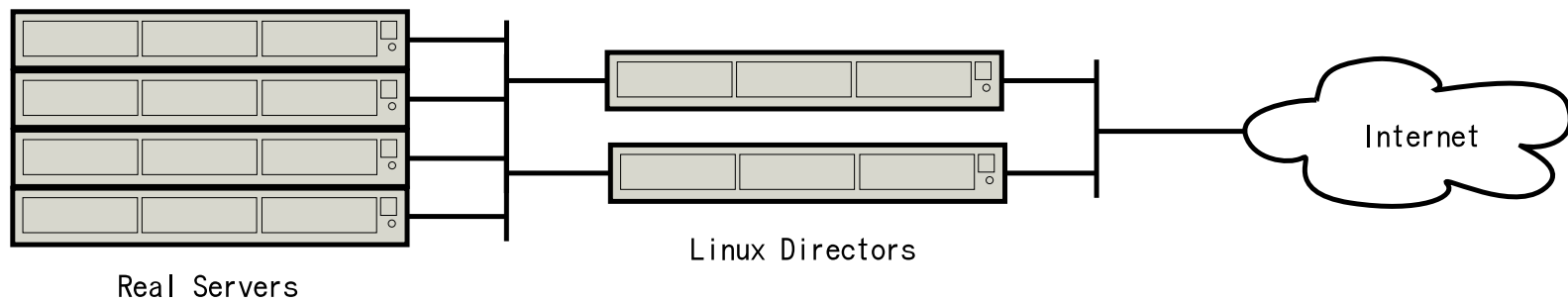
Connection Synchronisation (TCP Fail-Over)

- Active/Stand-By Linux Directors can be used to create High Availability
- After fail-over occurs the new active Linux Director is able to forward packets for new connections
- However, it does not know about existing connections and can't forward them
- This means that when fail-over occurs all existing connections will break or stall indefinitely
- This is particularly undesirable for long connections such as streaming or large software downloads

Active/Stand-By



One LinuxDirector, Single Point of Failure



Active/Stand-By Linux Directors

Synchronising Connections

- To alleviate this problem a system of synchronisation between the Linux Directors was developed
- The critical information that needs to be synchronised is:
 - protocol (TCP or UDP)
 - source ip address and port
 - virtual ip address and port
 - destination ip address and port
 - state of the connection

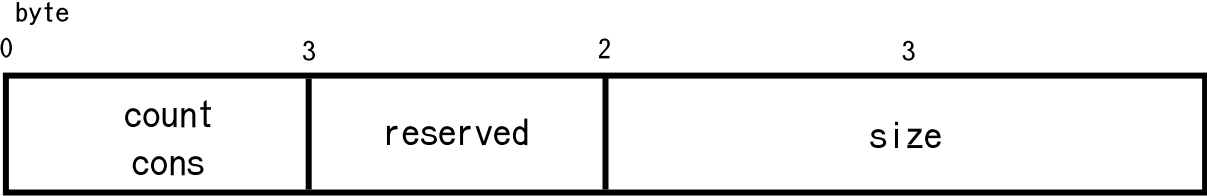
Synchronising Connections (continued)

- A connection is synchronised once it passes a threshold on the number of packets received (default is 3)
- The connection is then synchronised every 50 packets

