

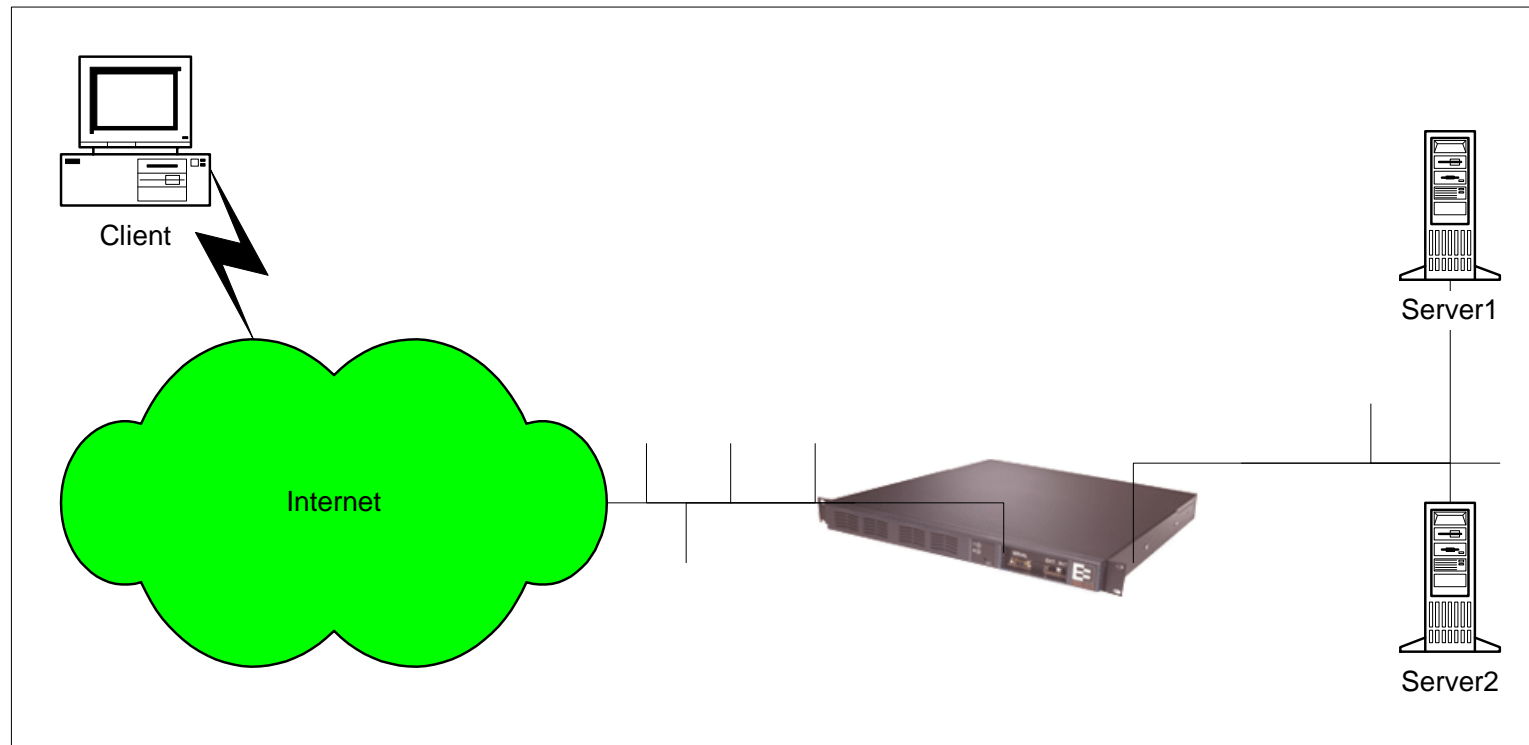
Load Balancing Implementation

- How Load Balancing Works
- Simple Applications
- Practical Implementation Using Coyote Point Equalizer
- Advanced Implementation Issues
 - Redundancy: configuring a redundant pair of Equalizers
 - Geographic load balancing
 - Accessing load balanced services from within the server network
 - Active Content Verification
 - Session state (sticky connections, persistence etc)
 - SSL
 - Match Rules

