



FireBrick Technical Reference Manual

Andrews & Arnold/Watchfront

Table of Contents

FireBrick Technical Reference Manual	1
Basics	3
IP addresses.....	3
Public IP addresses.....	3
Private IP addresses.....	3
Funny IP addresses.....	4
Subnets.....	4
Address Ranges and notation.....	5
Ethernet and MAC.....	6
ARP.....	6
Connectors	7
.....	7
Sockets and connectors.....	7
Lights.....	7
Factory reset.....	8
DHCP mode.....	8
Stealth	9
ARP request propagation.....	9
ARP reply propagation.....	9
Broadcast packet propagation.....	10
TTL/Routing.....	10
Stealth intercept address.....	10
Stealth controls.....	11
ARP Handling	13
Generating ARP requests.....	13
Processing ARP replies.....	13
Processing ARP requests.....	13
ARP session tracking.....	14
ARP status.....	14
ICMP Errors	17
Firewall.....	17
Closed ports.....	17
Ping response.....	17
TTL.....	17
Fragment failure.....	17
ARP failure.....	17
Source IP address.....	17
Contents.....	18
Routing	19
Pre routing.....	19
Routing.....	19
Stealth IP handling.....	20
Interaction with session tracking.....	20
Interaction with port mapping.....	21
MTU.....	21
Weighted routing.....	21

Table of Contents

Routing	
Multiple gateway load sharing.....	22
Session Tracking.....	23
Timeouts.....	23
TCP sessions.....	23
UDP sessions.....	24
ICMP sessions.....	24
Other sessions.....	24
Broadcast sessions.....	24
Non sessions.....	24
Fragments.....	25
Session status.....	25
Filtering.....	27
No match.....	27
Ports.....	27
Actions.....	28
TCP.....	28
Delay.....	28
Fragments.....	28
What the FireBrick sees.....	28
Traffic from the FireBrick.....	28
Port mapping.....	29
Port Mapping.....	31
Changing target IP and port.....	31
Changing source IP.....	31
ICMP errors.....	32
Forced routing.....	32
Special port maps.....	32
Fragments.....	32
Other protocols.....	32
Address Translation.....	33
Traffic Shaping.....	35
Speed Lanes.....	35
Bandwidth sharing.....	35
Shaping rules.....	35
Buffer overflow.....	36
Changing rules.....	36
Fast ACKs.....	36
Master speed lane.....	36
Profiles.....	39
Master switch.....	39
Time controlled.....	39
Time setting profile.....	39
Ping scanning.....	39
Changes.....	40
Type of ping.....	40

Table of Contents

Profiles	
Pinging via specific interfaces.....	40
ALERT LED.....	40
Tunnels.....	41
Where a tunnel goes.....	41
What protocol tunnels use.....	41
IP Security.....	41
Authentication.....	41
Dynamic addresses.....	41
Fragments / MTU.....	42
Keep Alives.....	42
Technical Details.....	42
DHCP.....	45
DHCP Client.....	45
Checking.....	45
Routes.....	45
DHCP Server.....	46
Basic operation.....	46
Protection.....	46
Reservation.....	46
Persistence.....	46
Requested.....	47
Logging/Status.....	47
Backup operation.....	47
Restricted addresses.....	47
Sharing.....	47
Running out.....	47
No clock.....	47
Mirroring.....	48
Factory reset modes.....	48
Time.....	49
Setting the clock.....	49
Every hour.....	49
Time server.....	49
Time zones.....	49
Logging.....	51
Log options.....	51
Blink.....	51
Flash.....	51
Memory log.....	51
Syslog.....	51
Email.....	51
End-log.....	52
Filter logs.....	52
Colour coding.....	52
Log format.....	52

Table of Contents

DNS	55
DNS Relay.....	55
DNS Lookup.....	55
Statistics	57
Reporting.....	57
Ethernet counters.....	58
Security	61
Users.....	61
How it works.....	61
Nobody user.....	61
Logging.....	61
Security levels.....	61
Config save/load.....	62
Tips	63
FAQ	65
I can only see Setup and Users menus, shouldn't there be more?.....	65
I cannot access the FireBrick any more – HELP!.....	65
I have set the user to WAN access, but it just says "Goodbye".....	65
I cannot set the clock!.....	65
I cannot get FTP to work.....	66
I have set port mapping to one of my other public addresses but it does not work.....	66
I think filters are getting in the way.....	67
Limits and timeouts	69
Examples	71
ADSL/Stealth.....	71
ADSL/Stealth with external machines.....	72
ADSL/Stealth + FB address.....	73
ADSL and private network behind FireBrick.....	74
ADSL with ISDN fallback.....	75
Cable modem, with one machine having external address.....	78
Multiple ADSL lines using bonded uplink.....	79

FireBrick Technical Reference Manual

The FireBrick technical manual is designed to provide a detailed technical description of the operation of the FireBrick, and is not a beginners guide. Please read the general manual at <http://userguide.Firebrick.co.uk/> for a basic understanding of the operation of the FireBrick.

More information about the FireBrick generally can be found at <http://www.FireBrick.co.uk/>

FireBrick is a registered trademark of Andrews &Arnold Ltd and Watchfront Ltd.
E&EO Copyright © 2001 Andrews &Arnold/Watchfront

Issue 1.6, 14 May 2001



Basics

In order to understand the detailed operation of the FireBrick it is necessary to have a good basic understanding of IP networks, routing, and in particular the operation of ethernet networks, subnets and ARP. This section provides a brief overview of the basic principles, and covers the way in which these apply to the FireBrick generally.

For a more detailed introduction to IP networking, we recommend TCP/IP Illustrated, Volume 1, ISBN 0201633469 .

IP addresses

An IP address is normally shown in *dotted quad* notation, e.g. 192.168.42.1, but is in fact simply a 32 bit number. Dotted quad notation simply shows each of the bytes in the 32 bit (4 byte) address in decimal separated by a dot, so each part is 0 to 255. Normally, all devices connected using an IP network have an IP address, and a device with multiple interfaces will normally have one for each interface. As such do not consider the address to necessarily be the address of a computer, but the address of its ethernet interface.

The FireBrick allows IP addresses to be entered in dotted quad notation (or pure decimal). The FireBrick will display addresses in dotted quad notation, but has an option to show such addresses with *zero padding*. e.g. 192.168.042.001. This notation is convenient for aligning addresses above each other as it can give the address a fixed width (depending on the font). It is important to note that you cannot always use addresses with leading zeros in some applications, e.g. ping 192.168.042.001 on windows or linux will actually ping 192.168.34.1 as 042 and 001 are interpreted in octal !

Public IP addresses

Internet addresses are allocated in large blocks to regional internet registries which cover large geographic areas (e.g. Europe is covered by RIPE). Each regional registry allocates smaller blocks to local registries, typically an ISP. The local registry / ISP will assign smaller blocks or individual addresses to customers or dialup servers. Customers with small blocks then assign addresses to individual devices on their network.

As such, IP addresses are globally unique, and the job of your ISP is to ensure that any packet anywhere in the world sent to one of your addresses will reach your network. Similarly they ensure packets you send will reach the destination. They interconnect (peer) with other ISPs to exchange traffic destined for globally unique IP addresses. It is important to note that IP addresses do not belong to the end users, they are allocated to ISPs and assigned to users, but if you change ISP you will normally have to change IP addresses.

Private IP addresses

Global public IP addresses are limited, simply because there are only so many 32 bit numbers! As such they are managed carefully, and networks that do not need to have every machine addressable from the public internet do not need global public IP addresses. As such a number of addresses have been allocated for use in private networks. These private IP addresses will never be allocated to anyone, and so their use can never clash with publicly available addresses. Private addressed networks can still communicate with machines on the internet, but only via address translation or proxy as the private addresses are not valid on the public internet.

If you make a private network and need addresses, you must only ever use these reserved private addresses, otherwise the address you have picked may belong to someone, and you will effectively mask access to those addresses from your network. When setting up subnets on the FireBrick, the address block will be noted as public or private automatically as a reminder.

Private IP address ranges

Address block	First	Last
10.0.0.0/8	10.0.0.0	10.255.255.255
172.16.0.0/12	172.16.0.0	172.31.255.255
192.168.0.0/16	192.168.0.0	192.168.255.255

Funny IP addresses

IP addresses up to 223.255.255.255 are used for normal routed IP networking, from 224.0.0.0 they are special. You will find addresses 224.0.0.X often occurring on networks with large routers, as these are group addresses used to locate types of machines (e.g. *any router*). Unless you are using complex inter-router protocols, you can disregard (and drop) such packets in your FireBrick configuration.

The address 255.255.255.255 is a special address which means a local network broadcast. It is received by all machines on your subnet and is not routed to your ISP or other subnets.

It is worth noting that 127.X.X.X is also special in that it is normally directed to a local loop back interface. 127.0.0.1 is an address for yourself from most computers. It is not treated specially by the FireBrick in any way.

Subnets

A subnet is a group of IP addresses which are normally on one ethernet network. The subnet is defined using a netmask. This is a mask of bits in the IP address which define the part of the address that two IP addresses must have in common to be considered in the same subnet. As such, all devices on a subnet must have different IP addresses and the same subnet mask in order to work together.

You can have more than one subnet on the same actual ethernet network, but machines on one subnet cannot normally communicate with machines on the other subnet without using an intermediate device (a router) just as if they were in fact on two different networks. This is not a secure way or separating machines though as they could change IP address to fit in with a different subnet.

If communicating with a device on the same subnet, an ARP (see below) is used to talk to that device directly. If talking to a device on another subnet, then an ARP is used to locate a router which can talk to that other subnet.

The first and last addresses in a subnet are special in that they are treated as network and broadcast address and so cannot be used for devices on the subnet. This makes very small subnets (e.g. 4 address blocks) very inefficient.

Address Ranges and notation

A subnet is defined by a netmask, and this can be shown in a number of ways. e.g. 192.168.42.1 with a subnet of 255.255.255.0 means a range of addresses 192.168.42.0 to 192.168.42.255 are all on the same subnet. Where the subnet is only 255s and 0s, this simply means that the part of the IP addresses where there is a 255 must be the same and the part where there is a 0 may be different. So, 192.168.42.1 with subnet 255.255.0.0 is 192.168.0.0 to 192.168.255.255.

A subnet mask is always a number of bits which are *common* between IPs in the same subnet, and the remainder which can be different. So sometimes this is shown as a bit count. e.g. 192.168.42.1/24 means 24 bits of subnet, or a mask of 255.255.255.0.

Where the subnet mask is not a multiple of 8 (i.e. whole bytes), the mask is more complex, e.g. 192.168.42.1 netmask 255.255.255.240 is 28 bits and gives the range 192.168.42.0 to 192.168.42.15 (i.e. 16 addresses).

The FireBrick has a number of places where a range of addresses can be entered. In these cases the range need not be an exact subnet, so you could enter 192.168.42.1 to 192.168.42.5 for example. You can however enter a subnet mask in the second box (addresses that are valid subnet masks from 128.0.0.0 to 255.255.255.252 are treated as such), so you could enter 192.168.42.0 and 255.255.255.240 and get 192.168.42.0–15. You could also enter the bit count (any IP less than 0.0.0.33 is treated as a bit count and can be entered without the leading 0.0.0.).

So, the same range 192.168.42.0–15 could be entered as 192.168.42.0 to 192.168.42.15, or 192.168.42.0 and 28, or 192.168.42.0 and 255.255.255.240. Note that entering a mask or bit count will make the range start and end at the right point, so the above could be entered as 192.168.42.5 and 28 with the same effect (but 192.168.42.17 would be 192.168.42.16–31 as 17 is in the next block of 16).

The exception is the entry of subnets in the FireBrick where the second field must be a subnet, rather than just the end of a range, and the first part is the FireBrick's own IP address.

The FireBrick will normally display ranges of IPs using a hyphen, e.g. 192.168.42.0–15 or 192.168.0.0–255.255. If set to pad with leading zeros then it displays the range in full one line above the other, e.g. 192.168.042.000– above 192.168.042.015.

When displaying subnet settings, the bit count is shown as well as the actual IP used on the subnet, e.g. 192.168.42.1/28.

Use of common bit counts and subnet masks

Bit count	Subnet mask	Number of addresses
8	255.0.0.0	16,777,216
16	255.255.0.0	65,536
17	255.255.128.0	32,768
18	255.255.192.0	16,384
19	255.255.224.0	8,192
20	255.255.240.0	4,096
21	255.255.248.0	2,048
22	255.255.252.0	1,024

23	255.255.254.0	512
24	255.255.255.0	256
25	255.255.255.128	128
26	255.255.255.192	64
27	255.255.255.224	32
28	255.255.255.240	16
29	255.255.255.248	8
30	255.255.255.252	4

Ethernet and MAC

An ethernet network provides a way for a number of devices to communicate in a local area network. This normally uses 10/100baseT RJ45 connectors and hubs or switches, but could include thick or thin ethernet. The collection of devices sharing the same group of hubs and switches are all one ethernet.

Packets sent on the ethernet are basically either broadcast (every device gets it) or unicast (one device gets it). The later works by addressing the packet to a specific MAC (Media Access Control) address. All ethernet devices have a manufacturer defined unique MAC address, and you should never encounter a duplicate.

MAC addresses are 48 bit (6 bytes) and unlike IP addresses they are normally shown in hex. You may see them with colons between the bytes, e.g. 00:03:97:FF:00:09 but the FireBrick shows this as straight hex, e.g. 000397FF0009.

When sending packets to other devices on the same subnet it is assumed that the device is on the same ethernet. This is a key assumption in routing and ARP is used to find the MAC of the device with the require IP, and then packets are directed to that device.

When sending packets to other devices, a gateway or router is needed. ARP is used to find the MAC address of this gateway and the packet sent to that MAC address.

The FireBrick can display its cache of where it can see different MAC addresses on each of its two ethernet interfaces.

ARP

ARP is a system of finding MAC addresses. A broadcast message is sent asking a machine with a specified IP address to identify itself, and that machine replies. Devices will cache ARP replies and keep a note of where machines are for a limited period of time.

The FireBrick can display its ARP cache, showing the MAC address of IP addresses it has requested via ARP, or devices from which it has seen ARP replies.