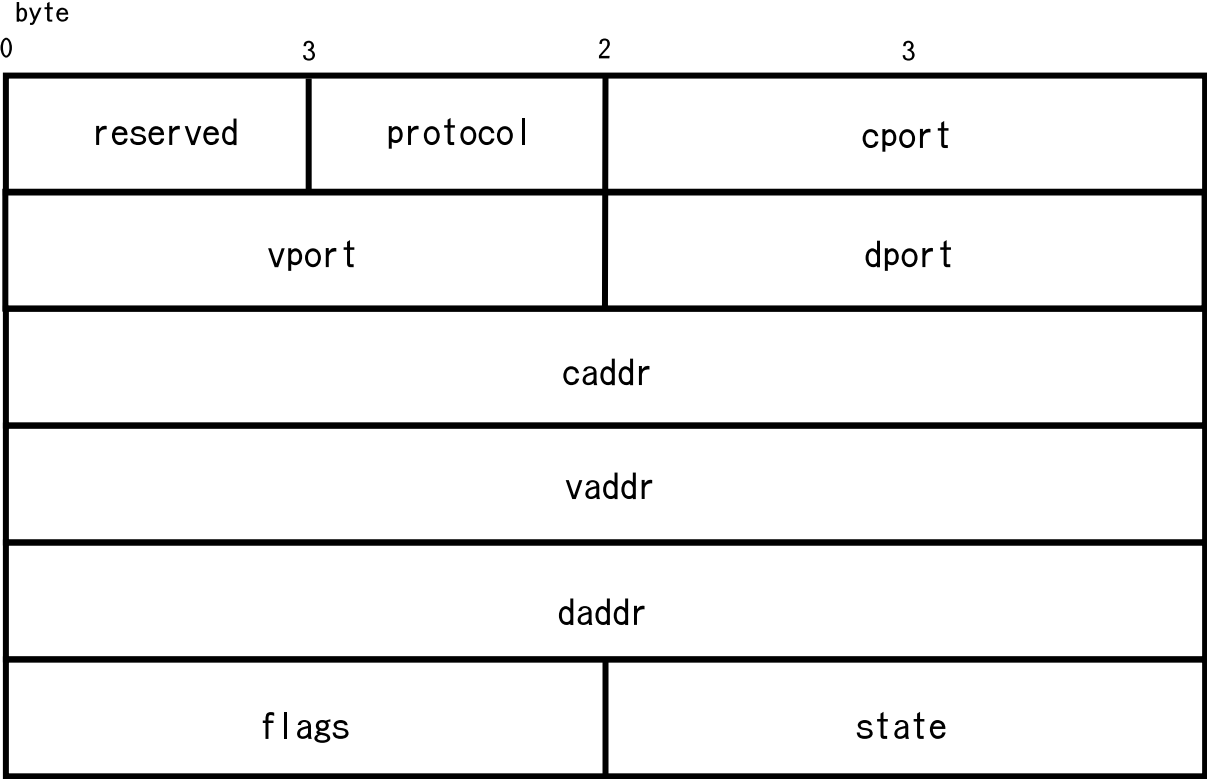
Synchronisation Packets

- When a connection is to be synchronised it is added to a queue
- Periodically the queue is flushed
- Connection synchronisation information from the queue is packed into a packet
- The connection synchronisation information for up to 50 connections may be packed into a single packet
- The synchronisation packet is sent using Multicast UDP