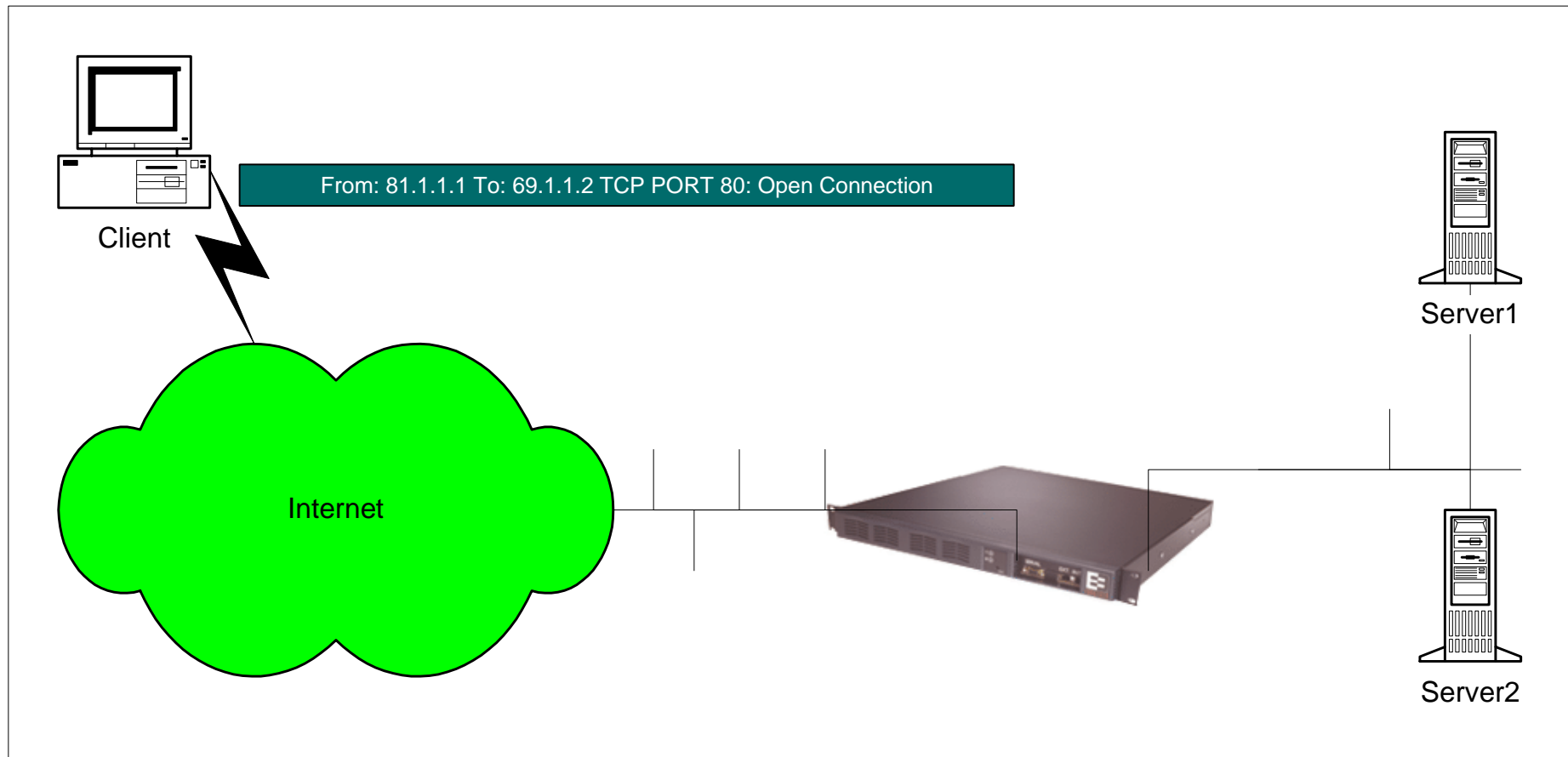
L3/4 Load Balancers:

- Name comes from layer in ISO Model, network or transport layer redirection of packets
- Uses similar technology to reverse NAT at a packet level
- Network (IP) and Transport (TCP/UDP) headers re-written from a single external IP address/port combination to one of several potential internal server addresses
- Couple the above with dynamic algorithms to determine load and probe current operational status in order make forwarding decision.

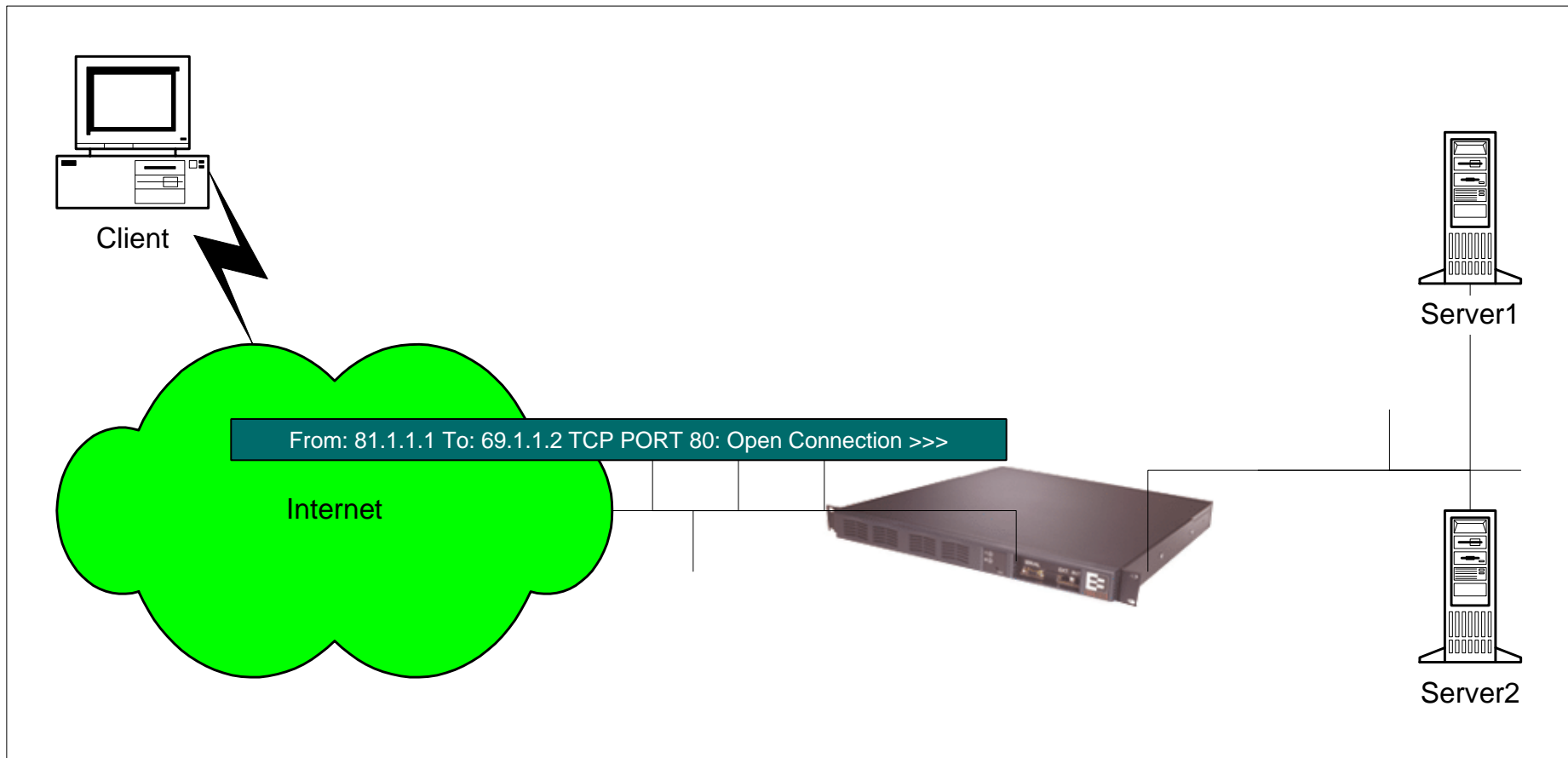
- Simple Architecture:
 - Two servers behind Equalizer LB
 - Equalizer is default router to Internet for servers