Connectors



Sockets and connectors

Front	Left	WAN port	10baseT connection for WAN configured as a host (i.e. straight lead to a hub)
	Right	4 LAN ports	10baseT network hub for LAN (i.e. straight lead to PCs)
Rear	Left	9 way D	Serial connection for factory test May be used for other functions in future software
	RFar right	Power	Connection for power, 9–24V DC centre positive

Lights

WAN	Yellow	Network activity on WAN
	Green	Link active on WAN
Front	ALERT	Blinks briefly on power up Blinks rapidly for <i>blink</i> filtering rules Blinks slowly for <i>flash</i> alert status having been set Solid on if profile set to affect ALERT LED
	POWER	On when power applied Blinks off when reset (e.g. software upload, factory reset)
LAN	Yellow	Normally network activity – can be configured for other usage
	Green	Normal link active – can be configured for other usage

The 8 LAN hub LEDs can be configured to operate in the following modes :-

Yellow	Green	Note
Partition	Link	Link off if partitioned
Activity	Link	
Partition Blink for collision	Link Blink for activity	
Partition	Link Blink for activity	LEDs flash at reset
Partition	Link Blink for activity	
Collision	Link Blink for partition	LEDs on for 4 seconds on reset
Bar graph of LAN activity		All 8 LEDs make up bar

	graph
Cycling lights	Same as when all lines unplugged

Note that if manually selected as cycling lights, then a time stamp reply packet is sent on the WAN every few seconds. This is a harmless packet and is used to allow multiple FireBricks to be synchronized if all connected to the same WAN hub. The LED selection screen shows a number of links (shown as dots) which allow the phase of the LEDs to be set allowing effects such as a "Mexican Wave" to be created. Note that this requires ICMP traffic from WAN to FireBrick from and to the FireBricks LAN stealth IP address to be allowed (create a filter).

Factory reset

To factory reset your FireBrick®, follow these steps :-

1. Remove power and all network connections
2. Connect a straight network lead (such as the one provided with the FireBrick®) from the WAN (left) to the right hand LAN port.
3. Power on the FireBrick®
4. Observe that the link light on the LAN port is lit and on the WAN port blinks rapidly.
5. After a moment the red ALERT comes on.
6. Remove the network cable with the power still connected
7. The FireBrick® will reset (green and red LEDs go off for a moment), and then start as normal with cycling lights on the hub.

The factory reset is now complete – bear in mind that the FireBrick is now operating on the stealth IP as detailed in getting started.

DHCP mode

Normally, the FireBrick does not have DHCP enabled after factory reset. However, an alternative factory reset procedure will reset the FireBrick and enable DHCP. Simply connect the loop back cable to a different one of the WAN ports in the above procedure :-

Hub port	Factory reset operation
0 (left)	DHCP server on LAN and DHCP client on WAN
1	DHCP server on LAN
2	DHCP client on WAN
3 (right)	Normal, non DHCP mode

Stealth

The FireBrick can operate in stealth mode – which basically means it is invisible. Obviously it cannot be completely invisible, otherwise it would be no good as a firewall. The key part of stealth mode is that devices on one side of the FireBrick can send packets to devices on the other side of the FireBrick without knowing the FireBrick is there – i.e. using MAC addresses as normal. To do this the FireBrick propagates ARP requests and replies, allowing devices on one side to locate devices on the other side as normal. All IP packets passing through the Firebrick are subject to its normal session tracking, filtering, and port mapping rules.

As such the FireBrick is transparent subject to the following caveats :-

1. Runt, overlong, bad CRC, and otherwise deformed frames are dropped
2. 802.3 style ethernet packets are changed to standard ethernet packets (removal of 802.3 headers)
3. Packets which are not IP or ARP (request and reply) are discarded. In particular, this means IPX and NETBEUI are not passed through the FireBrick
4. As the FireBrick has two half duplex 10baseT interfaces, this can result in additional collisions and packet loss when running at very high throughputs (e.g. 900KB/s+) which means the FireBrick is not quite the same as having a hub.
5. The FireBrick maintains a MAC cache, recording which side of the FireBrick each MAC has been seen. Packets which are known to be destined for MAC addresses on the same side that they were received are dropped. This is much the same as having a switching hub in front of the FireBrick.
6. ARP requests and replies are specially processed
7. All IP packets are subject to session tracking, filtering, and port mapping rules
8. The FireBrick introduces around 1ms additional delay, depending on packet size, with an extra 2ms to 3ms on the initial packets in a session.
9. The FireBrick intercepts traffic to a specific stealth IP address per interface, allowing the <http://my.FireBrick.co.uk/> set up pages to operate.

ARP request propagation

In order for the FireBrick to operate in a stealth mode, it is necessary for ARP requests to be passed from one interface to another. This then allows devices on each interface to address devices on the other interface by MAC address as if the FireBrick was not there and they were on the same LAN.

After factory reset the FireBrick is in a full stealth mode. This means ARP requests are passed to the other side. However the FireBrick can operate in a partial stealth mode where it has some subnets defined. The subnet defines the IP and netmask to be used on a specific interface, and the FireBrick can have many subnets defined. Subnets can be marked as stealth. Note that subnets are subject to profiles, so may only be operational at certain times.

The exact rule for passing ARP requests is described in ARP handling.

ARP reply propagation

As devices will normally cache ARP replies regardless of whether they requested them, incorrect propagation of ARP replies could cause ARP poisoning (i.e. incorrect ARP data in ARP caches). As described in ARP handling, ARP requests that are passed through are session tracked, and only valid replies passed back.

Broadcast packet propagation

A broadcast packet is one sent to the broadcast MAC address.

If the target IP is the network or broadcast IP of a subnet on the interface from which the packet was received, and the subnet is not marked stealth, then the FireBrick considers this a packet for itself and does not propagate it.

If the target IP is 255.255.255.255 and either we have no IP on a DHCP client subnet, or there is no source IP, or we have a subnet where the source IP is within the subnet, then the FireBrick considers the packet for itself and does not propagate it.

Otherwise broadcast packets are propagated subject to global stealth controls.

TTL/Routing

When packets are propagated in stealth mode, they are not treated as routed packets and as such the routing rules in the FireBrick do not apply. Routing only applies where the FireBrick must decide where to send a packet, but stealth packets already have an identified target MAC address and so do not require routing.

Packets to the FireBrick itself (by MAC, or broadcast MAC as above) are treated as routed, although port mapping can be used to force a stealth packet to be routed. Only routed packets will cause the TTL to be counted down and a time exceeded ICMP to be generated if this reaches zero.

Stealth intercept address

The FireBrick has a *stealth address* which is pre-programmed for the LAN interface. This is by default 217.169.0.1 which is a public IP address assigned to the FireBrick and so should never clash with any other network.

This address is used to allow communication with a stealth FireBrick. The FireBrick will answer ARP requests on the LAN for this address, and accept traffic to this address (even stealth traffic) from the LAN. Both explicit and implicit filtering rules allow this traffic for web page configuration of the FireBrick. This address can be changed, but this is not normally necessary.

The address has a public DNS entry of my.firebrick.co.uk, which allows configuration using a web browser to this address. On the internet this address has a page explaining that you have not reached your FireBrick, in case it is used when a FireBrick is not in place. Formerly the FireBrick had 62.190.255.253, and this is now old.firebrick.co.uk, allowing access to older FireBricks which have not been reconfigured with the new address.

When answering TCP traffic on this address, the FireBrick will use the external router MAC address in its replies, hence remaining in complete stealth mode.

There is also an option for a WAN stealth address. This is an address which is only used where the FireBrick must generate traffic on the WAN and where no subnet exists. This is typical of a full stealth configuration, and allows the FireBrick to initial time setting requests, and even send emails. The address must be the address of a machine on the LAN so that traffic for the address will reach the FireBrick from the WAN side. Also the address must be of a machine that is power on so that the router on the WAN can ARP the address as the FireBrick does not presume to reply for the machine.

Apart from intercepting specific replies to its active outgoing requests, the FireBrick does not otherwise interfere with traffic to the address it is borrowing.

Stealth controls

Global controls in the log/filter options allow stealth to be disabled :-

1. The default passing through of ARP requests can be disabled. i.e. only ARPs within explicit stealth subnets are passed through.
2. The passing through of subnet broadcast packets can be disabled.
3. The passing through of local broadcast (255.255.255.255) packets can be disabled.
4. The passing of any stealth packets can be completely disabled (stopping all of the above as well).

In all cases, such disabled packets are dropped silently before any filtering or logging.

ARP Handling

The FireBrick maintains an ARP cache for each interface, recording the mapping of IP addresses to MAC addresses. The cache is independent for each interface, and within each cache an IP address will only have one MAC address.

The ARP cache is used to allow routing to IP addresses for routed traffic and traffic generated by the FireBrick itself.

Generating ARP requests

The FireBrick will generate an ARP request if it needs to know the MAC address for an IP address. This is only necessary for routed traffic.

The request is NOT generated if the FireBrick cannot complete the source IP/MAC information. To complete the source information the FireBrick checks :-

1. If it has a subnet on the appropriate interface appropriate to the requested IP, then it will use its IP on that subnet and its MAC address
2. If not, but the FireBrick has a stealth IP address for that interface, and has noted a MAC for that IP, then it will use that IP and MAC
3. If not, but the FireBrick has a stealth IP address for that interface, then it will use that IP and its own MAC
4. If not, then the ARP is not sent

Note, subnets are searched in order for the first match.

While waiting for a reply, a limited number of packets requiring the MAC address are queued. On timeout these packets are discarded and a "no route to host" ICMP generated. On reply, these packets are sent with the MAC address completed.

An ARP request is sent on the first packet requiring a MAC address, and repeated every few seconds for several tries.

Processing ARP replies

Any ARP reply causes the sender IP/MAC to be stored. This may complete a pending ARP request and cause pending packets to be sent.

1. ARP replies that are targeted at a MAC known to be on the same interface are discarded as local traffic.
2. ARP replies that are effectively ARP announcements (to self and target broadcast MAC) are simply logged (debug level)
3. Otherwise, ARP replies that match a previous passed through ARP request are passed back through to the originator
4. Otherwise the ARP reply is simply logged (debug level) as unexpected

Processing ARP requests

Any ARP request causes the sender IP/MAC to be stored. This may complete a pending ARP request and cause pending packets to be sent.

ARP requests received are checked as follows. The result may be to discard, or to reply or to pass through.

1. If the request is for the FireBrick's stealth IP on the LAN and is from the LAN, then a reply is sent
2. If the request is for the FireBrick's IP address on a subnet on the sending interface then a reply is sent
3. Routes are checked for a route where the source interfaces matches the source of the ARP request, and the source IPs (if any) cover the sender IP of the ARP and the target IPs cover the requested address of the ARP and proxy ARP is set, then a reply is sent
4. If the request is from and for an address in a subnet on the interface from which it came which is marked stealth, then it may be passed through.
5. If not, then if it is for an address in a subnet on the interface from which it came which is not marked stealth, then it is dropped
6. If the packet would be passed through, and matches one of the FireBricks own address on the far side interface, then a reply is sent
7. If not replied or dropped, then the ARP is passed through (subject to global stealth settings) and a session created for the reply to be checked.

The reply quotes the FireBrick's MAC for the interface on which the ARP was received, even if it is for a far IP side address.

ARP requests that are passed through are stored in a session table (discarding the oldest session if the table is full).

ARP session tracking

ARP requests that are passed through are session tracked. A limited number of outstanding ARP requests are tracked, with the entry with the shortest remaining timeout being discarded if there are more than this number received. When an ARP reply is received that matches the request then it is passed back to the originator.

The ARP session is only held open for a limited time, and normally ARP replies are received within milliseconds of the request being sent. However, it is possible for a host to send several requests in rapid sequence without waiting for replies. This can then lead to several replies (i.e. request request reply reply, rather than request reply request reply). As such, when an ARP reply is matched, the session remains in place for a short period of time to pick up further replies and pass these through.

ARP status

The diagnostic display page on the FireBrick will show the current state of the ARP cache, with an indication of the status of each entry as follows :-

ARP status

Yellow	Cache OK, but being retried
Green	Cache OK, and recently used
Blue	Cache OK, not recently used
Red	Cache not OK, waiting for initial reply

The *used* status of the cache is set whenever the FireBrick uses the ARP entry to send a packet. This is cleared periodically, and the cache refreshed. As such, entries being regularly used will appear as Green or Blue. If a cache entries is not used, then it will be discarded. Yellow or Red will

only show very briefly, or if there is no reply from an ARP request. Red entries do not show a MAC, just zeros.

ICMP Errors

The FireBrick can generate ICMP packets for a number of reasons.

Firewall

Packets that reach a "Reject" filter cause an ICMP error to be generated. This indicates a host unreachable error 13 (admin prohibited filter). Such packets are delayed by a random amount of time to reduce load as a result of attacks.

Pings to a "bounce" filter result in ping responses, but these are also delayed.

Closed ports

Packets for the FireBrick for which there is no internal handling (or port map) result in a port unreachable ICMP error. The exception to this is TCP traffic, which indicates a RST. Such ICMP packets are delayed by a random amount of time to reduce load as a result of port scans.

Ping response

The FireBrick will generate ping responses to a ping to one of its addresses. These packets are not delayed.

TTL

The FireBrick will count down TTL if acting on a routed packet, and can generate an ICMP TTL Expire error. These packets are not delayed.

Fragment failure

If a fragment cannot be made for a packet which needs to go via a smaller MTU (e.g. down a tunnel with DF bit set), then a *Cannot Frag* error is returned. This is not delayed.

ARP failure

If the FireBrick is unable to identify the MAC address for a routed packet, it will return a "host unreachable" error. This is not delayed.

Source IP address

ICMP errors are normally sent from the router that generated them, and as such the FireBrick is expected to identify itself. Many types of error result only from routed connections, and so the FireBrick will have a suitable address.

The FireBrick will use its normal rules to work out a suitable source address, and if that fails will not send the error. This means that "reject" is often ineffective in a complete stealth mode as the FireBrick has no IP to quote in the reject message.