Synchronisation Packet Header



Synchronisation Packet Entry



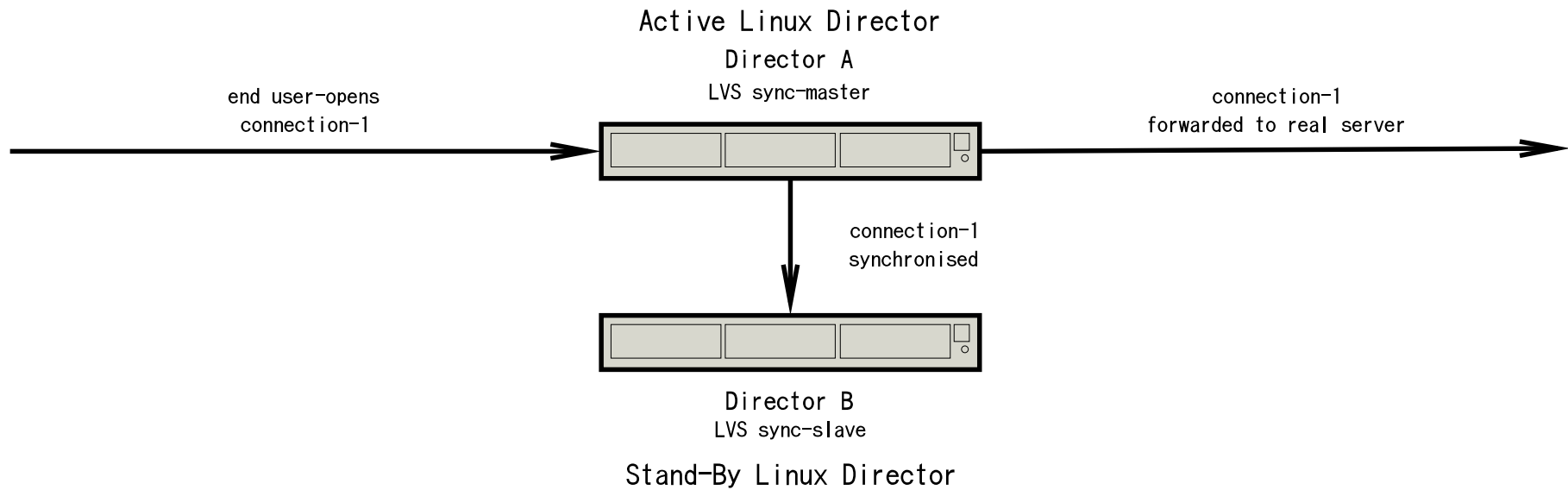
Master / Slave

- A Master/Slave relationship is used
- The master linux director sends synchronisation information using Multicast UDP
- The slave(s) receive this information and update their in-kernel LVS connection table
- Sending and receiving is handled by kernel threads

Master / Slave Problem

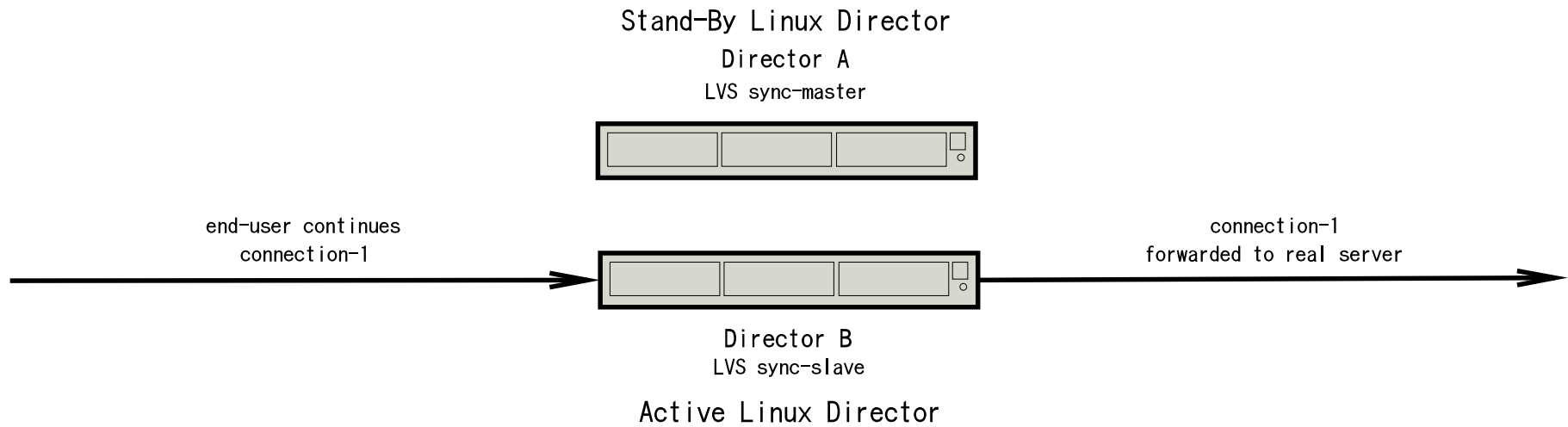
- After failover slave is the active Linux Director
- Thus, connections are not synchronised
- When a second failover occurs connections will break

Master/Slave Problem (continued)