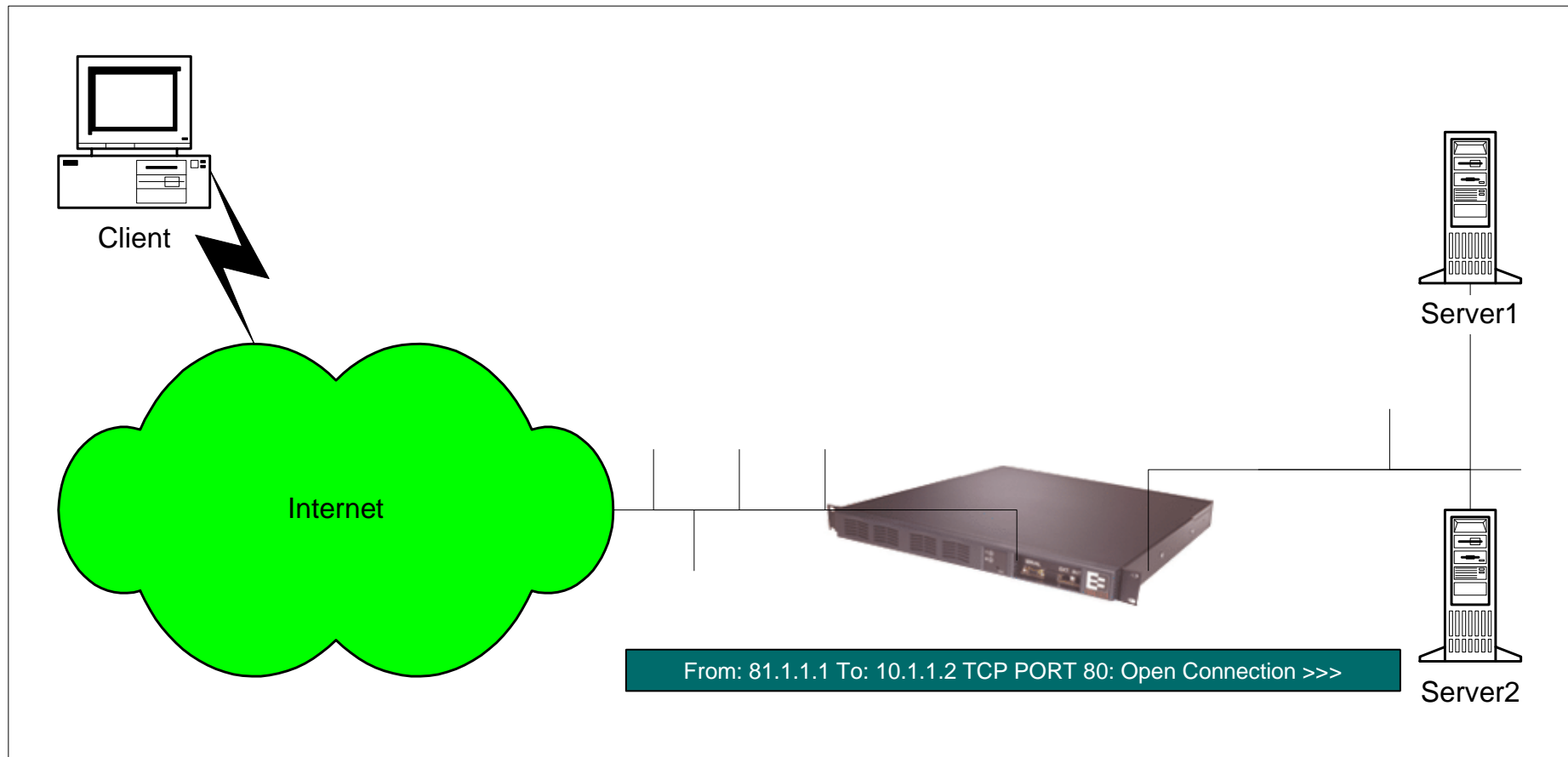
- Client makes request to `www.mysite.com`, resolves to address `69.1.1.2` configured on LB:



