

## 8:ipvsadm

### IPVSADM(8) Linux Administrator's Guide IPVSADM(8)

#### NAME

ipvsadm - Linux Virtual Server administration

#### SYNOPSIS

```
ipvsadm -A|E -t|u|f service-address [-s scheduler]
        [-p [timeout]] [-M netmask]
ipvsadm -D -t|u|f service-address
ipvsadm -C
ipvsadm -R
ipvsadm -S [-n]
ipvsadm -a|e -t|u|f service-address -r server-address
        [-g|i|m] [-w weight] [-x upper] [-y lower]
ipvsadm -d -t|u|f service-address -r server-address
ipvsadm -L|l [options]
ipvsadm -Z [-t|u|f service-address]
ipvsadm --set tcp tcpfin udp
ipvsadm --start-daemon state [--mcast-interface interface]
        [--syncid syncid]
ipvsadm --stop-daemon state
ipvsadm -h
```

#### DESCRIPTION

Ipvsadm(8) is used to set up, maintain or inspect the virtual server table in the Linux kernel. The Linux Virtual Server can be used to build scalable network services based on a cluster of two or more nodes. The active node of the cluster redirects service requests to a collection of server hosts that will actually perform the services. Supported features include two protocols (TCP and UDP), three packet-forwarding methods (NAT, tunneling, and direct routing), and eight load balancing algorithms (round robin, weighted round robin, least-connection, weighted least-connection, locality-based least-connection, locality-based least-connection with replication, destination-hashing, and source-hashing).

The command has two basic formats for execution:

```
ipvsadm COMMAND [protocol] service-address
        [scheduling-method] [persistence options]
ipvsadm command [protocol] service-address
        server-address [packet-forwarding-method]
        [weight options]
```

The first format manipulates a virtual service and the algorithm for assigning service requests to real servers. Optionally, a persistent timeout and network mask for the granularity of a persistent service may be specified. The second format manipulates a real server that is associated with an existing virtual service. When specifying a real server, the packet-forwarding method and the weight of the real server, relative to other real servers for the virtual service, may be specified, otherwise defaults will be used.

#### COMMANDS

ipvsadm(8) recognises the commands described below. Upper-case commands maintain virtual services. Lower-case commands maintain real servers that are associated with a virtual service.

-A, --add-service

Add a virtual service. A service address is uniquely defined by a triplet: IP address, port number, and protocol. Alternatively, a virtual service may be defined by a firewall-mark.

-E, --edit-service

Edit a virtual service.

-D, --delete-service

Delete a virtual service, along with any associated real servers.

-C, --clear

Clear the virtual server table.

-R, --restore

Restore Linux Virtual Server rules from stdin. Each line read from stdin will be treated as the command line options to a separate invocation of ipvsadm. Lines read from stdin can optionally begin with "ipvsadm". This option is useful to avoid executing a large number of ipvsadm commands when constructing an extensive routing table.

-S, --save

Dump the Linux Virtual Server rules to stdout in a format that can be read by -R|--restore.

-a, --add-server

Add a real server to a virtual service.

-e, --edit-server

Edit a real server in a virtual service.

-d, --delete-server

Remove a real server from a virtual service.

-L, -l, --list

List the virtual server table if no argument is specified. If a service-address is selected, list this service only. If the -c option is selected, then display the connection table. The exact output is affected by the other arguments given.

-Z, --zero

Zero the packet, byte and rate counters in a service or all services.

`--set tcp tcpfin udp`  
Change the timeout values used for IPVS connections. This command always takes 3 parameters, representing the timeout values (in seconds) for TCP sessions, TCP sessions after receiving a FIN packet, and UDP packets, respectively. A timeout value 0 means that the current timeout value of the corresponding entry is preserved.

`--start-daemon state`  
Start the connection synchronization daemon. The state is to indicate that the daemon is started as master or backup. The connection synchronization daemon is implemented inside the Linux kernel. The master daemon running at the primary load balancer multicasts changes of connections periodically, and the backup daemon running at the backup load balancers receives multicast message and creates corresponding connections. Then, in case the primary load balancer fails, a backup load balancer will takeover, and it has state of almost all connections, so that almost all established connections can continue to access the service.

`--stop-daemon`  
Stop the connection synchronization daemon.

`-h, --help`  
Display a description of the command syntax.

#### PARAMETERS

The commands above accept or require zero or more of the following parameters.

`-t, --tcp-service service-address`  
Use TCP service. The service-address is of the form host[:port]. Host may be one of a plain IP address or a hostname. Port may be either a plain port number or the service name of port. The Port may be omitted, in which case zero will be used. A Port of zero is only valid if the service is persistent as the `-p|--persistent` option, in which case it is a wild-card port, that is connections will be accepted to any port.