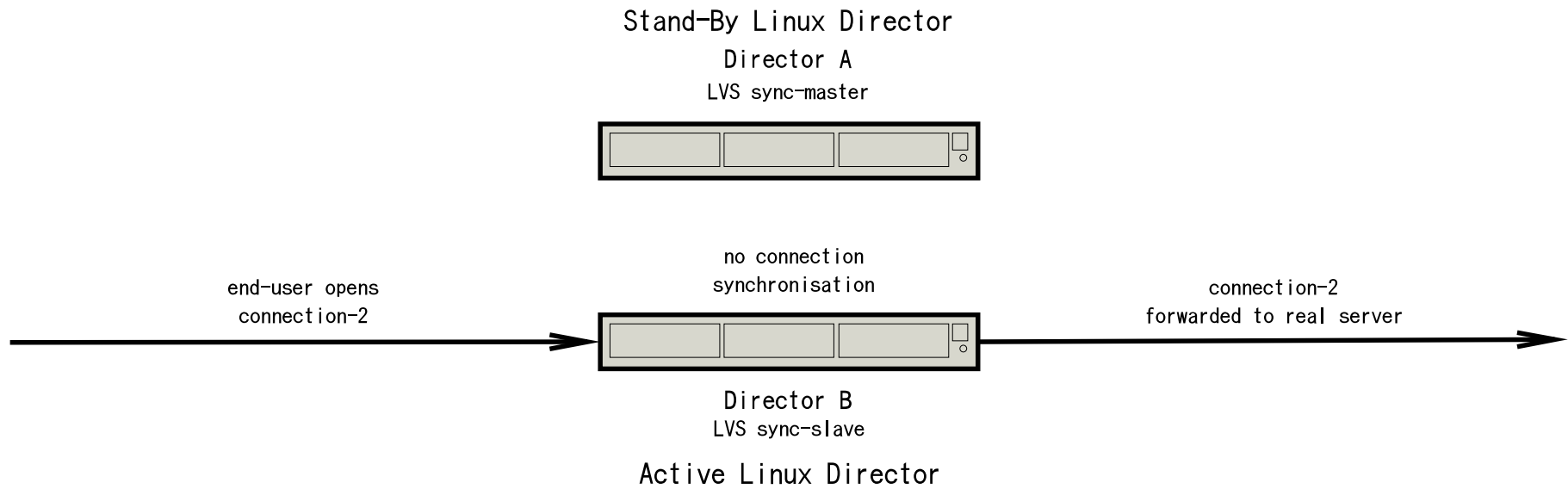
Master Linux Director is Active, connections are synchronised

Master/Slave Problem (continued)



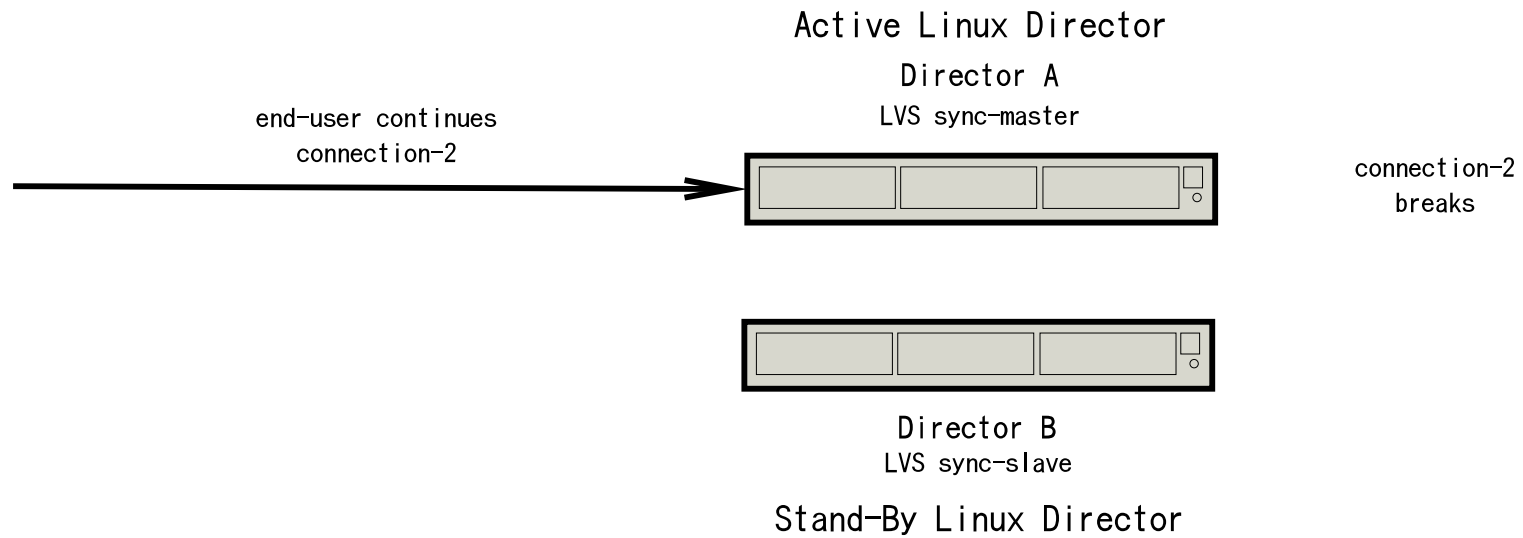
Failover occurs, synchronised connections continue

Master/Slave Problem (continued)



Slave is Active, connections are *not* synchronised

Master/Slave Problem (continued)



Another failover occurs, unsynchronised connections break.