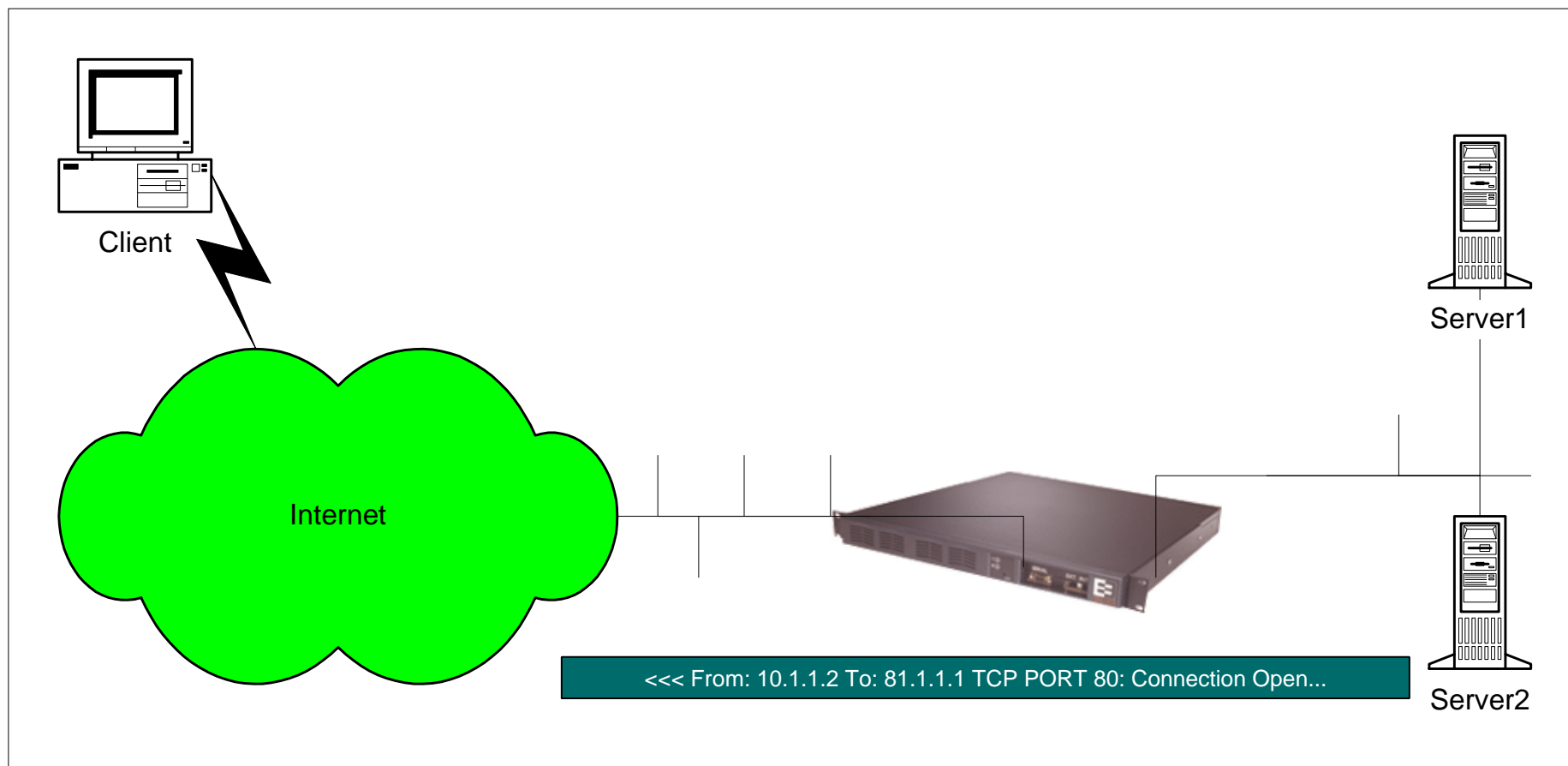
- Equalizer on 69.1.1.2 gets request



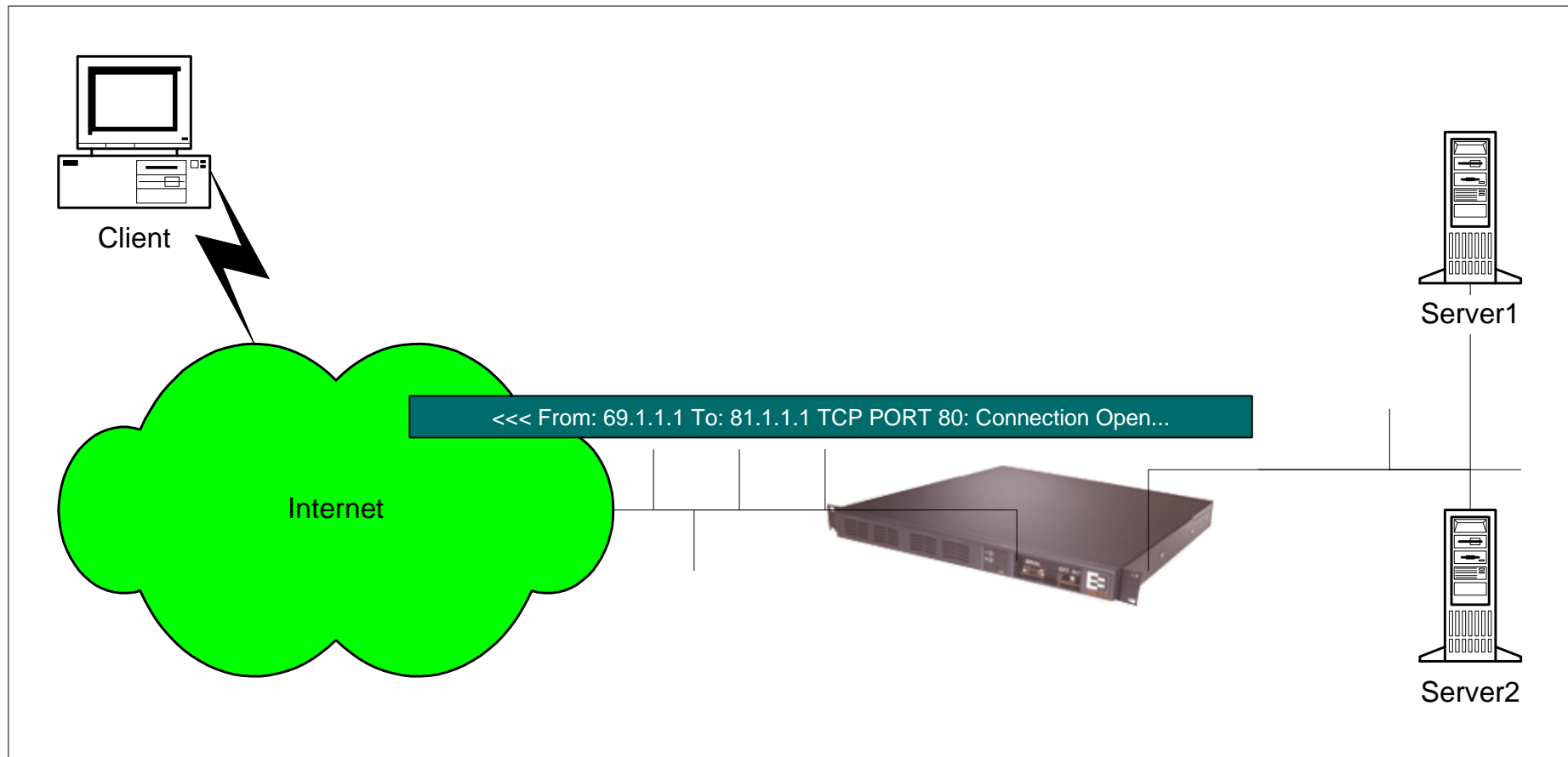
- Determines Server2 is least