1. If the FireBrick is sending the packet to an address on a subnet, then that subnet is used as the IP

2. If not, and the FireBrick has a stealth IP on the interface in question, that is used
3. If not, the packet is not sent

Contents

The packet contents includes, as per ICMP protocol, details of the original packet including the IP header and up to 8 bytes of payload. If the packet was processed via NAT or port mapping, then this is reversed as normal, including changing the payload IPs, ports and checksums, so as to ensure the recipient can associate the packet with a session correctly.

Routing

Routing is the process whereby the FireBrick works out where to send a packet. This involves working out which interface to send the packet to (WAN, LAN, Tunnel, Itself) and for ethernet interfaces, which MAC address to send the packet to.

For ethernet, the MAC address is based on an IP address – either of the destination of the packet (if on the same subnet as the FireBrick) or of the router/gateway which will pass on the packet to its destination. ARP is used to find the MAC from the gateway IP address.

Routing only applies where the FireBrick needs to find where a packet must go. A stealth packet already has a destination interface (the other side from which it came) and a destination MAC (as received). As such, stealth packets are not routed (see port mapping for details of how to force routing of stealth packets).

Pre routing

Before routing is done, all packets (received or to be sent) arrive in the routing queue. These may possibly have a pre-decided target interface (e.g. TCP replies from the FireBrick itself go back the the source interface, etc).

Packets from tunnels or serial links are set for target *unknown* and the routing decides. Packets from the ethernet interfaces has the following choices :-

1. Checking by target MAC, packets that are for the same interface on which they arrive are dropped
2. Checking by target MAC, packets for the other interface are set for that interface, marked as stealth and marked as cross interface.
3. Checking by target MAC, packets for unknown targets are set to the other interface and marked as stealth
4. Packets for a broadcast MAC are checked if they are to be propagated or handled by the FireBrick as per the stealth rules.
5. Packets for the FireBrick's MAC on the interface they arrive are checked if they are for the for the FireBricks IP on that interface, and if so directed to the FireBrick itself, otherwise they are set to *unknown* allowing unrestricted routing

Routing

To decide where a packet has to go, the FireBrick considers the following, in order. Packets may have a target interface pre-decided (as above), in which case only rules setting that interface are considered, and these rules may set the gateway (for ethernet interfaces) or specific sub-interface (for tunnels/serial).

1. Packets for the FireBrick itself do not have any further routing as they can only end up for the FireBrick itself.
2. The routing table, in order. Each route is checked for a match, considering the source and destination IP addresses and the source interface. The first match is used. If the target interface is already decided, then only routes to that interface are considered. The routing entry found will define the destination interface and may also define a destination gateway address. The routing entry can also set the NAT option. The target can be *any* in which case the target interface is not set and further routes are considered (useful to set NAT

- addresses, or force proxy ARP without affecting routing decisions).
3. At a configurable point during routing, routes to the subnets are checked. This is at the end by default.
 4. If no route was found (other than target *any*) or the route was to a subnet, then the subnets are checked to see if the packet is from a subnet for which NAT is defined. The first subnet matched is checked for the NAT option. (See address translation for how this is used). Either routes or subnets can set the NAT option.
 5. If no gateway is yet defined then routing to a subnet is considered again. If a destination interface has been defined, then the search is restricted to subnets on that interface. The first subnet encompassing the destination IP is considered – setting the gateway as the target IP directly.
 6. If there is still no gateway, then the default route is considered. If the destination interface has already been decided, then this is only considered if it is on the same interface. The default gateway and interface are set.
 7. If there is still no gateway, and the packet is destined for an ethernet interface, then the target IP is assumed to be the gateway.
 8. Having established a gateway, the subnets are checked to identify which subnet the packet is being sent on based on the gateway address. A gateway would normally have to be on a valid subnet for the FireBrick to be able to send ARPs for the gateway. This check also confirms if the destination is the network or broadcast address of a subnet, in which case it is sent to the broadcast address. It is also checked if the destination is the FireBrick's own address on the subnet, in which case it is destined for the FireBrick itself.
 9. If no valid target interface can be established, then the packet is dropped

This means that the routing tables are the highest priority. If a routing rule exists for a packet, then this decides the target interface and gateway and also determines if NAT applies. Only if there is no route, are routing via subnets considered, and only if there are no matching subnets is the default gateway considered.

Some types of routed packets may have the interface and possibly the gateway predefined (profile pings for example). In this case only rules with the same target interface are considered in the routing. Similarly, routing rules can set a target interface without a gateway, in which case subnets and possibly the default route is considered if it has a matching target interface.

For non ethernet interfaces (e.g. tunnels), the sub-interface (which tunnel) is specified in the routing rules. In such cases no gateway is needed as gateways are an ethernet feature.

Stealth IP handling

The FireBrick has two stealth IP addresses, one for each interface. By default there is only one defined, for the LAN, which is the address of my.FireBrick.co.uk. This is used to allow access from the LAN to the FireBrick web config pages. If a packet is received on the LAN for this address, then the FireBrick will answer any ARP for this and will accept traffic to this address as to the FireBrick.

Traffic to the WAN stealth is not handled in the same way – it is not accepted as traffic to the FireBrick itself and does not answer ARP requests. However, this is used as the reply address for outgoing traffic from the FireBrick for functions such as time setting – so a machine on the LAN side should reply to ARPs, and the FireBrick will then intercept these replies as part of the session tracking.

Interaction with session tracking

A packet may be stealth or routed. Only non stealth packets are subject to routing. At the start of a

session, the first packet is considered, and routed if not stealth. It is then checked against the port mapping. If port mapping applies then the packet is made routed regardless.

If this first packet ends up as a routed packet then the session is marked as routed. All packets through the session, forward and reverse, are then changed to non stealth.

The session holds the routing information (target interface, and gateway for ethernet). The first reply packet that is not stealth causes routing to be done on the reply – constrained to the known target interface (where the original packet in the session came from) and the gateway recorded for future non stealth packets.

Routed sessions are shown on the session display page with an R next to the protocol.

Interaction with port mapping

Port mapping applies on the initial forward packet after routing, but will cause routing to be re-applied after the port map (possibly restricted by the target interface of the port map). This is then the routing applied to all forward packets in the same session.

In the reverse direction the portmapping is reversed before the routing of the reverse packet is decided. Again, the routing is only done on the first packet in that direction and that routing used for all further reverse packets in the same session.

MTU

Ethernet interfaces operate with the full 1500 byte MTU, but some interfaces are smaller (such as tunnels which are 1455 to allow for tunnel headers). Serial interfaces may also negotiate a smaller MTU. As the packet is sent to the target interface, it is checked against the MTU.

If the packet is too large and has DF set, then it is dropped and an ICMP fragmentation error returned. Otherwise it is fragmented, dividing in to reasonably equal size blocks to minimise re-fragmentation later. This does involve additional copying of data which is not normally necessary and so does add some small additional latency. Generally it is best to try and avoid fragments.

If the packet is already a fragment then it is fragmented further regardless of the DF bit.

Weighted routing

Each route can be assigned a percentage, default 100%. This is used to determine the chance of a packet being routed via a specific route. This might be used to route different traffic via different gateways. The decision is made when the route is determined – i.e. when the session is established. All traffic on a session then follows that route (as it must, because a different route probable uses different IP addresses and NAT).

The way it works is that each routing decision, a number is picked (pseudo random) from 0 to 99. As each route is considered, if it is valid in all other respects then the percentage is checked. If the percentage is more than the *number picked* then the route is taken as normal. If the percentage is less than or equal to the *number* then the number is decreased by that percentage.

This means you can make two routes each marked 50%, and matching traffic will always go down one or the other. Similarly you could have three routes marked 40% 30% 30%. You can obviously leave the last route at 100% if you wish.

Multiple gateway load sharing

The FireBrick Plus allows for multi-gateway load sharing. To use this you will need multiple external gateways such as multiple ADSL lines. If these are on different subnets then you should set each as a subnet on the FireBrick. Pick a gateway address on the subnet for the external link on which you want the replies to arrive (e.g. if you had a 2Mb and 500K ADSL you would want to use an address on the subnet for the 2Mb line). This should be an unused address, and can be the subnet network address as it is simply used to tell the firebrick to use the multi-gateway list. Then complete up to 4 gateways.

When any traffic is directed to the gateway that is the default gateway, it will have its gateway changes at the last moment, on a per packet basis, to one of the 4 gateways you have listed in a simple cycle. This allows you to bond multiple uplinks on ADSL lines for example.

Session Tracking

Session tracking is key to the operation of the FireBrick. It means that filters can be set up allowing traffic in one direction, and the reply traffic is implicitly allowed.

Session tracking involves the FireBrick following the replies to packets that are sent. The complexity of this depends on the protocol. However, a session is tracked based on one or more of :-

1. The IP protocol
2. The source IP
3. The target IP
4. For TCP/UDP, the source port
5. For TCP/UDP, the target port
6. For ICMP such as ping the ID
7. The source interface

One of the jobs of session tracking is to record the decision of the routing function, and so determine the target interface and gateway.

Also, session tracking provides the means for port mapping and address translation allowing the IP and ports to be changed on the fly.

The FireBrick can track a limited number of sessions. For TCP the type of packets affect the session tracking – i.e. a RST (reset) will cause a session to be closed quickly as it has finished. Generally sessions are managed by time outs. The two key time outs (initial and ongoing) have per protocol defaults, but are also set using filters. How these are used are defined below. If the FireBrick runs out of sessions, then the packet is dropped.

Checking against an existing session is performed before filtering. If a packet matches an existing session then it is always allowed. If not then routing and filtering are considered and if acceptable a session is established.

Timeouts

In general, a session is established by a packet being sent. This is considered to be *one-sided* until a matching reply packet is received. While *one-sided*, the session timeout is the *initial* timeout. Once a packet is sent both ways then the session timeout is the *ongoing* timeout. If a session times out, then it is discarded, and any new packets are considered to be a new session.

ICMP error messages are normally associated with the corresponding session (for TCP/UDP and ICMP), and result in a shorter timeout applying until some other traffic is received.

The actual timeouts are defined in the limits appendix.

TCP sessions

In addition to the initial and ongoing timeouts, the TCP session tracking checks for a FIN sent each way, and an apparent ACK to a FIN (e.g. an ACK received one way after a FIN sent the other way). Sequence numbers are not checked for this, so packets being exchanged at the same time could mean the session tracking thinks a FIN has been ACKed when it has not. Once a FIN and apparent ACK are received both ways then a shorter timeout applies as the session is assumed to be closed.

Also, the session tracking follows any RST being sent either way. If a RST has been sent then a shorter timeout applies.

As TCP sessions are normally answered quickly, but may continue for hours without exchanging traffic (e.g. telnet), the initial timeout is normally short and the ongoing timeout is long.

Session tracking of TCP takes in to account the source and target port numbers.

UDP sessions

UDP sessions simply track the one-sided initial timeout and the ongoing timeout. As UDP is normally a transaction based message with a single response (e.g. DNS), the initial timeout is normally long and the ongoing timeout short.

Session tracking for UDP takes in to account both source and target port numbers. A UDP reply which is to the original source port but from a different target port will not match and will be treated as a different session. This can affect some protocols such as TFTP.

ICMP sessions

ICMP non errors, e.g. pings, are also session tracked. Only the initial and ongoing timeout states are considered.

The ICMP ID word is used for session tracking, and allows a sequence of pings to be treated as a single session.

Other sessions

Other IP protocols are session tracked simply on the source and target IP, as the content will depend on the protocol. The initial and ongoing timeout states are tracked, but these have long timeouts.

Filters can be set up to allow, and define the appropriate timeouts for specific protocols as necessary.

Broadcast sessions

ICMP and UDP session handling caters for the possibility that a session may be to a broadcast destination, and receive a reply from a specific IP. This is typical of DHCP requests and replies. A specific IP source will match as a reply to a broadcast destination but the ports have to match as normal.

Non sessions

Filtering also allows sessions to be flagged as *bypass*. This means the session is considered valid for only the one packet.

This is mainly aimed at port scans, or other situations where many thousands of sessions may be created and not properly closed.

The effect is that the reply packet is treated as a new session also, and so may fail filters unless allowed. Also the packets will be much slower at being processed as every packet is checked for

filtering and possibly routing rules.

Note, that this must not be used for TCP traffic to/from the FireBrick itself as this will break the TCP stack which follows the session tracking and so stop the operation (e.g. web page access to the FireBrick itself). You have been warned.

Fragments

Session tracking will note the initial fragment which matches a session, and allow subsequent fragments with the same IP addresses and IP ID. However, only one "last fragment" is tracked per session, so intermixed fragments will not be automatically passed through sessions. Also, if the initial fragment is not received first, fragments are also not recognized as part of a session.

If a fragment is not recognized as part of a session, it is processed through filtering normally, and if allowed is sent without establishing a new session. As no session is established for misplaced fragments, they cannot be processed via port mapping rules.

Try to avoid fragments.

Session status

The session status shows the current sessions meeting a specific criteria. This is available from speed lanes, filters, or by protocol. It is also possible to list all sessions, and sessions with over 1MB or 10MB of data transfer so far. The display is only available where diagnostic view rights exist, and sessions can only be killed where diagnostic edit rights exist. The user will only see sessions where the applicable filter is one to which they have view rights (setup view rights for the default filter sessions).

The display shows the protocol, source and target ip/port/interface, filter, and speed lane applicable. It also shows the number of bytes each way, and where any mapping has been performed it shows the mapped to source and target ip and ports. A link from the session table allows an active session to be killed.

Filtering

Traffic filtering is the main job of any firewall – it defines what is allowed and what is not allowed.

The filtering rules for the FireBrick apply when setting up a new session. Once a session is established, the reply traffic, and further forward traffic on that session are permitted without re-checking the filtering rules.

Packets are checked as follows :-

1. Session tracking to confirm if part of an established session
2. Routing if not stealth
3. Filtering
4. Port mapping, and re-routing

Filtering itself operates on the following criteria, finding the first filter which matches, and then applying the various flags and options of that filter. A filter can be out of profile or suspended, in which case it is skipped :-

1. Source interface – this is where the packet is from
2. Target interface – this is where a stealth packet it to, or where routing has determined the packets destination
3. TOS (mask and value check)
4. Protocol
5. Source IP
6. Target IP
7. For UDP/TCP, source port
8. For UDP/TCP, target port
9. For ICMP, type and code
10. For TCP, an optional SYN check (i.e. TCP packet must have SYN and not ACK to pass)

Having found a filter, the filter itself can dictate the applicable session tracking timeout, and also the basic action. The filter also dictates the logging options.