`-u, --udp-service service-address`  
Use UDP service. See the `-t|--tcp-service` for the description of the service-address.

`-f, --fwmark-service integer`  
Use a firewall-mark, an integer value greater than zero, to denote a virtual service instead of an address, port and protocol (UDP or TCP). The marking of packets with a firewall-mark is configured using the `-m|--mark` option to `iptables(8)`. It can be used to build a virtual service associated with the same real servers, covering multiple IP address, port and protocol triplets.  
Using firewall-mark virtual services provides a convenient method of grouping together different IP

addresses, ports and protocols into a single virtual service. This is useful for both simplifying configuration if a large number of virtual services are required and grouping persistence across what would otherwise be multiple virtual services.

-s, --scheduler scheduling-method

scheduling-method Algorithm for allocating TCP connections and UDP datagrams to real servers. Scheduling algorithms are implemented as kernel modules. Ten are shipped with the Linux Virtual Server:

rr - Robin Robin: distributes jobs equally amongst the available real servers.

wrr - Weighted Round Robin: assigns jobs to real servers proportionally to their real servers' weight. Servers with higher weights receive new jobs first and get more jobs than servers with lower weights. Servers with equal weights get an equal distribution of new jobs.

lc - Least-Connection: assigns more jobs to real servers with fewer active jobs.

wlc - Weighted Least-Connection: assigns more jobs to servers with fewer jobs and relative to the real servers' weight ( $C_i/W_i$ ). This is the default.

lblc - Locality-Based Least-Connection: assigns jobs destined for the same IP address to the same server if the server is not overloaded and available; otherwise assign jobs to servers with fewer jobs, and keep it for future assignment.

lblcr - Locality-Based Least-Connection with Replication: assigns jobs destined for the same IP address to the least-connection node in the server set for the IP address. If all the node in the server set are over loaded, it picks up a node with fewer jobs in the cluster and adds it in the server set for the target. If the server set has not been modified for the specified time, the most loaded node is removed from the server set, in order to avoid high degree of replication.

dh - Destination Hashing: assigns jobs to servers through looking up a statically assigned hash table by their destination IP addresses.

sh - Source Hashing: assigns jobs to servers through looking up a statically assigned hash table by their source IP addresses.

sed - Shortest Expected Delay: assigns an incoming job to the server with the shortest expected delay. The expected delay that the job will experience is  $(C_i + 1) / U_i$  if sent to the  $i$ th server, in which  $C_i$  is the number of jobs on the  $i$ th server and  $U_i$  is the fixed service rate (weight) of the  $i$ th server.

nq - Never Queue: assigns an incoming job to an

idle server if there is, instead of waiting for a fast one; if all the servers are busy, it adopts the Shortest Expected Delay policy to assign the job.

`-p, --persistent [timeout]`

Specify that a virtual service is persistent. If this option is specified, multiple requests from a client are redirected to the same real server selected for the first request. Optionally, the timeout of persistent sessions may be specified given in seconds, otherwise the default of 300 seconds will be used. This option may be used in conjunction with protocols such as SSL or FTP where it is important that clients consistently connect with the same real server.

Note: If a virtual service is to handle FTP connections then persistence must be set for the virtual service if Direct Routing or Tunnelling is used as the forwarding mechanism. If Masquerading is used in conjunction with an FTP service than persistence is not necessary, but the `ip_vs_ftp` kernel module must be used. This module may be manually inserted into the kernel using `insmod(8)`.

`-M, --netmask netmask`

Specify the granularity with which clients are grouped for persistent virtual services. The source address of the request is masked with this netmask to direct all clients from a network to the same real server. The default is 255.255.255.255, that is, the persistence granularity is per client host. Less specific netmasks may be used to resolve problems with non-persistent cache clusters on the client side.

`-r, --real-server server-address`

Real server that an associated request for service may be assigned to. The `server-address` is the host address of a real server, and may plus port. Host can be either a plain IP address or a hostname. Port can be either a plain port number or the service name of port. In the case of the masquerading method, the host address is usually an [RFC 1918](#) private IP address, and the port can be different from that of the associated service. With the tunneling and direct routing methods, port must be equal to that of the service address. For normal services, the port specified in the service address will be used if port is not specified. For `fwmark` services, port may be omitted, in which case the destination port on the real server will be the destination port of the request sent to the virtual service.

`[packet-forwarding-method]`

`-g, --gatewaying` Use gatewaying (direct routing).

This is the default.