Enhancements

- A number of enhancements have been made to the existing LVS connection synchronisation code
 - Synchronisation Method Hooks
 - User-Space Synchronisation Method
 - Additional tuning parameters

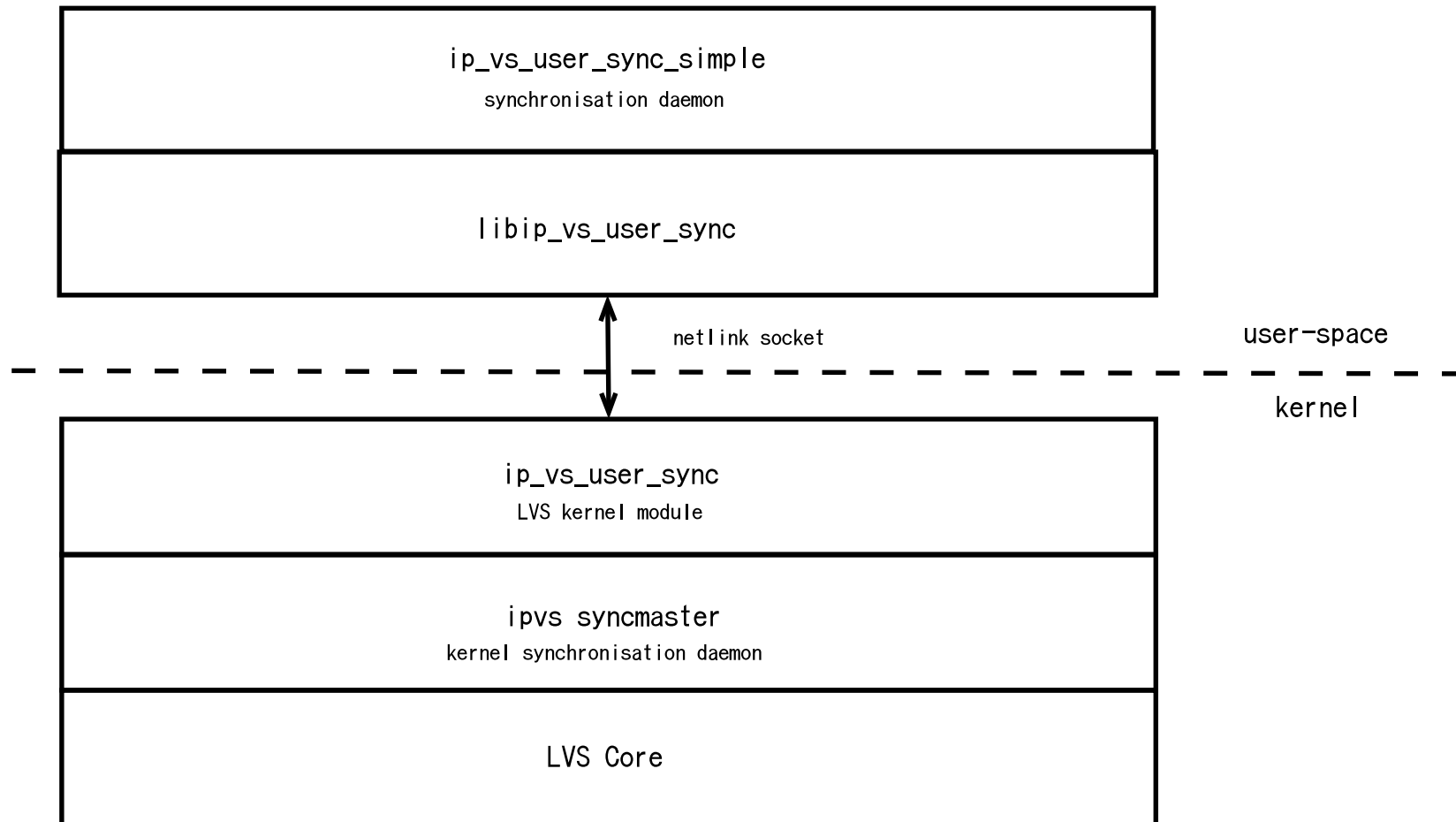
Synchronisation Method Hooks

- Allow any user-defined functions to be used to send and receive synchronisation data
- Typically implemented as a kernel module that starts a kernel thread to handle the sending and receiving
- Default functions implements the existing Master/Slave behaviour