No match

If no filter matches, then :-

1. Packets from the FireBrick are silently allowed
2. Packets to the FireBrick from the LAN are silently allowed
3. TCP packets without SYN but with FIN or RST are silently dropped as these are normally trailing packets from closed TCP sessions
4. Other packets are processed using the default filter rules in the log/filter options

Ports

If ports are specified in a filter, then they apply as a port range for TCP or UDP traffic, and as code and type ranges for ICMP traffic. However, if the protocol ANY is used and port ranges are specified, then the protocol must be TCP or UDP only. ICMP, and other protocols will not match. This is because it makes sense to have port ranges for either TCP or UDP as many protocols have the same ports for both – however there is no correlation with TCP/UDP ports and ICMP type/codes.

Actions

There are 4 basic actions :-

Drop	The packet is simply ignored
Allow	The packet is allowed, and a session created
Reject	An ICMP admin prohibited filter message is returned
Bounce	For echo, an echo reply is returned For TCP SYN+noACK, a SYN+ACK with window 0 and MSS=0 is returned Otherwise dropped

TCP

TCP has some special handling. The filter can have a SYN-ONLY flag meaning that a filter will only match if SYN+noACK is set.

A filter will in any case only match if FIN and RST are not set. This allows a session to be resumed after a timeout (if there is no address translation) by further normal data in the direction allowed by the filter.

If rejecting a TCP request which is not SYN, then a RST is sent if the packet is routed or known to be for a MAC on the far side.

Delay

Reject, RST, and bounce replies are all delayed so as to ensure that these are low priority packets. For example, this will give randomly delayed response times for pings if the FireBrick is set to bounce pings.

Fragments

Fragments have to be filtered if they cannot be matched to a session. If a fragment is checked against a filter, then the filter must not require a specific port range. i.e. a filter for UDP/TCP which can contain a port range will not match a fragment. This is simply because a fragment will not include the port numbers to be checked.

What the FireBrick sees

It is worth bearing in mind that the FireBrick will only see packets that it is sent. If you have a FireBrick in stealth mode, then attempts to send packets to machines you don't have will result in failed ARPs which are not logged. This means the packet which you may well have wanted to log will not actually ever be seen by the FireBrick and so not logged. You can arrange proxy ARPs for this if you want to log everything.

Traffic from the FireBrick

It is important to note that traffic from the FireBrick does not have a source address set when initially filtered. As such, checking for traffic from the FireBrick should always be done using the source interface rather than checking for a FireBrick source IP address. The source address is set before port mapping.

Port mapping

Filtering is done before port mapping, and as such the filter settings must reflect the addresses and interfaces that apply at that point.

Port Mapping

Port mapping is a the general mechanism for changing IP addresses and TCP/UDP ports for packets that pass through the FireBrick, and where necessary changing them back as they reply. This is used for address translation as well.

The port mapping table is checked for all new sessions, after the packet has been routed and filtered. Port mapping applies to both stealth and non stealth traffic, and causes the traffic to become non-stealth.

The port mapping rules check :-

1. Source interface
2. Target interface
3. Protocol
4. Source IP address
5. Target IP address
6. Target ports (TCP/UDP) or ID (ICMP)

If a match is found then the port mapping rule sets :-

1. New target interface (may be ANY to allow normal routing rules to apply)
2. New target IP address (optional)
3. New target port (optional)
4. New source IP address (optional)
5. New source port may be implicitly set if source IP is changed

Changing target IP and port

If a range of target IP addresses was specified, and a new target IP addresses is specified, then this is mapped as a block change. The first matching IP maps to the specified target. The next one maps to the specified target plus 1, etc.

The same is true for target port ranges and new target ports.

If the new target port/IP is blank, then they are not changed.

If the matching target IP range is blank, then packets to the FireBrick are matched (note, you must give the mapping rule a name).

The change to target IP and/or port does not change the source address. Replies will have to come via the FireBrick for the change to be un-done correctly, otherwise if the replies can reach the original source without going via the FireBrick, then they will not match and will not work correctly. For this reason, it is often necessary to change the source address as well (to the FireBrick or an address routed via the FireBrick) so as to ensure replies will be un-mapped.

Changing source IP

The source IP can be changed. Just the specified IP is used, even if the matching source IPs are a range. If the new source IP is blank, then the source IP is not changed.

If the source IP is set, then the source port number (for TCP/UDP) or ID (for ICMP) is also changed.

These changes are reversed when the reply returns via the FireBrick.

ICMP errors

ICMP error messages have their content checked for existing sessions, and port mapping is also reversed, not only in the header of the ICMP error, but also the contents showing the original packet. All checksums are adjusted accordingly. This only applies for TCP, UDP, and ICMP.

Forced routing

If you wish to force routing of packets that would normally be handled only as stealth, you can create a port map with no change or source or target IP/ports. This forces the packet to be routed. Note that this will only work if the packets reach the FireBrick – in a stealth mode this would mean the original target IP would have to have responded to an ARP or be via a gateway which responded to an ARP allowing the packet to be sent via the FireBrick.

Another option would be to use proxy ARPs to force traffic via the FireBrick as routed traffic.

Special port maps

There are two special port maps if no others match. These are for DNS relaying and syslog relaying – i.e. traffic sent to the FireBrick specifically is relayed to a DNS or syslog server if defined in the setup. DNS is UDP/TCP port 53, and syslog is UDP port 514. These are only mapped if the server is defined in the FireBrick. The source port is changed to the FireBrick (NAT).

Fragments

It should be noted, that whilst port mapping can re-direct packets, and in some cases session tracking can work out a fragment is part of a redirected session, normally fragments cannot be handled by port mapping and so may be dropped or simply routed as normal if they pass filtering rules.

Other protocols

Protocols other than UDP, TCP and ICMP are session tracked and can be IP mapped using port mapping or NAT. However, as these protocols do not have a port change, there is no way to track different sessions between two IP addresses. The session is assumed for each unique pair of IPs and protocol. As such IP mapping cannot track multiple sessions causing replies to be misdirected.

Address Translation

Network Address Translation (NAT) is simply a special case of port mapping.

It is normally used where there are a set of private addresses on one side of the FireBrick and a single address on the other, and access from the private side is to be mapped via the single public address.

NAT is normally set by flagging NAT on a subnet. This means that all traffic from IP addresses on that subnet is flagged to be translated. In addition, routing entries can be used to mark specific traffic as NAT or non NAT and this takes priority over the subnet setting.

This only applies to routed traffic.

Once traffic has been marked as NAT, then routing rules are applied as normal. The source address is then mapped to the FireBrick's address on the destination interface and the source port (TCP/UDP) or ID (ICMP) is changed.

Reply packets to the FireBrick reverse these changes as per normal port mapping.

Where the traffic goes via a tunnel, there is no source address, and so the NAT is actually applied at the far end of the tunnel when the packet emerges to go via an ethernet interface. This can mean a duplicate IP in a traceroute of packets via a FireBrick tunnel.

Note, that whilst NAT will map IP addresses for protocols other than ICMP, TCP, and UDP, there is no way to track multiple sessions as the FireBrick cannot allocate a port or ID. As such NAT for such protocols can only be relied on where there is only one session at a time. If multiple sessions, then replies may go to the wrong one depending on the last session that was active.

Traffic Shaping

Traffic shaping is the system by which traffic is constrained to a specific speed. Its use applies to rationing internet access for certain users, as well as reducing the load certain types of traffic can create (e.g. email can be given a slow traffic lane to allow fast responses for web traffic).

The basic principle of traffic shaping is based on the fact that the outgoing traffic from the FireBrick is scheduled. Each packet has a time stamp stating when it is to be sent, and all traffic is (normally) sent in order, and not before its time. This method is used to deliberately slow responses from reject and bounce filters, as well as for speed lanes.

The objective is to emulate a router connected to a slow link.

One key advantage is that by limiting the speed of traffic leaving the FireBrick to a real router, you can ensure that the queuing of traffic is always in the FireBrick where you have more control (such as queue jumping outgoing TCP ACK messages, or making some traffic slower than others).

Speed Lanes

The FireBrick maintains a number of speed lanes. Each is given a name, and a speed to the WAN and a speed to the LAN. Speed limiting only applies to traffic which is to the LAN or the WAN (i.e. traffic that is going to a tunnel is not shaped in that direction but the final tunnelled data may be).

Internally, the FireBrick keeps track of how long it would have taken for the packets to be sent at the required speed. This means that it knows when the next packet could start sending based on what has already been sent. If that time is passed when a packet is to be sent, it is caught up with now.

When sending a packet, its length is considered, and that transmission time added to time for the next packet to be sent.

This ensures packets can only actually leave at the designated rate, and no faster. Note that the speed lane stats collect data as packets go in to the queue and not when they are actually sent, and so can show a higher rate.

Bandwidth sharing

A speed limit has two additional options – give and take. Each direction of each speed lane may give unused bandwidth away, and each lane may be set to take unused bandwidth. This is calculated every second, adding up all of the given away bandwidth and allocating to all lanes which will take such bandwidth. This does mean that a lane that gives bandwidth may receive it back if it takes bandwidth – but this means that the lanes maintain a relative priority. In addition, a cap can be set on the bandwidth a lane can take.

When bonus bandwidth is allocated to a lane, increasing its allocation, this is added gradually over several seconds. When decreasing the bonus, this happens immediately.

Shaping rules

A list of rules, like the filtering rules, are used to decide what speed lane each session is placed in. This is based on the initial packet that creates the session and applies independently of any filtering rules.

The first matching rule is applied. All traffic in a session is then sent via that speed lane.

The rules include :-

1. Source interface
2. Target interface
3. Source IP
4. Target IP
5. Source Ports (for UDP/TCP)
6. Target Ports (for UDP/TCP)

In addition, the shaping rule has a *both ways* option which checks for traffic in the other direction. Note that this checks for traffic from the target IP and to the source IP, but still checks that it is from the source port and to the target port. This is because it is more common to need to trap, say, all SMTP traffic either way between two points that it is to try and trap traffic to SMTP or from SMTP, the later being meaningless in most contexts. If you require the former, create two separate rules.

Buffer overflow

The outgoing queues are limited in size, and also a packet will not be sent if it is too far in the future. This limits the traffic that can be carried.

Also, new packets with earlier times to send can also push out old traffic if the queue is full. This can mean packets are silently dropped (and this is not reflected on the statistics or the speed lane control).

Changing rules

Unlike filtering and routing rules, it is desirable to ensure that traffic shaping rules can be changed on the fly, even for established sessions. This is particularly important where profiles (e.g. based on time of day) are used to change shaping rules. It would not be helpful to restrict web traffic after 9am, if a large download started at 8:59am could continue unrestricted. As such the shaping rules are constantly re-evaluated as a background task, updating the sessions to new speed lanes if appropriate.

Also, speed lane speeds can be changed at any time with immediate effect. However it should be noted that TCP may take several seconds to react to large changes.

Fast ACKs

The speed lane settings include a FastACK check box. This turns on a queue jumping feature which causes TCP packets with no payload (typically ACKs) to queue jump the normal speed lane. The time taken for the packet is still allowed when considering the timing of the next packet.

This feature can be used when limiting outgoing traffic on an ADSL line, so that the incoming traffic is not slowed by delayed ACK packets going through the outgoing speed lane.

Master speed lane

The first speed lane is the master speed lane and is applied to traffic even after it has gone through a specific speed lane. This allows the overall traffic rates to be controlled, and is invaluable if you are trying to ensure that an external router never fills its buffers. It is useful when used in conjunction

with the FastACK feature allowing the firebrick to control the order of some traffic. Note that setting FastACK on the master means it applies to all lanes. The Fast option on speed lanes allows traffic on that lane to not have the master speed lane applied. The traffic still affects the overall speed of the master speed lane, but Fast lane packets effectively queue jump the normal traffic.

Profiles

Most settings in the FireBrick are subject to a profile, the default being 24/7. A profile simply defines if the setting is active or not. When not active, the setting is completely disregarded. As most settings are checked in order, this means the setting will not match, and checking will continue to another match later.

There are 3 pre-defined profiles in addition to the ones that can be set manually – these include 24/7 (always on), 9–5 M–F (9am to 5pm Monday–Friday), 2am Sun (2am to 3am on Sunday). In addition, all profiles may also be selected for their "off" state, e.g. "Not 24/7" or "Not 9–5 M–F".

At any time a profile is active or inactive. This can be controlled by a simple master switch on the profile, or can be based on specific times of day and day of week, or based on the response of a particular host (responding to pings).

Master switch

Each profile has options for permanently enabled and permanently disabled. These settings affect the filter state regardless of the other options selected (such as ping IP, time settings, etc).

These profiles are shown on the *quick setup* page with a cheque box allowing them to be quickly changed.

Time controlled

The time profile is only available if the clock is set. When the clock is not set, the profile remains in its previous state. Time controls are set for each hour or each day of the week, and change on the hour.

Time setting profile

The time setting function is controlled using a time profile, resulting in a chicken & egg problem. To resolve this, the time setting function disregards the profile if a time based profile is used, and the time is not yet set. This means time setting is tried periodically for several minutes at a time until it is set.

Ping scanning

The ping scanning option is a profile which is active while a specified host is repoding to a ping.

Pings are only performed during the selected times, or anytime if the clock is not set.

- A ping is done for each ping scan or ping fail profile periodically
- Each profile is pinged in turn for its position in the list of profiles, i.e. one profile per second
- If a host was previous responding and did not respond this time, a ping is repeated every second for several more seconds until a reply.
- If there is no reply after these pings, the profile is changes to be the state for being unresponsive.

This means that a host will have to ignored several pings in a row at one second intervals to be considered unresponsive.

You can select not only the IP to ping, but the interface and if required the gateway router address – this is necessary if the ping scan itself affects routing, and you would otherwise continue pinging via a backup interface!. You can also select TTL to limit the distance the ping will go.