`-i, --ipip` Use ipip encapsulation (tunneling).

`-m, --masquerading` Use masquerading (network access translation, or NAT).

Note: Regardless of the packet-forwarding mechanism specified, real servers for addresses for which there are interfaces on the local node will be use the local forwarding method, then packets for the servers will be passed to upper layer on the local node. This cannot be specified by `ipvsadm`, rather it set by the kernel as real servers are added or modified.

`-w, --weight weight`

Weight is an integer specifying the capacity of a server relative to the others in the pool. The valid values of weight are 0 through to 65535. The default is 1. Quiescent servers are specified with a weight of zero. A quiescent server will receive no new jobs but still serve the existing jobs, for all scheduling algorithms distributed with the Linux Virtual Server. Setting a quiescent server may be useful if the server is overloaded or needs to be taken out of service for maintenance.

`-x, --u-threshold uthreshold`

`uthreshold` is an integer specifying the upper connection threshold of a server. The valid values of `uthreshold` are 0 through to 65535. The default is 0, which means the upper connection threshold is not set. If `uthreshold` is set with other values, no new connections will be sent to the server when the number of its connections exceeds its upper connection threshold.

`-y, --l-threshold lthreshold`

`lthreshold` is an integer specifying the lower connection threshold of a server. The valid values of `lthreshold` are 0 through to 65535. The default is 0, which means the lower connection threshold is not set. If `lthreshold` is set with other values, the server will receive new connections when the number of its connections drops below its lower connection threshold. If `lthreshold` is not set but `uthreshold` is set, the server will receive new connections when the number of its connections drops below three fourth of its upper connection threshold.

`--mcast-interface interface`

Specify the multicast interface that the sync master daemon sends outgoing multicasts through, or the sync backup daemon listens to for multicasts.

`--syncid syncid`

Specify the syncid that the sync master daemon fills in the SyncID header while sending multicast messages, or the sync backup daemon uses to filter

out multicast messages not matched with the SyncID value. The valid values of syncid are 0 through to 255. The default is 0, which means no filtering at all.

`-c, --connection`

Connection output. The list command with this option will list current IPVS connections.

`--timeout`

Timeout output. The list command with this option will display the timeout values (in seconds) for TCP sessions, TCP sessions after receiving a FIN packet, and UDP packets.

`--daemon`

Daemon information output. The list command with this option will display the daemon status and its multicast interface.

`--stats`

Output of statistics information. The list command with this option will display the statistics information of services and their servers.

`--rate` Output of rate information. The list command with this option will display the rate information (such as connections/second, bytes/second and packets/second) of services and their servers.

`--thresholds`

Output of thresholds information. The list command with this option will display the upper/lower connection threshold information of each server in service listing.

`--persistent-conn`

Output of persistent connection information. The list command with this option will display the persistent connection counter information of each server in service listing. The persistent connection is used to forward the actual connections from the same client/network to the same server.

`--sort` Sort the list of virtual services and real servers. The virtual service entries are sorted in ascending order by <protocol, address, port>. The real server entries are sorted in ascending order by <address, port>.

`-n, --numeric`

Numeric output. IP addresses and port numbers will be printed in numeric format rather than as host names and services respectively, which is the default.

## EXAMPLE 1 - Simple Virtual Service

The following commands configure a Linux Director to distribute incoming requests addressed to port 80 on 207.175.44.110 equally to port 80 on five real servers. The forwarding method used in this example is NAT, with

each of the real servers being masqueraded by the Linux Director.

```
ipvsadm -A -t 207.175.44.110:80 -s rr
ipvsadm -a -t 207.175.44.110:80 -r 192.168.10.1:80 -m
ipvsadm -a -t 207.175.44.110:80 -r 192.168.10.2:80 -m
ipvsadm -a -t 207.175.44.110:80 -r 192.168.10.3:80 -m
ipvsadm -a -t 207.175.44.110:80 -r 192.168.10.4:80 -m
ipvsadm -a -t 207.175.44.110:80 -r 192.168.10.5:80 -m
```

Alternatively, this could be achieved in a single ipvsadm command.

```
echo "
-A -t 207.175.44.110:80 -s rr
-a -t 207.175.44.110:80 -r 192.168.10.1:80 -m
-a -t 207.175.44.110:80 -r 192.168.10.2:80 -m
-a -t 207.175.44.110:80 -r 192.168.10.3:80 -m
-a -t 207.175.44.110:80 -r 192.168.10.4:80 -m
-a -t 207.175.44.110:80 -r 192.168.10.5:80 -m
" | ipvsadm -R
```