loaded, re-writes header...



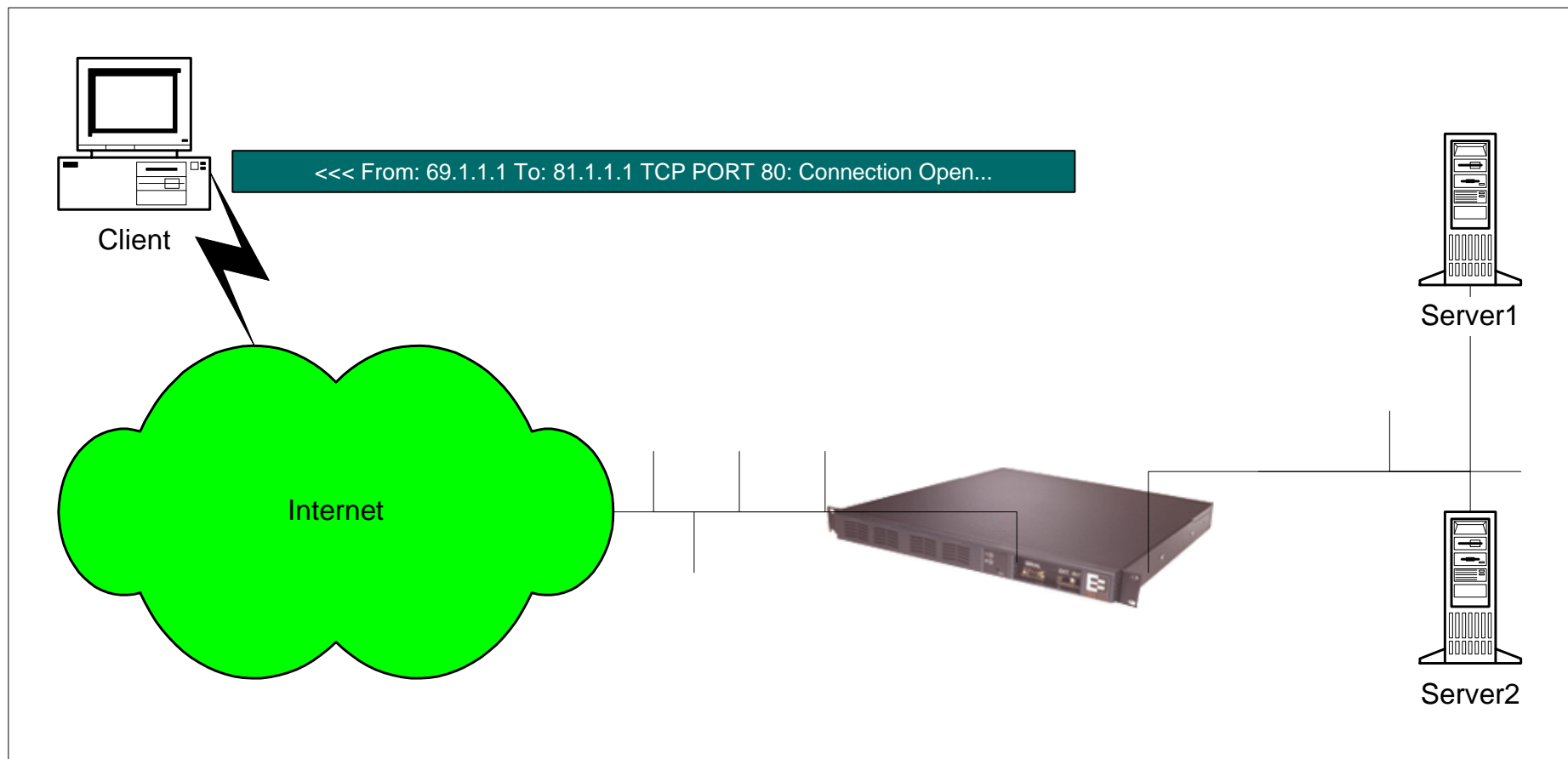
- Server2 responds



- LB re-writes header with external IP address...