Changes

A change to a profile will have immediate effect on the rest of the operation of the FireBrick. However, bear in mind the time when filters and routers are tested – at the start of a session. As such the removal of a filter will not make a session invalid. Also, the removal of a subnet will stop the FireBrick answering ARPs for that subnet, but will not clear other devices caches of the ARP previously given.

A special case is traffic shaping rules which regularly check the validity of shaping rules and re-assign sessions on the fly. This may take several seconds to respond, but allows traffic shaping based on time profiles to be effective.

In addition a profile can have a *re-route* flag set which means a change in the profile causes all routed sessions to be re-evaluated when the profile changes. This is used for ISDN backups using the same IPs as it allows re-routing of existing sessions.

Type of ping

A standard ICMP echo request is used, with TOS 4 (high reliability), and no payload.

Pinging via specific interfaces

Because ping scanning is often used as the basis for changing routing – e.g. making use of a backup ISDN router when a leased line goes down, etc, it is often necessary to force the routing of the ping allowing the faulty route to continue to be monitored. As such the interface and gateway IP can optionally be pre-defined. Normal routing rules are followed within these constraints.

ALERT LED

A profile can also be set to affect the ALERT LED, setting it on solid red if the profile is active, or inactive as required. If any profile causes the LED to be on, then it is on regardless of blinking or flashing settings.

Tunnels

A tunnel is simply a means of packets being wrapped up and sent within another packet to another FireBrick, and then unwrapped and sent on its way. Tunnels are used for many purposes, but mainly to allow private network addresses to be carried between two internet connected FireBricks allowing a large virtual private network (VPN) to be established.

In filtering, etc, an interface of tunnel refers to any tunnel. Routing allows a specific tunnel to be selected as the destination for specific traffic.

Where a tunnel goes

A tunnel is defined with a name, and an IP address. It also has a reference which is the number of the tunnel at the other end. Both ends must define the tunnel, and it works both ways.

Packets sent to the tunnel by the routing rules are wrapped up and sent to the specified IP address.

What protocol tunnels use

Once wrapped up, the packet is sent using a UDP port 1 packet. This allows it to be set up on other firewalls and routers to allow this packet through to the FireBrick at the far end. UDP has been chosen because it will work with a wide variety of routers, filters, firewalls, and internet providers. Some internet providers may filter traffic to or from UDP port 1 – so such filters will need to be disabled.

IP Security

A FireBrick will only accept traffic on a tunnel if it is from the specified IP address (see dynamic addresses below). This provides some degree of security, and allows a convenient and fast tunnel to be set up between two known and trusted IP addresses.

Authentication

If specified, a secret is used on all packets. This slows the packets down slightly, but ensures that the packet is not changed, and is from the correct origin. The secret is used to construct an MD5 encryption check pattern (signature) on the packet. Any change or incorrect secret key and the packet will not be accepted.

Note however that contents of the packets going via a tunnel are not hidden.

Dynamic addresses

One end of a tunnel can have the IP left blank. This allows dynamic tunnels. In such cases it is recommended that secrets are used to ensure the authenticity of the packets received. Once a packet is received, replies are sent to the last IP and port from which they came.

This is specifically to allow for traffic originating from behind address translating (NAT) routers and internet services, and is why UDP was chosen as a common translatable protocol. This works best when data is always initiated from the dynamic end, such that replies will come back through the translation correctly. In such cases it is usually sensible to run keep-alives to ensure the NAT router does not drop the link (see below).

Fragments / MTU

Wrapping up your packet in another packet makes it bigger, and there is a limit to the size of packets that can be handled by ethernet (about 1500 bytes). This means that if a large packet is sent, then it will get too big when wrapped up.

The FireBrick handles this by splitting large packets in to small pieces (about 500 bytes each) and sending these pieces to the other FireBrick where they are put back together and continue as a single packet. This is not the same as IP fragmentation. This system has the disadvantage that if one piece is dropped, then the whole packet is dropped, so it is not that efficient. It also has extra overhead as each packet has an authentication signature and IP and UDP headers.

You can control some of the tunnel settings (MTU and *don't segment*) per tunnel and separately for each end of a tunnel. The MTU is the maximum size of the outgoing tunnel packet. This is, by default, 576, but can be set as high as 1500. A packet size of 576 should never be broken in to smaller bits on the way. To make most efficient use of a link, you can set this higher to reduce the chance of packets being broken up. The two ends do not need to have the same setting. It can also be used where the network in between has optimal packet sizes – for example ADSL carries large 1500 byte packets in two parts and puts them back together itself, so setting an MTU lower (say 1450) will ensure tunnel packets are not slowed down further in transit.

If your computers can handle "path MTU discovery", then you can also select *don't segment*. This causes large packets to be discarded and an error message to be sent back to the sending computer telling it to send things in smaller chunks. This does not work on all computers, but where it does, it will ensure that packets are not broken up unnecessarily. This does work and has been tested with Linux 2.4 kernels. If you use this feature, then you will need to set an MTU of more than 576, as the MTU reported in the ICMP error will be the size before the packet is wrapped up and so less than 576. Most computers will not accept an MTU of less than 576 and will still use 576 as the maximum packet size. When using *don't segment* set the MTU to at least 606 if you are not using a secret, or at least 622 if you are not.

Keep Alives

A tunnel has a state of being up or down. The tunnel is up if it has received any valid tunnel packets from the far end in the last 5 seconds. You can mark a tunnel to send keep-alive packets. These ensure that something (data or a keep-alive) is sent at least every second, and so the tunnel stays up. This is particularly useful for ensuring any router or link in between stays in operation. If keep-alives are being sent then keep-alives are automatically sent back in the same way. Where keep-alives are sent, one is sent at least every 10 seconds. The keep-alives also advise if the far end can see data (if it is up), and this allows you catch the situation where both ends are sending data but one end is not receiving it. You can also mark a tunnel as expecting data regularly, such that if it stops receiving it will log the tunnel as having gone down and also cancel any dynamic IP address/port in use.

A typical situation would have a dynamic linked unit sending keep alives and the other end expecting them.

Technical Details

The tunnelling protocol uses UDP port 1 packets, with one of two packet formats (single and multi-segment). In each format the first byte in the UDP payload is of the form SFPPVVV where S indicates the packet is signed, F indicates the last segment, PP indicates the part (0, 1 or 2) and

VVVV is the version (2). The second byte is a tunnel reference – which tunnel at the far end to use, and starts from 1.

For signed packets, a 16 byte MD5 signature is at the end of the packet. This is a signature of the whole UDP payload up to this signature, followed by the secret. The signature covers the UDP payload only so does not include IP or port numbers as they could change in transit via NAT, etc.

For packets that will fit in the sending MTU, the packet is sent in one go. In this case the tunnel reference is followed by the data in the packet, and then 30 bytes (before any MD5 signature) of header data. To reassemble the packet, simply move 30 bytes from the end to the start of the packet. This is a light weight tunnelling system which does not involve moving much data in the packet (only 30 bytes). These packets have F=1 (final) and PP=0 (first part).

For packets that will not fit, then the next two bytes after the tunnel reference are a cycling packet reference. Then there are up to 512 bytes of data. The packet may be in 2 parts (F=0/PP=0 and F=1/PP=1) or 3 parts (F=0/PP=0, F=0/PP=1, and F=1/PP=2) all with the same packet reference. All but the final packet are exactly 512 bytes. This is not as efficient because data has to be moved in the packet, and the 2 or 3 parts have to be reassembled. All segments in a packet are expected to arrive within 2 seconds.

For keep alive packets, F=0/PP=3 and one byte follows the tunnel reference with flags XXXXXSU1. U means the tunnel is up, and S means that keep alive sending is enabled in the config and so replies should be sent.

Debug level logging on the FireBrick will report if there are any unexpected tunnel packets received, etc.

DHCP

Dynamic Host Configuration Protocol (DHCP) is used to provide basic network parameters to devices on a network in an automatic way. DHCP servers can be found built in to simple routers as well as available for complex network servers. DHCP can be used to provide anything from the basic IP address and netmask through to any of hundreds of different parameters needed by a machine.

DHCP Client

The FireBrick can be a DHCP client. This means its basic network parameters can be set as a result of it requesting information from a DHCP server on the network. This is done by selecting DHCP client in any subnet. There is no point in selecting DHCP client on more than one LAN subnet and more than one WAN subnet as once the details are obtained they change all subnets marked as DHCP on that interface.

Once an address is allocated, the IP and netmask are set in the subnet configuration, and these can be viewed using the web interface.

In addition, the following parameters are also set :-

- Router (gateway) IP address – if the gateway interface is the one being assigned
- Time server IP address (the first time server, the second is not changed)
- Name server IP address
- Syslog server IP address
- Domain name

The subnet allows each of these settings to be ignored if required.

The provided renewal and expiry are followed, but are limited to a maximum. If renewal fails, then at expiry the IP and netmask are removed from the DHCP client subnets, and requests are retried. Renewals are normally to the server address, but can be forced to be broadcasts as an option on the subnet (some cable modems need this). If power cycled, all allocations are discarded and new requests sent.

For reference, the FireBrick requests a CLASS of "FireBrick" for its address – this can be useful for configuration of some DHCP servers. The FireBrick sends its own configured name as the client name on requests. It obviously works with another FireBrick as its server.

Checking

When offered an address, the FireBrick will normally do an ARP check on the address to confirm it is available. If it is already known at OFFER or ACK stage then the FireBrick will decline the address and start again. This feature can be disabled, as some cable modems will proxy ARP all addresses, even the ones being allocated !

Routes

Individual routing entries can be marked DHCP. When an address is allocated, these routing entries are updated automatically. This feature allows explicit routes to gateways and subnet, etc, to be used whilst having a DHCP allocated gateways and subnets.

DHCP Server

As a DHCP server, the FireBrick has a number of innovative features. It is not configurable to provide every possible DHCP parameter to clients, but can provide the basic parameters needed as well as some unique features useful on a network.

Basic operation

Any subnet can have a range of addresses specified. Only addresses within the subnet, avoiding network, broadcast and the FireBricks own address, are considered. Multiple subnets on the same interface allow different blocks to be allocated – i.e. the same subnet details can be repeated with different blocks of DHCP server allocations so as to make multiple blocks in the same subnet range. There is no harm if the DHCP ranges overlap as an address is only considered once.

The FireBrick will allocate IP, netmask, Router (itself), name server (itself), syslog server (itself), time server (itself) and domain name. However, the subnet can be configured not to provide router, name server, syslog server, domain name, or time server if required. The FireBrick provides itself for most of these addresses as it acts as a relay for them to the configured addresses (or as a server for time). This allows these to be set and changed manually or as a DHCP client on another interface without having to change each allocated machine.

Protection

Before OFFER and ACK of an address the FireBrick checks by ARP if the address is free. If not, then the REQUEST is refused, and the client must try again. Any addresses found in use are marked as such with a maximum lease time.

Reservation

If the FireBrick offers an address, it is marked as reserved (i.e. just expired) for that client. If the client does not request the address, then the slot is reserved, and may only be reused if addresses run out. This helps stop two overlapping requests from different clients trying to get the same address.

If the client declines an address then the address is blacklisted (marked as just expired but with no MAC). This means it will only be considered if no addresses are left. If the same client then asks for an address again it will not get the address it just declined (unless it is the only address left). This is important as some clients may have other ways to identify an unacceptable address and so would otherwise get stuck in a loop requesting an address and getting the same unacceptable address over and over again.

In both cases the allocation is marked as "just expired" and so the allocation can be immediately re-used if there are no addresses left. It simply makes them the last choice.

Persistence

Address allocations are persistent. Even when expired, the address allocated is not re-used for another machine unless there are no addresses available. This makes the allocations permanent, which is a very useful, albeit unusual, feature of a DHCP server.

Requested

If a machine requests an address, then it is honoured if that address could be allocated (within ranges, and rules for allocation and available for allocation). This means that if a set of machines were using a different DHCP server, and switch to the FireBrick – it will usually keep the same allocations. This can be very useful in maintaining persistence if replacing a FireBrick, and also for maintaining addresses if the FireBrick is operating as a backup server.

Logging/Status

The allocated addresses are available via the web interface, showing not only the IP and MAC address but also the expiry and the client name (if specified). This allows an immediate overview of all allocations. Individual allocations can be manually cleared. Allocations and releases are also logged.

Backup operation

The FireBrick can be marked as a backup server – only allocating addresses if another server has not yet done so. This is done by ignoring the first request (time 0) from any client.

Restricted addresses

Each subnet has a name, and it can be marked for restricted DHCP allocation. This restricts allocation to machines whose client name starts with the subnet name (case insensitive). This allows allocation of addresses to groups of machines based on the machine name.

If a machine sends a discover with no name, and it was previously allocated an address, the name is assumed to be the same – this covers some print spoolers that only send their name on the REQUEST and not the DISCOVER phase. i.e. it can be allocated by removing the restriction briefly, and then it will be able to renew or re-allocate after reinstating the restriction.

If the requesting machine has a name that matches a restricted subnet on that interface, then it will only be allocated an IP from one of the restricted subnets for which its name matches. If the requesting machine does not match a restricted subnet, then it can only be allocated from unrestricted subnets.

Sharing

The FireBrick server follows rules correctly for multiple DHCP servers, unlike some servers. This should mean it is not a problem if it is on a LAN with another server. Its protection mechanism even allows this where the allocation addresses overlap.

Running out

If no unexpired addresses are left or the DHCP allocation table is full, addresses are not offered (or if requested are NAKed). However, if all addresses have been used but some are expired, then the oldest address is re-allocated.

No clock

If the clock is not set then the lease time is the same, but is not known by the FireBrick. It effectively treats all allocations as permanent and will never re-use one even if it runs out. When the clock is set, all such allocations are set for the normal expiry from that point.

Mirroring

The DHCP server can be configured in mirror mode. This means it will be set when the FireBrick is allocated an address as a client on its other interface. In this mode, the IP address of the FireBrick on the mirroring interface is set to the router address from the other interface, and the DHCP server range is set to the single address allocated to the FireBrick on the other side.

This is normally configured such that WAN is DHCP client, and LAN is DHCP mirror and DHCP restrict and a blank WAN to FireBrick portmap to LAN. The client gets an address on the WAN, and sets the gateway, etc. The LAN side is set as the router and allocates a PC the FireBrick's WAN address. The PC then uses the FireBrick as a gateway which forwards to the real gateway. The FireBrick receives traffic for the PC and via a general portmap relays to the PC (subject to filtering rules).