As masquerading is used as the forwarding mechanism in this example, the default route of the real servers must be set to the linux director, which will need to be configured to forward and masquerade packets. This can be achieved using the following commands:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

## EXAMPLE 2 - Firewall-Mark Virtual Service

The following commands configure a Linux Director to distribute incoming requests addressed to any port on 207.175.44.110 or 207.175.44.111 equally to the corresponding port on five real servers. As per the previous example, the forwarding method used in this example is NAT, with each of the real servers being masqueraded by the Linux Director.

```
ipvsadm -A -f 1 -s rr
ipvsadm -a -f 1 -r 192.168.10.1:0 -m
ipvsadm -a -f 1 -r 192.168.10.2:0 -m
ipvsadm -a -f 1 -r 192.168.10.3:0 -m
ipvsadm -a -f 1 -r 192.168.10.4:0 -m
ipvsadm -a -f 1 -r 192.168.10.5:0 -m
```

As masquerading is used as the forwarding mechanism in this example, the default route of the real servers must be set to the linux director, which will need to be configured to forward and masquerade packets. The real server should also be configured to mark incoming packets addressed to any port on 207.175.44.110 and 207.175.44.111 with firewall-mark 1. If FTP traffic is to be handled by this virtual service, then the ip\_vs\_ftp kernel module needs to be inserted into the kernel. These operations can be achieved using the following commands:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
modprobe ip_tables
iptables -A PREROUTING -t mangle -d 207.175.44.110/31 -j MARK --set-mark
```

```
modprobe ip_vs_ftp
```

## NOTES

The Linux Virtual Server implements three defense strategies against some types of denial of service (DoS) attacks. The Linux Director creates an entry for each connection in order to keep its state, and each entry occupies 128 bytes effective memory. LVS's vulnerability to a DoS attack lies in the potential to increase the number entries as much as possible until the linux director runs out of memory. The three defense strategies against the attack are: Randomly drop some entries in the table. Drop 1/rate packets before forwarding them. And use secure tcp state transition table and short timeouts. The strategies are controlled by sysctl variables and corresponding entries in the /proc filesystem:

```
/proc/sys/net/ipv4/vs/drop_entry
/proc/sys/net/ipv4/vs/drop_packet
/proc/sys/net/ipv4/vs/secure_tcp
```

Valid values for each variable are 0 through to 3. The default value is 0, which disables the respective defense strategy. 1 and 2 are automatic modes - when there is no enough available memory, the respective strategy will be enabled and the variable is automatically set to 2, otherwise the strategy is disabled and the variable is set to 1. A value of 3 denotes that the respective strategy is always enabled. The available memory threshold and secure TCP timeouts can be tuned using the sysctl variables and corresponding entries in the /proc filesystem:

```
/proc/sys/net/ipv4/vs/amemthresh
/proc/sys/net/ipv4/vs/timeout_*
```

## FILES

```
/proc/net/ip_vs
/proc/net/ip_vs_app
/proc/net/ip_vs_conn
/proc/net/ip_vs_stats
/proc/sys/net/ipv4/vs/am_droprate
/proc/sys/net/ipv4/vs/amemthresh
/proc/sys/net/ipv4/vs/drop_entry
/proc/sys/net/ipv4/vs/drop_packet
/proc/sys/net/ipv4/vs/secure_tcp
/proc/sys/net/ipv4/vs/timeout_close
/proc/sys/net/ipv4/vs/timeout_closewait
/proc/sys/net/ipv4/vs/timeout_established
/proc/sys/net/ipv4/vs/timeout_finwait
/proc/sys/net/ipv4/vs/timeout_icmp
/proc/sys/net/ipv4/vs/timeout_lastack
/proc/sys/net/ipv4/vs/timeout_listen
/proc/sys/net/ipv4/vs/timeout_synack
```

```
/proc/sys/net/ipv4/vs/timeout_synrecv  
/proc/sys/net/ipv4/vs/timeout_synsent  
/proc/sys/net/ipv4/vs/timeout_timewait  
/proc/sys/net/ipv4/vs/timeout_udp
```

## SEE ALSO

The LVS web site ( <http://www.linuxvirtualserver.org/> ) for more documentation about LVS.

ipvsadm-save(8), ipvsadm-restore(8), iptables(8),  
insmod(8), modprobe(8)

## AUTHORS

ipvsadm - Wensong Zhang <wensong@linuxvirtualserver.org>

Peter Kese <peter.kese@ijs.si>

man page - Mike Wangsmo <wanger@redhat.com>

Wensong Zhang <wensong@linuxvirtualserver.org>

Horms <horms@verge.net.au>

LVS Administration 5th July 2003 IPVSADM(8)