User-Space Synchronisation Method

- Sends and receives synchronisation data to and from user-space via a netlink socket
- Allows more sophisticated synchronisation methods to be implemented
- Synchronisation data is very low volume
- Thus, there is no performance disadvantage in moving it to user-space
- A sample user-space synchronisation daemon has been implemented
 - implements peer/peer synchronisation instead of master/slave

User-Space Synchronisation Block Diagram



Additional Tuning Parameters

- `sync_frequency`: How often the a connection will be synchronised after the `sync_threshold` is reached. Default is 50
- `msg_max_size`: The maximum size of the synchronisation packet. Default is 1248 which allows synchronisation of up to 50 connections in one packet

Active Active

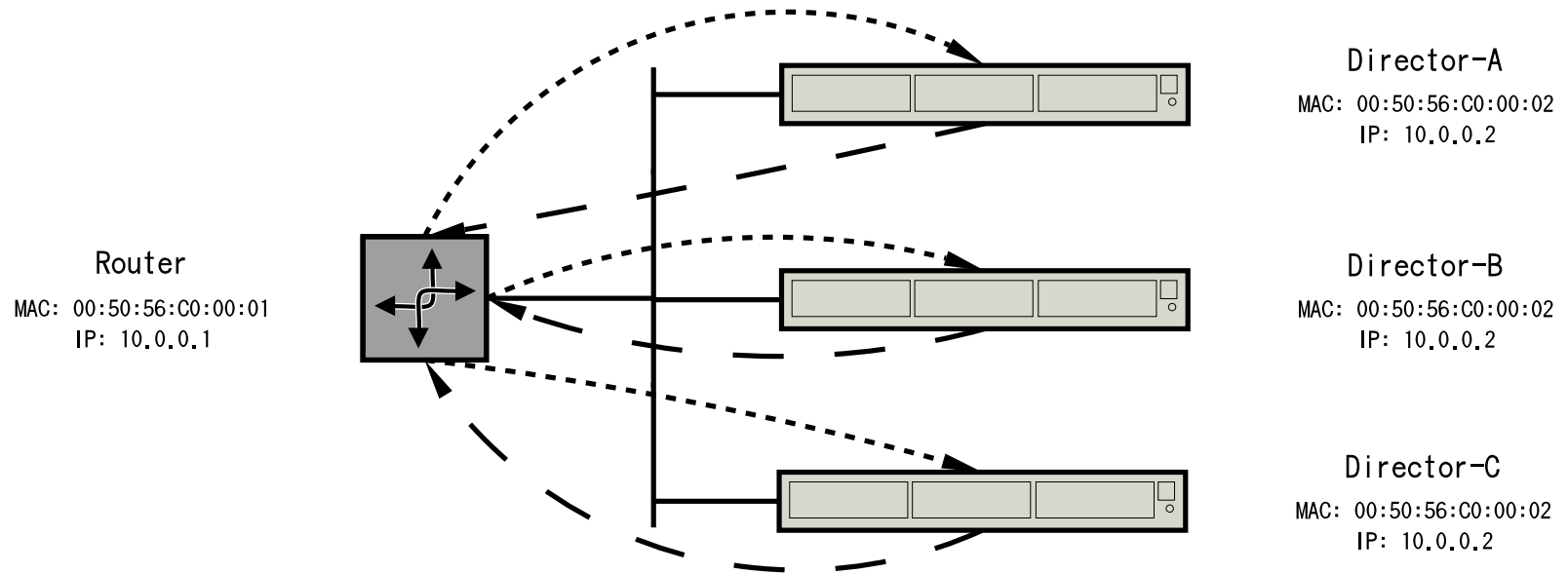
- The maximum throughput of the cluster is limited by the Linux Director
- Active Active allows throughput to be increased by adding Linux Directors
- The aim is to have multiple Linux Directors load balance traffic for the same virtual service
- But each connection should be load balanced by only one Linux Director