Normally the PC's name is the name of the LAN mirroring subnet, allowing only that PC to get that address. Another LAN subnet is then set up with private addresses and DHCP server range and NAT. This allows other machines on the LAN to get addresses and work on the internet using NAT.

The main PC therefore follows the allocation from, say, a cable modem, allowing full incoming and outgoing non NAT traffic to operate. The continual renewal of addresses from the FireBrick even when the PC is off means the address does not change (with some cable modems) and so can be treated as a fixed address.

In case the address is changed from time to time, the lease and renewal on the mirror interface are issued to renew/expire 10 seconds after the DHCP client side, allowing a quick change of IP on the internal machine if the FireBricks IP is changed by DHCP.

Factory reset modes

The factory reset procedure allows the FireBrick to start up with a DHCP server on LAN and/or DHCP client on WAN enabled. See connectors section.

Time

The FireBrick does not have an internal clock, but tracks time while powered up. To keep time it uses an internet time server.

Any major changes are logged (i.e. more than a 1 second correction). Two time server addresses are defined, with the second being used if the first does not respond.

Setting the clock

Time requests are sent using UDP port 37 (timed), allowing the clock to be set to the nearest second. The time is set approximately every hour, and tried for several minutes. If no reply, the internal time is clocked anyway, but may drift by a few seconds a day if no time server responds. On power up the time is not set, and some functions do not operate or operate differently if the time is not set.

Note, some versions of linux (e.g. Red Hat 6) have a broken UDP port 37 time server giving a time that is about 8 years fast. The two time servers pre-programmed are time-a.nist.gov and time-b.nist.gov, which are two US public time servers.

In order to set the clock, the FireBrick must be able to route packets to the internet. A simple WAN subnet and gateway is normally all that is needed, but if operating in a full stealth mode, then a WAN stealth address will be needed as well as a gateway. Once these are set the clock is normally picked up automatically. It can however be set by selecting the Set button on the time setup page.

Note that if the clock is set, it is likely to log you off if logged in as you will suddenly have been logged in for many years !

Every hour

The clock is set at least once per hour, but not on the hour. This is deliberate, as public time servers would be overloaded if everyone set the time on the hour. You can also restrict the time setting using a profile, so that it is only set at certain times – e.g. once a day, once a week, only during the day, etc. This profile only applies once the clock is set!

Time server

The FireBrick is a time server for UDP port 37 and TCP port 37 (timed) only, and normally provides time to within 2 second. No response is sent if the clock is not set. (2 seconds, as the reply is only to the second, and the time was only set to the second).

Time zones

The time is kept internally in UTC, but a time zone from -11 to +12 hours can be configured. In addition, a one hour offset can also be applied for "summer time". This is automatically adjusted in the early hours of two Sundays in the year. Automatic adjustment can be disabled by setting either of the dates for changes to "Manual". You can configure the summer time to start after the winter time in the year for southern hemisphere operation if required.

All displays and logs and time profiles are based on local time.

Logging

Logging is generated as a result of a filter being activated (at the start of a session), the end of a long session and at various key events in the operation of the FireBrick. In all cases the method of logging is based on a number of options. For filters, these are specified in the filter. For other events these options are specified in the log/filter options in the set up.

The FireBrick has a 128KB internal log buffer holding recent logs entries. This buffer is also used for all upload and download, and so is cleared whenever new software or configs are loaded or the config saved.

Log options

The following actions can be selected in any combination.

Blink

One of the logging options is to *blink* the LED on the front of the FireBrick. This will blink once, but the rate is limited, so continuous blinking log events will result in a rapid flashing of the LED.

Flash

Another log option is to start the LED flashing. This will then flash at a slow rate until the event is cleared. The time of the event that first starts the LED flashing is recorded so that this is visible even if the log has filled and lost the event.

Memory log

The FireBricks internal log memory is used to record log events, with a time stamp (if the time is set). When displayed, it is displayed in local time, so if viewing a log covering the change from winter to summer time, all entries will show with summer time even those before the change. The heading shows the UTC offset that applies to the log.

Syslog

An event can be marked to generate a syslog entry. The local0–7 syslog options can be selected and the syslog server. This is useful for making a permanent record of some events.

Email

An event marked for email will also cause the event to be logged in the memory log, but not displayed until the memory log option was also selected. As such emailed events take space in the log.

The email system then monitors the log continuously for an emailable event. When this happens a pre-email delay applies, and then the email is sent. All entries up to the current time that are emailable are sent in the email. If sent correctly, then all of these entries are marked as not emailable to avoid duplication and the log monitoring continues after a post-email delay. If the email fails, then the monitoring stays at the same point to try again.

Email can be sent to a specified IP address using a specified sender and recipient using normal SMTP. Sending or failing to send an email is also logged, but this has the email option removed to

avoid self generating email logs.

End-log

A filter can have a flag causing the "large session" logging to be applied regardless of the length of the session.

Filter logs

Log entries simply consist of a text line, which usually includes the relevant IP address, etc. Filter logs contain more detail and help allow filters to be constructed to allow or disallow specific types of traffic.

Colour coding

Most log entries are shown in black. Filter log entries are shown in Black, Green, Red, or Blue depending on whether the filter was Drop, Allow, Reject, or Bounce. This helps identify different types of filter log entries at a glance. Colour coding only applies to the live web display, and not to syslogs, etc.

Log format

The contents of the log entry depend on the protocol as follows.

Log formats

Fragments	<i>filter-name: Frag MAC proto-name(proto-num)/id@offset interface/IP-interface/IPaction</i>
TCP	<i>filter-name: MAC TCP(6) interface/IP/ port-interface/IP/portflag saction</i>
UDP	<i>filter-name: MAC UDP(17) interface/ IP/port-interface/IP/port action</i>
ICMP	<i>filter-name: MAC ICMP(1) interface/IP/ type-name(type:code)-interface/IP/id action</i>
Other	<i>filter-name: MAC proto-name(proto-num) interface/IP-interface/IP action</i>
Big sessions	<i>End filter-name: proto-name(proto-num) interface/IP/port-interface/ IP/port forward/reverse</i>

The meaning of these various fields are as follows. In all cases the source then the target interface/IP are shown.

filter-name	The name of the filter, or Default for the default filter rule
MAC	The source MAC address of the packet (0 if not known)
proto-name	The name of the protocol, e.g. TCP, GRE, etc. There are over 100 named IP protocols
proto-num	The protocol number, e.g. 6 for TCP, 17 for UDP, 1 for ICMP
interface	The name of the interface, e.g. WAN, LAN, Serial, etc. The FireBrick's name is used for the FireBrick
IP	The IP address, e.g. 1.2.3.4 or 001.002.003.004 if padded formatting has been configured
port	The port number for UDP or TCP, e.g. 1234 or 1,234 if comma formatting configured

id@offset	The IP packet ID and fragment offset
type-name	The ICMP packet type, e.g. ECHO
type:code	The ICMP type number and code number.
id	The ICMP id word
flags	TCP flags for S=SYN, A=ACK, R=RST, F=FIN which identify the type of TCP packet
action	The action, Drop, Allow, Reject, Bounce, matching the colour coding
forward/reverse	Total bytes sent in each direction on a session (forward is direction of first packet). If this exceeds 4.2 billion (32 bit wrap around) then it will be incorrect.

DNS

Domain Name Service (DNS) is used to look up IP addresses from names (as well as other types of data).

DNS Relay

The FireBrick acts as a DNS relay. A DNS server is specified in the config, or as a the result of DHCP. Any requests to the FireBrick to UDP/TCP port 53 are simply mapped to the currently specified DNS server.

DNS Lookup

Most of the input fields for an IP address in the FireBrick can has a domain name specified instead. The FireBrick can look up a domain name where the result is a simple A record and replace with the IP address found. This requires a DNS server and routing (gateway and subnet or stealth) to be specified so that the FireBrick can send/receive the DNS request/reply. This is provide for convenience only, and will not work in all cases (e.g. CNAMEs).

Statistics

The FireBrick records a number of statistics as part of its normal operation. The basic statistics recorded are :-

Current	The current data rate, i.e. the amount transferred in the last second – displayed in KB/s to one decimal place
Last 5 mins	The data rate per second used in the last whole 5 minutes
Today	The data transferred so far today, recorded in MB
Yesterday	The data transferred yesterday, recorded in MB
This month	The data transferred this month so far, recorded in MB
Last month	The data recorded last month, recorded in MB

The statistics for today actually include the number of bytes as well, and this is carried forward at the end of the day. As such a transfer of 512KB per day will log as 1MB one day then 0MB the next and so on. If any bytes are held in today, it is displayed as 0MB rather than blank. Erasing a filter or speed lane will clear this residual byte count.

Statistics are only shown when the clock is set, and are rolled to the next day/month if the day/month has changed. i.e. if the FireBrick is off for 2 days, the previous day will show as yesterday. If the clock was not set for some time, when it is set all of the recorded stats will appear as the last month and day even though they may span many days or months.

Statistics are recorded for :-

1. Filters. The filter applicable to a session means all traffic on that session is recorded against that filter. Total traffic both ways is recorded.
2. Speed lanes. Traffic is recorded in each direction (to WAN and to LAN) separately.
3. Interface – speed information is provided for WAN/LAN send and receive separately on the status page, and more details on the counters.

Reporting

A log message (e.g. syslog) can be generated automatically every 5 minutes. The general log/filter setup includes this option. The log is generated for each interface, speed lane and filter. On the 5 minute boundary the stats are collected and then reported at a rate of 1 report per second for all items where the stats are non zero.

In all cases the number reported is the total number of bytes recorded in the previous period, from which average rates per second can be calculated.

Interface	Stats eth <i>i</i> : tx rx (<i>name</i>)	Reported for i=0 name=WAN and i=1 name=LAN
Lane	Stats lane <i>l</i> : to-wan to-lan (<i>name</i>)	Reported for all lanes
Filter	Stats filter <i>f</i> : <i>n</i> (<i>name</i>)	Reported for all filters

Note that the first report after power on may cover less than 5 minutes.

Ethernet counters

A number of counters are provided for the ethernet interface as follows :-

Metric	Description	Example
Tx packets	Packets sent	
Tx collide	Packets which collided	Network is busy
Tx late collide	Packets which hit late collisions	Fault on network
Tx drop	Packets dropped due to 16 collisions	Network is too busy
Tx underrun	Packets where the FireBrick could not keep up sending	Should not happen, packet automatically resent
Tx lost carrier	Packets which failed due to a loss of carrier during sending	Network cable unplugged during sending
Tx jabber	Ethernet controller is sending for too long	Should not happen, hardware fault
Tx now	The number of bytes sent in last second	
Tx 5min	The bytes per second average for the last whole 5 minutes	
Tx today	MB sent so far today	
Tx yesterday	MB sent yesterday	
Tx this month	MB sent so far this month	
Tx last month	MB sent last month	
Rx packets	Packets received	
Rx miss	Missed packets	E.g. data too fast for ethernet controller to keep up
Rx crc	Packets with errors	Noise on network, unplugging cables, fault
Rx runt	Packets that were too short	Noise on network, unplugging cables, fault
Rx extra	Packets that were too long	Should not happen, network fault
Rx dribble	Packets with non whole byte content	Should not happen, may be network fault
Rx drop	Packets dropped due to lack of buffers	Too much data arriving too fast
Rx ignore	Packets on local network	Normal network traffic for other devices
Rx 802.3 ethernet	Ethernet packets sent in 802.3 format	Some devices send in this format
Rx 802.3 unknown	Other 802.3 packets	Some devices and protocols use such packets
Rx now	The number of bytes received in last second	
Rx 5min	The bytes per second average for the last whole 5 minutes	
Rx today	MB received so far today	
Rx yesterday	MB received yesterday	
Rx this month	MB received so far this month	
Rx last month	MB received last month	

Security

This section covers the user level security of the configuration data in the FireBrick, and is not a discussion of the general fire wall security aspects.

Users

There are a number of users definable in the FireBrick. Each has a name and password. The password is encrypted internally, and so can never be viewed. The user interface shows a single * if the password is set – as such a password of a single * cannot be used (its a bit short to be sensible anyway).

Each user has a number of security controls :-

1. Interface – this restricts the interface on which the user can log in.
2. Profile – this restricts when the user can log in
3. Timeout – this causes an automatic logout after a number of minutes
4. View rights – controls what parts of the config can be viewed
5. Edit rights – controls what parts of the config can be changed

How it works

The login operates using a directory and IP check. When you log in you are moved to a directory which is simply a number. This is a session ID, but as a directory will work with any browser without the need for cookies. The session is tagged against the logged in user along with the source IP address. If logging in from a different session or IP then the first is discarded (i.e. you can be logged in from one place at a time only). Most of this is *hidden* by a frame set.

Nobody user

The nobody user is a special user – it is the user that applies before you have logged in. As such it has no password. The access controls do apply to the nobody user, and if you cannot access the nobody user then the web page simply shows a 403 error "Goodbye".

As such, basic access to the nobody user is required in order to log in. This means if any user is to have WAN login access, so must the nobody user.

By default the nobody user has view and edit level 1, which allows general control of the firebrick, and particularly it allows the password to be set on the admin user which has level 1–8 view and edit rights. As such it is strongly recommended that the admin user is set up with a password, and the view/edit rights of the nobody user removed. In this state, the initial access will simply allow login, and no other actions.

Logging

Login attempts are logged, subject to log options. The password used on a failed attempt is not logged.

Security levels

Security levels are 1 to 8, and most settings (filters, routes, etc) have a security setting. Users have rights to view and edit levels 1 to 8. The levels themselves have no importance – i.e. 1 is not *better*

than 8. To see something the user must have the view right for that level. To change something the user must have the edit right for that level.

There are also general controls which hide the top level menus. E.g. filters can be set to a security level generally, and this stops access to filters for users without that right. Note however that the individual filters may allow access. This means that to be sure of stopping access to filters you must not only set the general level for filters to hide the menu, but also ensure that the individual filters have an appropriate security level as well.

Pay particular attention to the rights to upload new software and configurations (see below).