- Client sees response from external server address



- Load Balancer maintains state for active conversations, so that TCP traffic for a given src addr/port, dst addr/port conversation is always re-written to same server.
- Stale session timeout catches cases where conversation is incomplete to drop state and avoid clogging translation tables.
- UDP behaves in a similar way.
- Works with any TCP protocol which can pass through NAT, and stateless UDP protocols.

Algorithm to distribute transactions across servers:

- Basic up/down availability polling by LB combined with...
- Round Robin – simplest
- Static weight – specific proportion of requests go to each server
- Measured response time – LB measures time each server is taking to respond to requests and weights “best”
- Least connections – LB forwards more transactions to server with least current connections
- Server agent – software installed on server “tells” load balancer what it's current status/perceived load is

Adaptive load balancing based on measured values (i.e. response time, current connections, server agent) needs to incorporate damping factor to avoid oscillation.

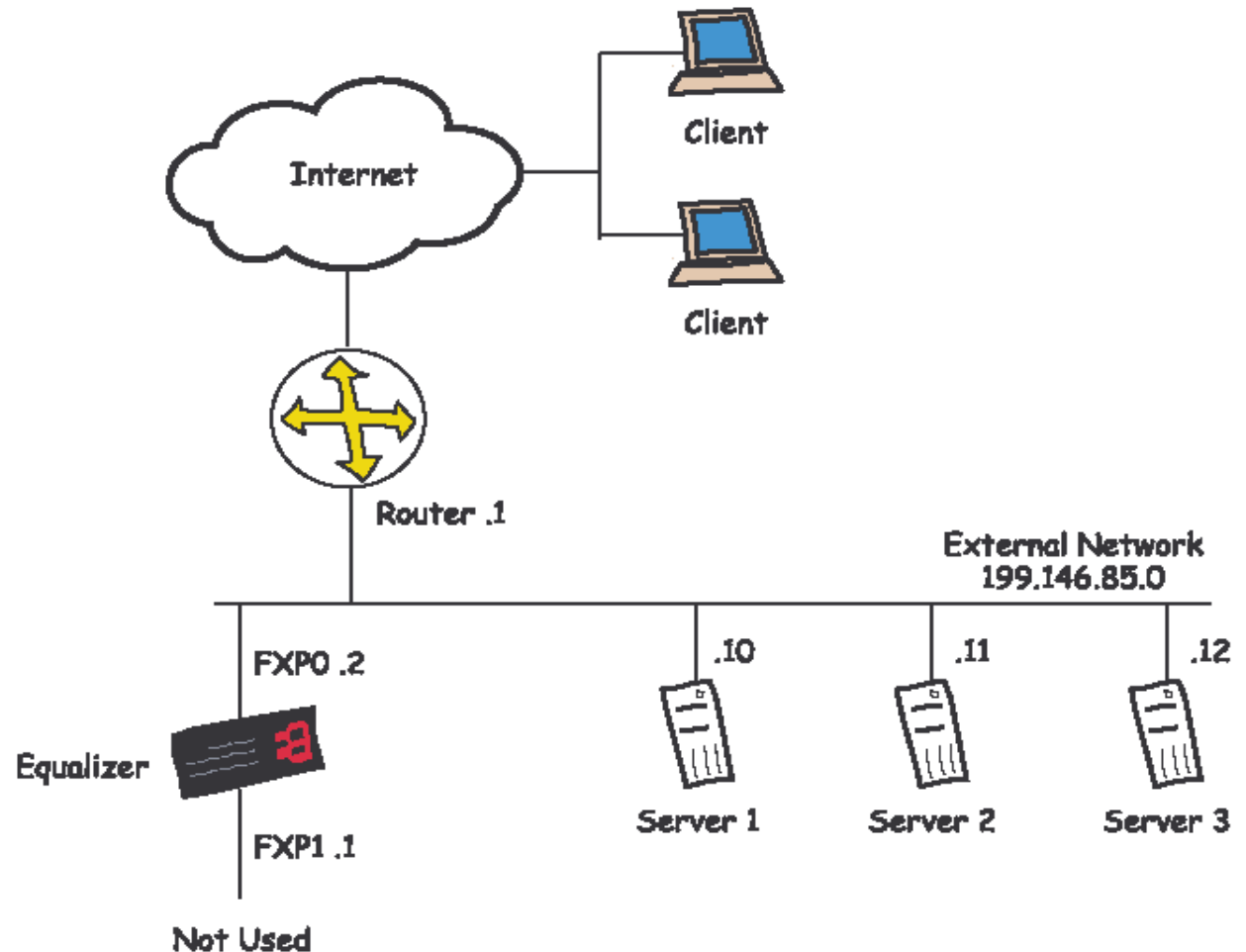
Conversely if the adaptive response is too slow then dynamic weight will not alter fast enough when server load alters and response slows.

- Nearly all TCP protocols:
 - HTTP, SMTP, FTP, NNTP etc etc etc
- Only danger areas are where end-point IP addresses are carried in protocol as NAT is involved.
- Care needs to be taken when underlying application is stateful (web applications) or depends on data store (e.g. Databases, POP mailboxes etc):
 - Can have problems if two consecutive transactions go to different servers.
 - Strategies like sticky connections (more later), sharing state between servers, or using “hot standby” servers can help here.

