NAME

balance 3.54 − A simple TCP proxy with load balancing and failover mechanisms.

SYNOPSIS

balance [**-b addr**] [**-B addr**] [**-t sec**] [**-T sec**] [**-adfpHM6**] port host1[:port1[:maxc]] [!|%] [... hostn[:portn[:maxc]]]

balance [**-b addr**] **-i** [**-d**] [**-M**] port

balance [**-b addr**] **-c cmd** [**-d**] [**-M**] port

DESCRIPTION

Balance is a simple, generic "userland" TCP proxy, which allows simple round-robin load balancing and graceful failover between several destination servers.

Balance supports IPv6 on the listening side which makes it a very useful tool for IPv6 migration of IPv4 only services and servers.

Balance is available at http://balance.sourceforge.net.

Definitions: A possible destination consisting of a host address and a port is called a "channel". A channel is member of a "channel group". Channels are numbered in a group starting with 0. Groups are numbered starting with 0, which is the initial default group.

Balance accepts connections on the given port and forwards them to the supplied channels. At least one channel (in the default group) must be specified. If there are two or more channels specified in a group *balance* performs a simple round-robin load balancing between the channels.

Balance allows the definition of further channel groups. The connection scheme works as follows: *balance* tries first to establish a connection to a channel in the first group (0), performing the standard round-robin load balancing scheme. If no channel in this group is available, *balance* proceeds with the next higher channel group. Groups are simply separated with a "!" at the command line at startup and can be controlled interactively with the "group" command.

A "%" instead of a "!" as a group separator declares the previous group to be of type "hash". This means that instead of a round-robin algorithm, a hash distribution based on the client ip address is used to determine the destination channel. This allows connecting one client always to the same server (e.g. balancing http sessions to a single server).

Hosts may be specified either by hostname or by IP address. Ports may be specified either by name (as listed in /etc/services) or numerically. If no port is specified in a destination, the destination port defaults to the source port that *balance* controls.

Balance allows the specification of the maximum number of connections per channel. This parameter can be optionally added after the port specification separated by a colon (":"). If a maximum number of connections is specified a channel will only be used for this maximum number of simultaneous connections. A maxc value of 0 denotes an unlimited number of connections. This is the initial default value of a channel.

The maximum number of groups and channels *balance* can handle is specified at compile time and is initially 16 channels in 16 groups.

Failover to another destination (a "channel") occurs if the connection is refused on the current channel or if the connect timeout is reached trying to establish a connection. If all possible destinations (channels) currently fail, the client connection to balance is closed.

Balance accepts the following options:

6 Forces to bind on IPv6 socket by setting hints.ai_family to AF_INET6.

a Enable autodisable option: A channel needs to be manually re-enabled after a failure.

b Bindhost: *Balance* binds to the specified host (or address) for listen() instead to INADDR_ANY.

B Bindhost: *Balance* binds to the specified host (or address) for outgoing connections (the connection will be initiated from this address).

c Command: allows to send a command to the balance master process (see interactive mode)

d Debug: *Balance* outputs debugging and tracing information messages on stderr.

H Hashfailover: *Balance* does failover to next node even if hash is used.

F Foreground: tells *balance* to stay in foreground. This might be useful for testing and debugging since *balance* can be stopped in that mode using ^C (or other interrupt character).

M Use memory mapping for IPC instead of shared memory

i Interactive Control: *Balance* connects to the running instance defined by local port and bind address via shared memory and allows to control the behaviour of it using a command line interface. The access permission using this interface are determined by the access restrictions of the shared memory segment in effect. *help or ?* prints out a short command overview, *assign* allows to change the host_port assignment of a channel (only if disabled), *create* allows to establish a new destination definition (channel) consisting of host and port in the current group, *disable* disables a channel in the current group, *enable* enables a channel again in the current group, *group* changes the current group in interactive mode where all following commands are targeted, *hash* changes the current group to be of type "Hash", *help* prints out online help informations, *kill* shuts down the master process and exits interactive mode, *maxc <channel> <maxc>* sets the maximum number of connection ot the channel (0 means infinite), *mrtg-bytes <group> <channel>* prints out the bytes received/sent in MRTG compatible format (intended to be called with -c automatically by MRTG), *mrtg-conns <group> <channel>* prints out the total connections in MRTG compatible format (intended to be called with -c automatically by MRTG), *quit* exits the interactive mode, *reset* resets the byte counters of a channel, *rr* changes the current group to be of type "Round Robin", *show* shows an overview and the status of all channels including the incoming and outgoing transfer volume in bytes. The output is sorted by groups. Additionally the current connections (c) and the maximum allowed connections (maxc) are printed, *version* prints out the version and MAXGROUPS and MAXCHANNELS constants at compile time.

p Packetdump: *Balance* shows all incoming and outgoing data on stdout using a simple always readable external representation of data. This might be useful for debugging and protocol analysis.

t Connect Timeout: the default timeout trying to establish a connection to any destination can be changed using this option. The default timeout after which a destination is regarded to be currently inaccessible is 5 seconds.

T Select Timeout: Timeout for select(), default = 0 (never). This feature is currently untested.

EXAMPLES

$ balance smtp host1.test.net host2.test.net

Connection to the local SMTP port will be forwarded alterating to the SMTP port on host1 and host2. *Balance* runs automatically in background.

$ balance -b 2001:DB8::1 80 10.1.1.1

Balance binds on port 80 of the local IPv6 IP address 2001:DB8::1 and distributes connections to the IPv4 addresses 10.1.1.1 and 10.1.1.2.

$ balance -b ::ffff:10.1.1.3 80 10.1.1.1

Balance binds on port 80 of the local IPv4 IP address 10.1.1.3 (provided in IPv6 notation) and distributes connections to the IPv4 addresses 10.1.1.1 and 10.1.1.2.

$ balance -fp imap mailserver

Connections to the local IMAP port will always be forwarded to the host "mailserver". *Balance* stays in foreground and all data is printed in readable format on stdout.

$ balance -f 8888 host1 10.1.1.1:8000

Connections to the local port 8888 are forwarded alternating to host1, port 8888 and the host 10.1.1.1, port 8000. *Balance* stays in foreground connected to the "controlling tty".

$ balance imap mailserver1::16 ! mailserver2

Two groups are specified, each containing one channel member. First up to 16 simultaneous connections are forwarded to "mailserver1". As soon as they are consumed, *balance* proceeds with the next group (1) which will consume all remaining connections forwarding them to the imap ort on "mailserver2".

$ balance pop3 host1 host2 host3

Balance does round robin load balancing for the three hosts in the default group 0 for pop3 services. If all three hosts in group 0 fail, all connections are then forwarded to the host "failover1".

$ balance telnet target.munich.net::1

Here *balance* is used to restrict all connections to exactly one at a time forwarding the telnet port.

$ balance 8888 localhost::12 ! localhost::4

This is a simple test, forming 5 groups where balance is self referencing its own services 20 times. This is simply a test which definitely can be tried at home.

BUGS

In case that *balance* is not able to forward the connection to any destination the inital connection to balance is always first accepted and then closed again immediately. This is not in every case the behaviour that would have been seen directly on the destination host.

AUTHOR

Thomas Obermair, Inlab Software GmbH (obermair@acm.org)