Config save/load

A security level is specified for *upload/download* which controls loading and saving of the configuration as well as uploading new software. The configuration can be saved if the user has view access to the *upload/download* security level. The configuration and any new software can be uploaded if the user has edit permissions to the *upload/download* security level.

Note that saved configs do not contain passwords as they are encrypted, but the config file itself, whilst scrambled, could be decoded to show all other configuration parameters, including those that the user could not normally see. As such access to *upload/download* should be restricted.

Tips

This section is mainly aimed at IP consultants who install FireBricks. There are a few special functions which can be of use when installing and maintaining FireBricks.

These extra features are rarely used, and in some cases can have significant effect – as such, to avoid complicating the user interface and avoid accidental use, they require manual entry of a specific URL.

To access these features, you must log in using a fixed session. Do this by going to <http://my.firebrick.co.uk/1/> and then log in as normal. This creates a session (number 1) which is associated with your login. Once logged in you can then access some special URLs, e.g. <http://my.firebrick.co.uk/1/reboot> will reboot the FireBrick immediately.

Keyword	Function
reboot	Resets the firebrick – same as a power cycle. (requires upload permission)
factoryinit	Does a factory init (requires upload permission)
datainit	Resets all statistics and counters. (requires upload permission)
zapfilters	Erases all filters for which the user has edit permission
language	Reports the current language (e.g. EN for English)
manufacture	Reports the manufacture date
serial	Reports the serial number
version	Reports the software version

FAQ

This document includes answers to frequently asked questions, as well as general tips.

I can only see Setup and Users menus, shouldn't there be more?

Out of the box the FireBrick does not require any login to access the basic settings. In this state most of the menus are hidden.

You should set up an admin user, and log in as the admin user. This makes the FireBrick much more secure and allows you to see all of the menus.

To do this, go to users, select admin, and enter a password (in both boxes) and save. Then select login from the top left and enter the username admin and the password you chose. You should then see more icons and menus. You can then go to users, select nobody, and un-tick the view and edit rights for level 1 and save. In future, always log in. You can change the user name from admin to something else if you wish, and add others users.

I cannot access the FireBrick any more – HELP!

The FireBrick has a lot of security, and it is quite easy to configure yourself in to a hole – not allowing you access. There are default filters to stop you doing this by mistake, and it is wise to leave these in place and active until you are sure what you are doing. It is also wise to save the config regularly, so that if you have to factory reset the FireBrick, you can go back to the last working state and not have to start again. If you have managed to make such a configuration, or even simply forgotten your password, then the only option is a factory reset.

To factory reset your FireBrick, remove all connections from it, and then use a straight patch lead (as supplied) to connect the WAN (left hand) port to the far right hand LAN port. Then power up and wait a second for the red light to come on. Then remove the patch lead. The FireBrick will blink its lights then go show cycling lights – it is now factory reset.

I have set the user to WAN access, but it just says "Goodbye"

So you want access from outside ? You must also set the nobody user to WAN access, as this is required to show the log in screen at all.

I cannot set the clock!

The clock is set from the internet, but to do this the FireBrick must be able to talk to the internet. This is normally via the WAN port to the time servers configured in the FireBrick by default. Just because you can talk to the internet from a PC on your LAN via the FireBrick does not mean the FireBrick knows how to – this is the case in stealth mode.

To get packets to the internet, the FireBrick will need two key things – a gateway, and an IP address. The gateway is just the address of your router on the WAN side and is set in the setup menu. The IP address is a different matter – it can either be a public IP address set up using a subnet on the WAN side, or can be a stealth address. Either way it must be an address which will

find its way back to the FireBrick from the time servers on the internet.

If you do not want to give your FireBrick its own address, then it can *borrow* one from your LAN. In the setup/stealth menu set the WAN stealth address (previously blank) to the public address of a machine on your LAN which is normally switched on. The FireBrick can then borrow this address to set the clock. This should have no effect on the operation of that machine. Don't change the LAN stealth address (normally 217.169.0.1).

Normally after a change in config, the clock is set, but you can force it by selecting the Set button in the clock setting menu under setup. If the clock is set for the first time and you are logged in, you will probably find you are logged out.

I cannot get FTP to work

The way FTP works means that it normally tries to make a separate connection back to you when you try to transfer a file or view a directory. This connection is quite separate and is seen by the FireBrick as an unwanted incoming connection.

There are two possible problems with this – firstly that you will quite sensibly have filtering stopping such unwanted incoming connections. You can get around this, reducing your security, by allowing some traffic in. You should restrict the IP addresses if possible, e.g. if it is your web server you are FTPing to – allowing connections only from that FTP server. Also, only allow connections to ports 1024 to 65535. If you look at your logs you may find that the incoming connections only come from a specific port, such as port 20 or 21, and this can make the filter even more specific. The other solution is to use passive mode (see below).

The other problem is with NAT – i.e. you have a private address, and the FireBrick is converting this to a real address for you. The FTP control session tells the other end to connect back to a specific place and port, but if you are on a private address block, it will tell the other end to connect back to a non-existent address and it won't work. The only way around this is to use passive mode.

Passive mode simply means that instead of the other end connecting back to your FTP client when you transfer data, you connect to the FTP server again. This solves most problems, but not all FTP clients have a passive mode. It is recommended that you use a client that does have a passive mode (it may also be called a firewall mode).

You will however be unable to use passive mode where the ftp server at the far end also has a firewall, simply because it will not allow your extra connections to the FTP server. This is simply one of the downsides of having some security.

I have set port mapping to one of my other public addresses but it does not work

Typically, if you have a small block of public addresses, with the FireBrick on one of them, and you want to set port mapping of some of the other public addresses you have through to machines on your LAN. You set up the port map on the FireBrick, and ensure the filters are allowing traffic, but it still does not work and nothing appears in the log even...

This is an ARP issue. The internet router expects the other public addresses to be on the ethernet (WAN) and tries to ARP for them. This gets no reply, and so the router does not even try to send the packet (hence no log entry on the FireBrick).

To solve this you need to make the FireBrick ARP reply for these other addresses. This can be done in one of two ways.

1. Add additional WAN subnets quoting the IP addresses you want the FireBrick to answer on.
2. Add a route from WAN to LAN for a range of public addresses, marked "Proxy ARP"

Either way the packets will get to the FireBrick, and so should work. Check logs for any clues to missing filters that you may need.

I think filters are getting in the way

Basically, if you set up anything complex, such as port mapping, complicated routing, or tunnels, you can be caught out by filters. It is important to realise that filters are checked in order – so an early filter may block traffic which you allow in a later filter.

A good tip to eliminate filters to to move a filter to the top of the list that is Any->Any with everything blank, Allow, and Log. If what you are doing works in this situation, the problem was filters and you can check the log to see what is happening. Pay attention to the interfaces (WAN/LAN/Tunnel, etc) and IPs and ports of the sessions being allowed by the filter, and set up filters to allow the traffic you require.

When you have finished remove or suspend the Any->Any filter to ensure you are firewalled again.

Limits and timeouts

There are various specific timeouts and limits applicable to the FireBrick operation. These are not specified within the text of the technical reference manual, but collected together in this one appendix.

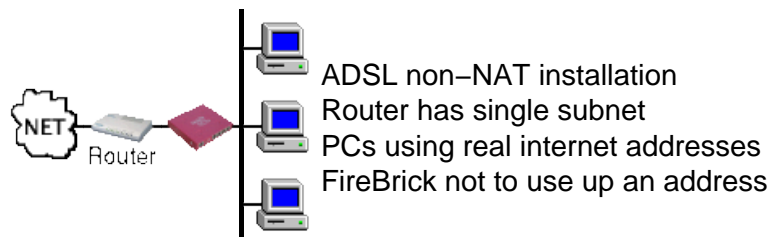
Ethernet	
Total MAC cache size	2048
Ethernet MTU	1500 bytes
ARP	
Cache size per interface	256
Time from initial request to first repeat	1.00 to 3.56 seconds
Repeat request time	2.56 seconds
Timeout for ARP request	10 seconds
Hold time before refresh	60 seconds
Max packets pending ARP reply	20
ARP max outstanding sessions	10
ARP session timeout	2 seconds
ARP session hold after reply	0.1 seconds
Session tracking	
Max sessions	4000
Default TCP initial timeout	10 seconds
Default TCP ongoing timeout	2 hours
TCP timeout after RST received either way	5 seconds
TCP timeout after ACK after FIN each way	5 seconds
TCP timeout after ICMP error	5 seconds
Default UDP initial timeout	60 seconds
Default UDP ongoing timeout	10 seconds
UDP timeout after ICMP error	10 seconds
Default ICMP initial timeout	10 seconds
Default ICMP ongoing timeout	2 seconds
ICMP timeout after ICMP error	10 seconds
Default other initial timeout	100 seconds
Default other ongoing timeout	100 minutes
Profiles	
Normal ping interval	10 seconds
Repeat ping interval on no response	0.869 seconds
Repeat ping attempts on no response	7
Max loss of host detect time	16 seconds
Min loss of host detect time	6 seconds
Average loss of host detect time	11 seconds
Max acquire of host detect time	10 seconds + ping response time
Min acquire of host detect time	0 seconds + ping response time
Average acquire host detect time	

FireBrick Technical Reference Manual

	5 seconds + ping response time
DHCP	
Max DHCP expiry	7 days
Default renewal	Half expiry
Default expiry	2 hours
Max allocations	256 per interface
Logged client name length	20 characters
Time	
Clock refresh time	55 to 56 minutes
Clock request interval	10 seconds
Clock request abandon	2 minutes

Examples

ADSL/Stealth



In this configuration the FireBrick operates in a full stealth mode, not using one of the addresses allocated by the ISP.

1. The FireBrick will operate out of the box with no extra configuration if required
2. PCs on the LAN must have the router address as their gateway address
3. Access the FireBrick config from a PC on the LAN using <http://my.firebrick.co.uk/>
4. Adjust filters as required

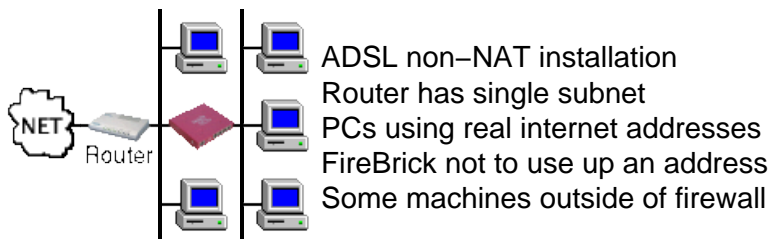
For clock setting, and any external communication from the FireBrick such as emailed logs :-

1. Pick one of the PC addresses for a PC that is normally on
2. Set this as the WAN stealth address in the setup menu
3. Set the router address as the gateway in the setup menu

This example equally applies to :-

1. Any installation with a router and a single subnet
2. BT net start lines
3. Existing network installations with a router

ADSL/Stealth with external machines



In this configuration the FireBrick operates in a full stealth mode, not using one of the addresses allocated by the ISP. Some of the PCs are on the LAN side and some are on the WAN side. This is usually done where the external machines are carefully configured to be secure, but if the external machines are compromised then this does not allow access to the internal machines.

The FireBrick provides no protection for the PCs on the outside.

1. The FireBrick will operate out of the box with no extra configuration if required
2. PCs on the LAN must have the router address as their gateway address
3. Access the FireBrick config from a PC on the LAN using <http://my.firebrick.co.uk/>
4. Adjust filters as required

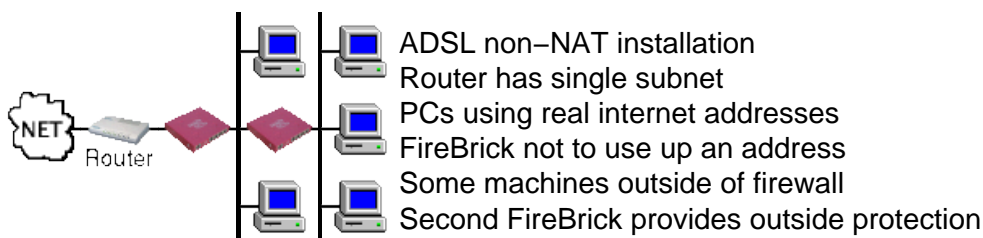
For clock setting, and any external communication from the FireBrick such as emailed logs :-

1. Pick one of the PC addresses for a PC that is normally on and on the LAN side
2. Set this as the WAN stealth address in the setup menu
3. Set the router address as the gateway in the setup menu

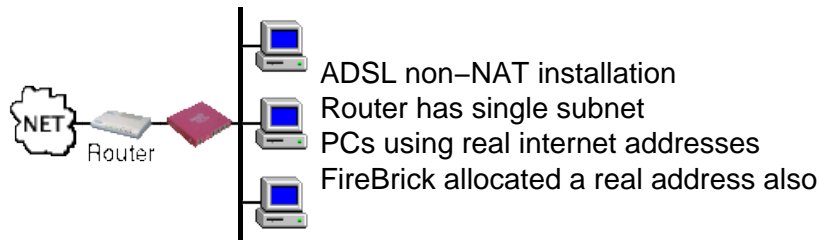
This example equally applies to :-

1. Any installation with a router and a single subnet
2. BT net start lines
3. Existing network installations with a router

In such cases, a second FireBrick is normally recommended. In this case, you may wish to change the LAN stealth address of the outer FireBrick to a different address, such as 217.169.0.2, so that it can be accessed from PCs on the inside without picking up the internal FireBrick by mistake.



ADSL/Stealth + FB address



In this configuration the FireBrick operates in stealth mode but has a real address. This is normally done to allow external access to the FireBrick configuration.