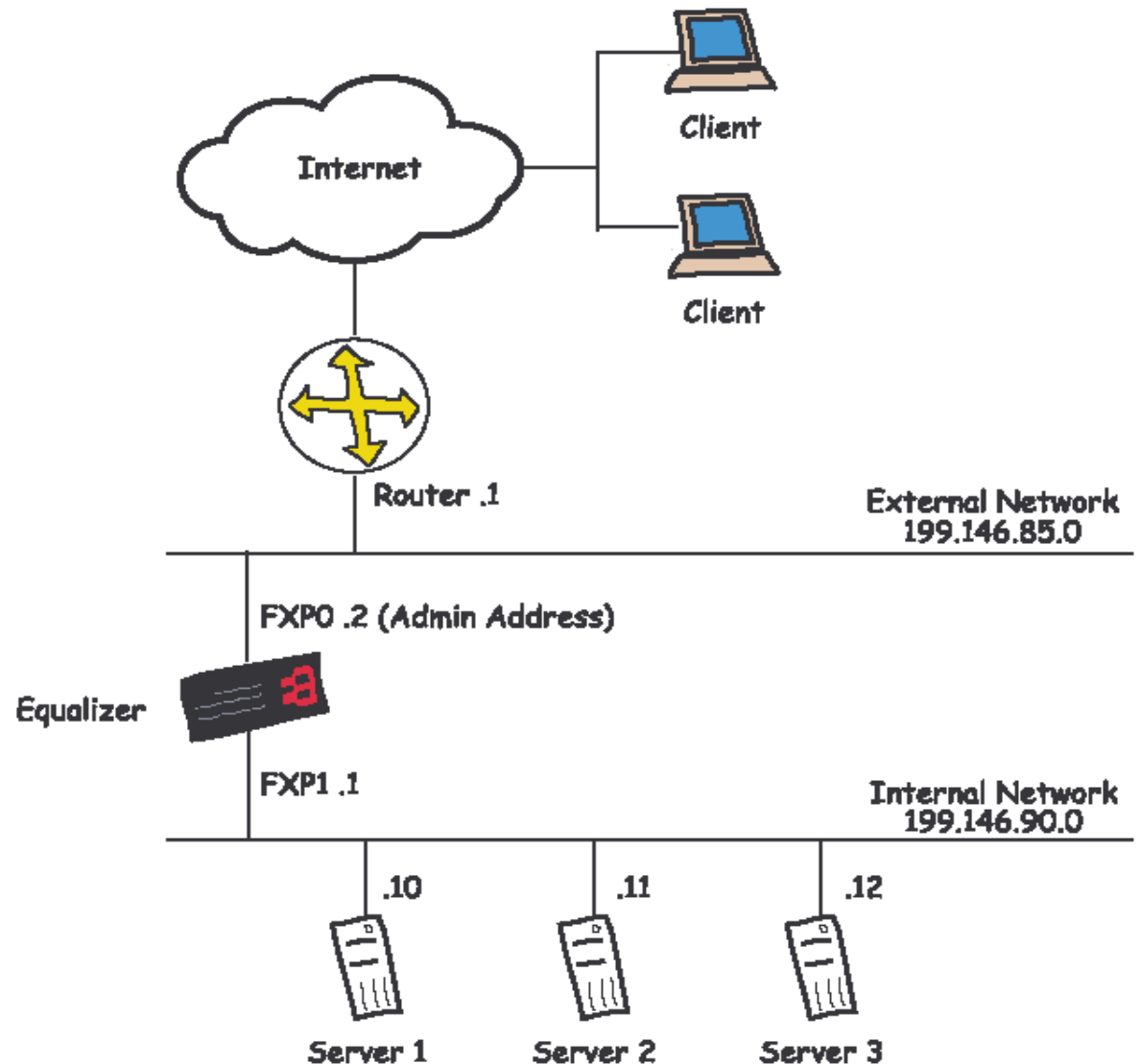
- L7 Clusters:
 - Load balancer actually listens on HTTP(S) socket and accepts initial TCP SYN and first data segments from client itself.
 - Defer LB decision so that it is based on HTTP headers once conversation has started.
 - This is somewhat different from L4 load balancing where only packet headers are being inspected, and forwarding decision has been made before data flows.
 - Persistence can now be based on Cookies rather than client IP address *and be preserved across HTTP/HTTPS transition*.
 - Solves AOL/Freeserve problem where IP address of proxy hops around and changes between HTTP/HTTPS

- Using Coyote Point Equalizer and two web-servers to form a load balanced server farm
- Two possible architectures with single or dual network

- Equalizer has external IP address on fxp0
- Servers have Equalizer as default route
- One or more clusters configured on Equalizer



- Equalizer has external and internal IP address
- Servers have Equalizer as default route
- Router has static route via Equalizer to servers
- One or more clusters configured on Equalizer



- In single network, **everything** has a real routable IP address
- In dual network, Equalizer acts as gateway, either:
 - two separate subnets of routable IP space required with Equalizer configured as a static route to Internal subnet at boundary router or
 - Server network numbered from private (RFC1918) space and Equalizer configured to do outbound NAT (only required if servers need direct network access)
- In either case, Equalizer requires one external IP address for management + any addresses required by external clusters, i.e. minimum of 2 external IP addresses

- Choose network architecture: single or dual network
- Collect network information: Allocate IP address on external (+ internal?) network, identify netmask, default router, name server etc
- Use serial cable to perform initial configuration of Equalizer
- Log into Equalizer via web interface, set passwords, check configuration (add outbound NAT if required)
- Configure Equalizer as default route on servers
- Check connectivity between servers and external router etc