Common MAC and IP address

- Each Linux Director is given a common MAC and IP address
- ARP requests for the common IP address will always be answered with the common MAC address
- Thus, packets to the common IP address will be sent to the common MAC address
- The NIC and kernel on all the Linux Directors will accept these packets as it is to their IP and MAC address

Common MAC and IP address

- - - - - 1. ARP Request whois 10.0.0.2 (broadcast)
- — — 2. ARP Response 10.0.0.2 is at 00:50:56:C0:00:02



Switch Port Problem

- Switches send packets to for a given MAC address only to the port that MAC address is attached to
- In this case multiple ports have the same MAC address attached
- But the switch may still want so send the packets down only one of these ports
- This means that only one Linux Director will received packets for the common IP and MAC address

Switch Port Solution

- Switches know which MAC address is connected to which port by observing packets which are received from the port
- If the Switch doesn't know which port a MAC address belongs to it will send packets for that MAC address to *all* ports
- If Packets are never sent with the common MAC address then the switch is not able to associate the MAC address with a port
- Thus packets for the common MAC address will be sent to all ports

Switch Port Solution: Implementation

- The MAC address of the NIC is used to receive packets
 - This is the normal behaviour
- Patch the Kernel so an alternate MAC address can be used to send packets
- Alternate MAC is set using `/proc/sys/net/ipv4/conf/*/outgoing_mac`
- Simple change to `ethernet.c` to check this proc value and use it if set
- Otherwise the address of the NIC is used

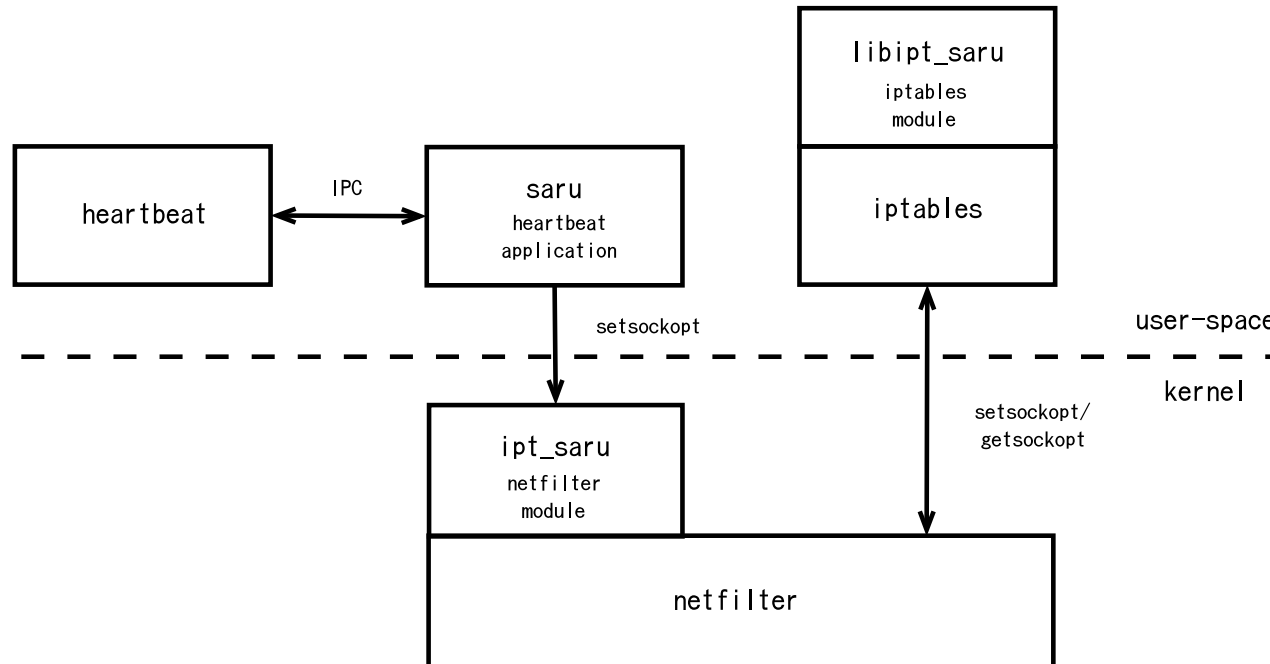
Filtering

- All the Linux Directors receive all packets for the common IP address
- Filtering ensures that only one Linux Director load balances each connection
- This can be done using netfilter/ipchains
- Static filters are easy to establish
- But do not adapt as Linux Directors are added and removed