1. Pick an address for the FireBrick
2. Create a LAN subnet with that address and the appropriate subnet, marked stealth
3. Create a WAN subnet with that address and the appropriate subnet, marked stealth. Ensure this is after the LAN subnet
4. Set the gateway on the FireBrick to the router on the WAN
5. PCs can have the router or the FireBrick as their gateway
6. Always ensure all PCs, and the firebrick subnets have the subnet mask allocated by the ISP.
7. Adjust filters as required

For external access to FireBrick web management pages :-

1. Enable a filter allowing WAN to FireBrick for at least TCP port 80
2. Ensure the admin user has a password, and disable the view and edit rights for the nobody user
3. Set the required user to WAN access, and the nobody user to WAN access (to allow the login)

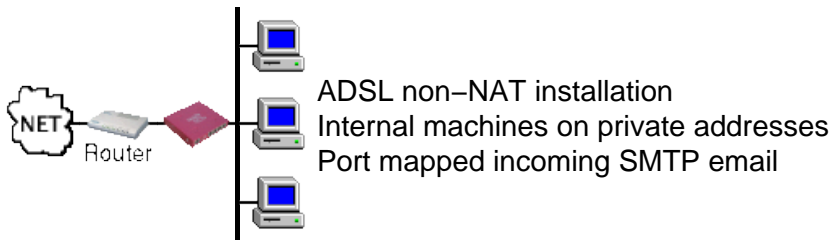
If DHCP allocation to PCs is required :-

1. Set the DNS server address in the FireBrick so the FireBrick can be used as a DNS relay
2. Pick a range of addresses for DHCP use, and set these on the FireBrick LAN subnet
3. Mark the LAN subnet as not stealth – this allows the DHCP server to work correctly
4. Add a route from LAN to WAN with target IP of the router and proxy ARP. This allows access to the router.
5. Ensure PCs are set to automatic IP and (for windows) DNS disabled.

This example equally applies to :-

1. Any installation with a router and a single subnet
2. BT net start lines
3. Existing network installations with a router

ADSL and private network behind FireBrick



In this configuration there is a routed non-NAT internet feed (e.g. ADSL). The PCs are to be on private addresses. In this example we will assume that the ADSL router has address 123.4.5.1 and the subnet is a block of 16 (/28 or 255.255.255.240).

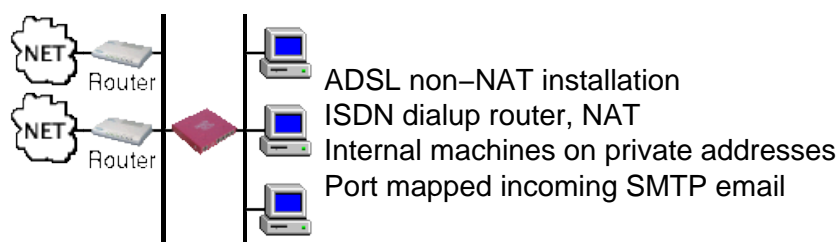
The FireBrick provides a NAT configuration to private addresses :-

1. Allocate a private network address for the internal machines, e.g. 10.0.0.0/24
2. Allocate the FireBrick a private address, e.g. 10.0.0.1 creating a LAN subnet for the FireBrick on this address and subnet 24 (255.255.255.0), set NAT
3. Optionally, include DHCP allocation range on the private addresses to allocate addresses to machines on the LAN
4. Allocate the FireBrick one of the public addresses, e.g. 123.4.5.2 and create the WAN subnet with this address, subnet 28 (255.255.255.240)
5. Set the gateway on the FireBrick to the router on the WAN (i.e. 123.4.5.1)
6. PCs are set with the FireBrick as their gateway (i.e. 10.0.0.1) and subnet 24 (255.255.255.0)
7. You may want to set the FireBrick with an ISP allocated DNS server address, and set the PCs to use the FireBrick for DNS (needed for DHCP use)
8. Adjust filters as required

This example equally applies to :-

1. Any installation with a router and a single subnet
2. e.g. BT net start lines

ADSL with ISDN fallback



In this configuration there is a routed non-NAT internet feed (e.g. ADSL) and also a backup ISDN dialup router. The dialup router is using a conventional dialup which provides NAT from a single internet address. In this example we will assume that the ADSL router has address 123.4.5.1 and the subnet is a block of 16 (/28 or 255.255.255.240).

The FireBrick provides a conventional NAT configuration :-

1. Allocate a private network address for the internal machines, e.g. 10.0.0.0/24
2. Allocate the FireBrick a private address, e.g. 10.0.0.1 creating a LAN subnet for the FireBrick on this address and subnet 24 (255.255.255.0), set NAT
3. Optionally, include DHCP allocation range on the private addresses to allocate addresses to machines on the LAN
4. Allocate the FireBrick one of the public addresses, e.g. 123.4.5.2 and create the WAN subnet with this address, subnet 28 (255.255.255.240)
5. Set the gateway on the FireBrick to the router on the WAN (i.e. 123.4.5.1)
6. PCs are set with the FireBrick as their gateway (i.e. 10.0.0.1) and subnet 24 (255.255.255.0)
7. You may want to set the FireBrick with an ISP allocated DNS server address, and set the PCs to use the FireBrick for DNS
8. Adjust filters as required

The ISDN router needs to be configured to allow access whenever it is used :-

1. Allocate a public address for the ISDN router, e.g. 123.4.5.3, and set with subnet 28 (255.255.255.240)
2. Set the default incoming address translation/NAT-mapping to the FireBrick 123.4.5.2 allowing incoming mail, etc.
3. Set up dial on demand internet connection with NAT

The FireBrick needs to monitor the ADSL link :-

1. Find the next hop address on the ADSL (see below)
2. Create a profile called ADSL, set for ping scanning on interface WAN with gateway 123.4.5.1 to the next hop address, set *Alert if inactive*
3. Ensure the profile is set for all day every day (click the right hand box for each day, marked "24")
4. Confirm by reloading the profile index page, after 1 minute, that the profile is active

The normal FireBrick routing will need to be replaced with explicit routing rules allowing for a change to ISDN when required :-

1. Move the *Subnets* route up, and add a new route below it to the ISDN router, target Any (blank), from Any, to LAN, gateway 123.4.5.3 (the ISDN router), select NAT, Profile No-ADSL

If you need specific port maps for incoming mail :-

1. Create a port map, from WAN, to FireBrick, addresses Any, port 25 (may left blank), map target to your mail server, e.g. 10.0.0.2

Incoming email :-

1. Incoming email for SMTP could be set with MX records to go to your FireBrick, e.g. 123.4.5.2
2. The FireBrick would need to allow WAN->Any port 25 TCP traffic in its filters and have the port map as specified
3. If you want email when in ISDN backup, then ensure you have a fixed IP ISDN dialup and set this address as the secondary MX record (via an A record).

Testing :-

1. Confirm by viewing the profile index that the ADSL profile is active (ALERT LED off)
2. Traceroute to confirm routing via ADSL
3. Remove connection to ADSL, and up to wait 1 minute for ALERT LED to come on
4. Confirm by viewing the profile index that the ADSL profile is not active
5. Traceroute to confirm, routing via ISDN
6. Reconnect ADSL and reconfirm that the filter becomes inactive and ALERT LED off with 1 minute

Emailing to tell you the backup has happened :-

1. Set the log/filter options so that Email is selected for ping-scanning
2. Fill in target email address (and optionally, source email address)
3. Enter mail server address, e.g. 10.0.0.2
4. Click "test" to confirm email can be delivered.
5. If test fails, check Status Log for error message and configure mail server accordingly
6. Adjust email delay/timeouts if required

Next hop :-

Monitoring the ADSL link requires that a specific address is checked regularly using a ping. The ping-scanning and ping-failure features of the FireBrick allow for this, and change a profile accordingly. One issue is what address to monitor.

Using traceroute to some address on the internet (your favourite web site for example), you will see the FireBrick, your ADSL router and a next hop. This is a good candidate for monitoring, and means if your ADSL line goes down, the you will switch to ISDN. However, if your ISP has problems (e.g. their upstream fails) and your ADSL line is actually OK, you may lose internet access and not fall back to ISDN.

Using a later address or an address on the internet would allow you to protect against failures within your ISP, and switch to ISDN. Going too far can be a problem, e.g. picking some web site. If you do this, you would find you switch to ISDN simply because the one site you were monitoring was down, even though the rest of the internet was fine.

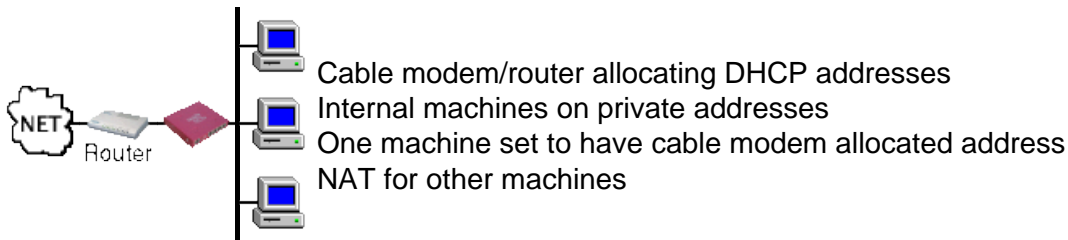
Your ISP may be able to suggest an address to be monitored like this, and this is the best one to use.

1. Check the address you pick answers a ping
2. Make sure nobody minds you monitoring the address – if it is the router next hop, then this is likely to be fine, but some address on the net may upset the owner of that machine. The pings are very light load, but that can be detected.
3. Bear in mind the address could go away. Again, the router next hop is unlikely to, but any other address could be removed or changed without warning. So check you are not using backup routing when you don't want to – we suggest the email alerts are used but keep an eye on the ISDN router just in case.

This example equally applies to :-

1. Any installation with a router and a single subnet
2. e.g. BT net start lines

Cable modem, with one machine having external address



In this configuration there is a cable modem allocating a single address by DHCP. This is normally intended for use with one PC (so check if terms and conditions allow for use of a network).

The FireBrick will obtain an address from the cable modem, and provide NAT to a private address block on the inside of the network. PCs on the inside are allocated addresses by DHCP.

One machine on the inside is to have a public address, so as to allow incoming email, web, etc. This address may change because the cable modem service allocated by DHCP, but with the FireBrick constantly renewing addresses, it is unlikely.

1. Create a WAN subnet, marked DHCP client
2. Create a LAN subnet marked DHCP mirror – give it a name such as "SERVER", and mark it DHCP Restrict
3. Create a LAN subnet on a private address range, e.g. 10.0.0.1 mask 24 (255.255.255.0) and set DHCP server addresses (e.g. 10.0.0.10 to 10.0.0.99), and mark as NAT
4. Create a portmap, WAN to FireBrick mapped to LAN with nothing else filled in
5. Ensure the server PC has a name, such as "SERVER" which is the same as the first LAN subnet
6. Adjust filters as required

You should find the WAN subnet gets an address, and the gateway and DNS server addresses are set up automatically.

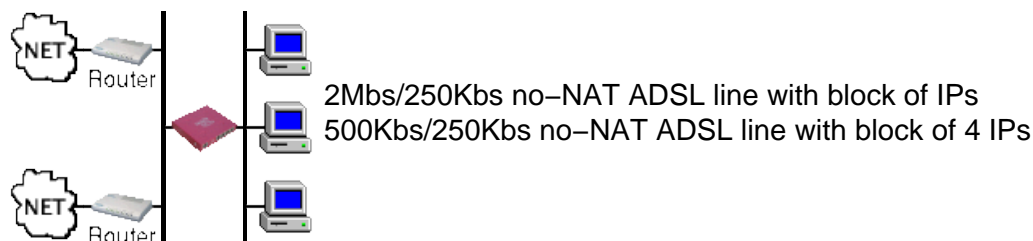
The LAN subnet should claim to be an address (the gateway address) and allocating a single DHCP address (the WAN address). Using DHCP restrict ensures this will only be issued to a machine called "SERVER".

The PC called SERVER should be set to collect IP automatically (DHCP), and should get the FireBricks WAN address allocated to it on the LAN

The port map ensures the FireBrick will pass on packets from the internet to the internal PC.

Other PCs get private addresses by DHCP and are NATed.

Multiple ADSL lines using bonded uplink



In this configuration a customer has a no-NAT 2Mb/s ADSL line (with 250Kb/s uplink) and a large block of IPs so that machines on the LAN have real addresses. The 2Mb/s ADSL is normally used, and the 500Kb/s ADSL is a backup and to provide additional uplink capacity.

- 2Mb/s ADSL router has an address A
- FireBrick allocated an address on 2Mb/s router subnet, address B
- 500Kb/s ADSL router has an address C
- FireBrick allocated an address on 500Kb/s router subnet, address D
- Network address for 2Mb/s ADSL line is address E

Basic IP setup :-

1. First subnet, LAN, no NAT, no Stealth, using address B. This gives machines on the LAN real addresses on 2Mb/s line
2. Second subnet, WAN, no NAT, Stealth, using address B. This allows the FireBrick to see router on address A
3. Third subnet, WAN, no NAT, no Stealth, using address D. This allows the FireBrick to see router on address C
4. Routing entry, LAN to WAN for address A, proxy ARP. This allows machines on the LAN to see router address A
5. Equipment on the LAN to use the 2Mb/s ADSL subnet's addresses and FireBrick address B as their gateway.

This basic setup allows machines on the LAN to have real addresses.

Gateway setup :-

1. Default gateway set to address E
2. Gateway alternative list set to addresses A and B

This means that all traffic from to the internet will use the pseudo address E, which is mapped to A and B alternatively for each packet allowing a bonded uplink of 500Kb/s for outgoing traffic. The pseudo address is used because if the router address such as A was used, then the profile based re-routing to use the 2Mb line would using gateway A would still be mapped to both gateways which would not work if one was down. By using a pseudo address, this is avoided and you can route to A, C or both (using E) based on routing rules as necessary.

Fallback setup :-

1. Profile (2MBADSL) monitoring an internet address, such as routers WAN address, via address A on WAN, set to alert when inactive and reroute on change
2. Profile (500KADSL) monitoring an internet address, such as routers WAN address, via address C on WAN, set to re-route.

3. Add route between subnets and gateway Any->WAN, gateway A, profile Not 500KADSL
4. Add route between subnets and gateway Any->WAN, gateway C, profile Not 2MADSL with NAT

This means if the 500K ADSL fails, the default route changes to A and traffic continues only via 2Mb ADSL.

If the 2Mb ADSL fails, the default changes to C and traffic continues via the 500Kb/s ADSL but NATed to ensure replies arrive.

Email alerts of profile changes are recommended.

If you have SMTP incoming email, then you may want to set FireBrick address D as an additional lower priority MX record target, and have a port map for this address to your mail server allowing incoming mail even if the main 2Mb link fails.