Dynamic Filtering

- Netfilter module that can have its rules changed on the fly
- User-Space daemon that updates the module's filter rules

Dynamic Filtering Block Diagram



Dynamic Filtering: User Space-Daemon

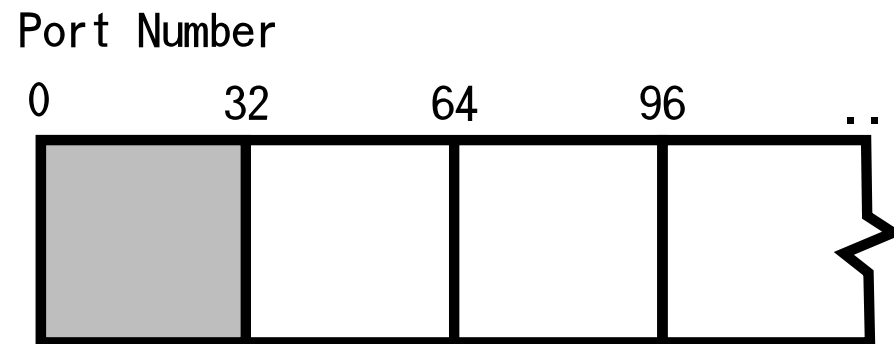
- Monitors the other Linux Directors in the cluster
- Master is elected to simplify resource allocations
- Master allocates blocks of the available connection-space to the available Linux Directors
- Blocks are reallocated by the master as Linux Directors join and leave the cluster
- If the master leaves the cluster a new one is elected

Blocks

- It is important that the decision about which Linux Director should load balance which connection is made in advance
- Making this decision as connections arrive would be too slow
- This is done by dividing the available connections into blocks

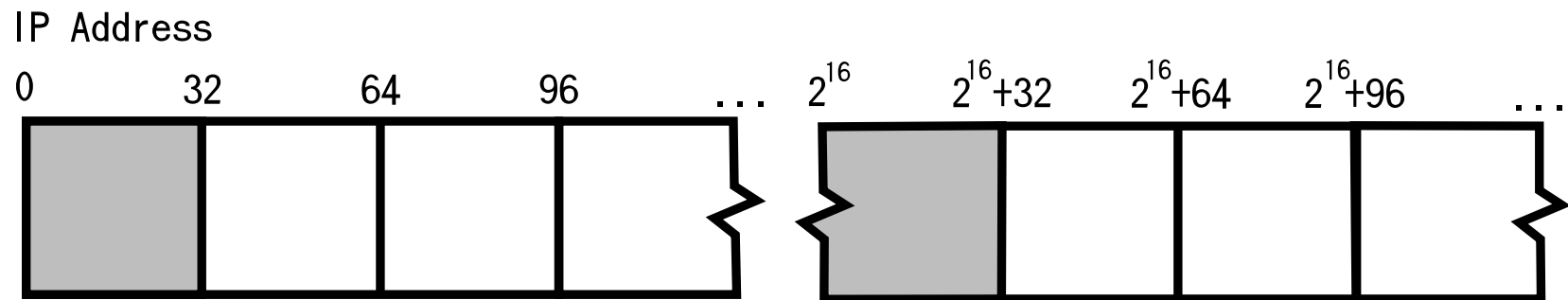
Blocks (continued)

- Currently this can be done by dividing the source or destination ports or IP addresses into 512 blocks
- Ports are divided up using $port/512$, thus there are $2^{16}/512 = 32$ contiguous ports per block.



Blocks (continued)

- IP addresses are divided up using $(addr \bmod 2^{16}) \bmod 512$, thus there are 2^{16} sub-blocks of $2^{32}/2^{16}/512 = 32$ contiguous addresses per block.



Blocks (continued)

- When a connection is received its block is looked up
- If this block has been allocated to the node by saru then the packet is accepted
- Otherwise it is dropped – another node will accept it

Active-Active Applications

- Designed to run on Linux Directors running LVS
- But the design is generic - may handle any type of network packets
- Thus, may be used on any type of server

Questions