- Called “virtual clusters” with four possible types:
 - “Generic TCP” - Layer 4 TCP (protocol opaque)
 - “Generic UDP” - as above for UDP
 - HTTP - Layer 7 cluster type, Equalizer terminates and forwards HTTP sessions.
 - HTTPS - as above for encrypted sessions, Equalizer terminates SSL transaction *and forwards plain HTTP transaction to servers.*

- Allocate external IP address for a cluster
- Add a new HTTP cluster to Equalizer
 - Use default settings (adaptive load balancing etc)
- Add servers to cluster
- That's it!
- Test by HTTP access to external address
- Take servers down to test failover etc



Advanced Implementation Issues

- Redundancy: configuring a redundant pair of Equalizers
- Geographic load balancing
- Accessing load balanced services from within the server network
- Active Content Verification
- Session state (sticky connections, persistence etc)
- SSL
- Match Rules
- Anything else...

Redundancy

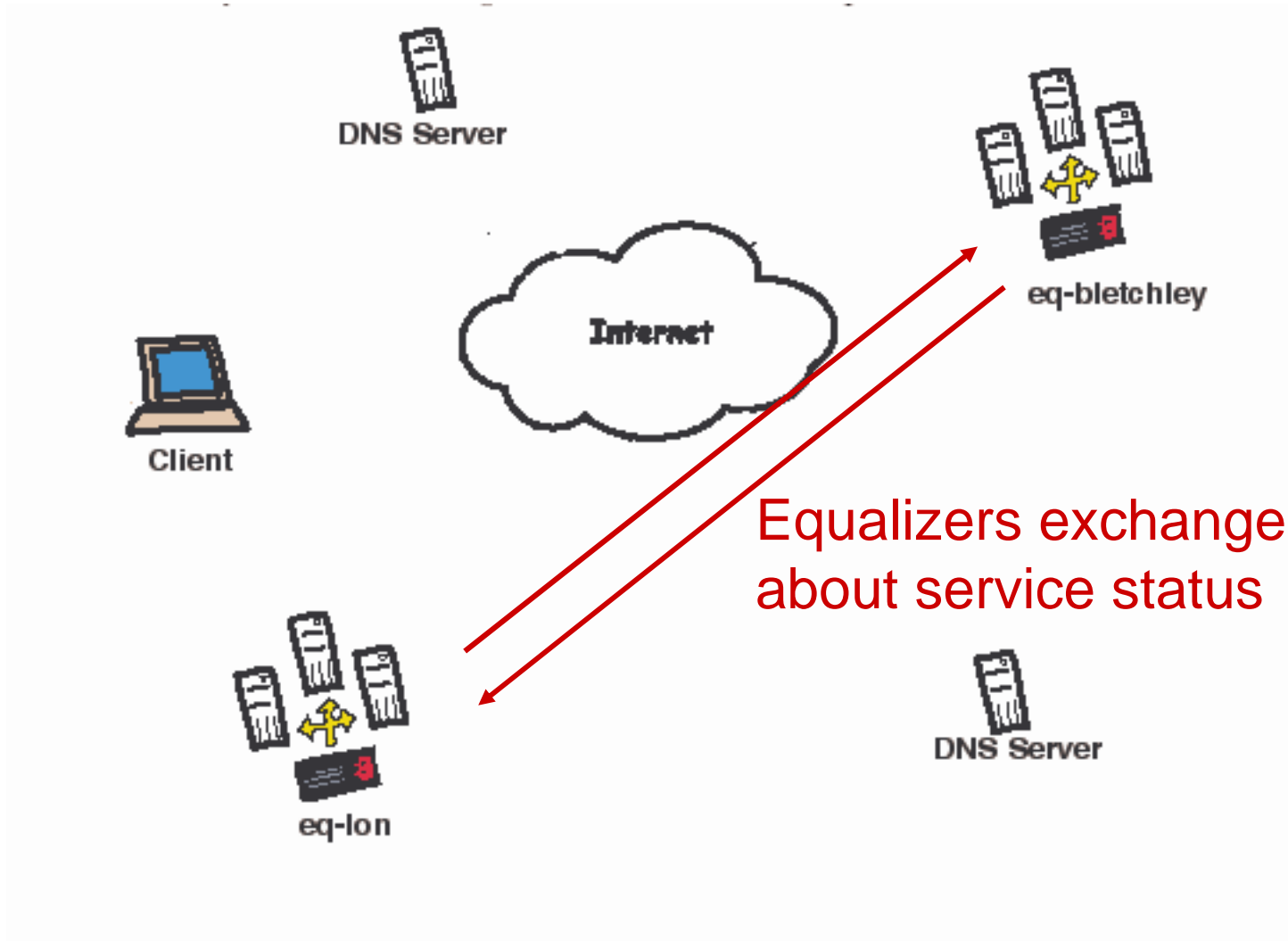
- Redundant pair of Equalizers allows service to survive failure of one of the units.
- At any point in time, one Equalizer is primary and operating, backup monitors primary and is ready to take over if it fails.
- Each Equalizer has it's own External (+ internal in dual network configuration) IP address.
- Additional shared gateway IP address is allocated to be assumed by whichever Equalizer is primary.
- Shared gateway address is used as default gateway by servers.

- Perform basic network configuration on both Equalizers
- Ensure all configuration options (except addresses) are identical
- Allocate extra IP address for shared gateway address
- Log in to web interface one Equalizer and select “Configure Redundancy”.
- Enter “Default Primary” for Failover Role
- Select network type and enter **sibling** IP address(es)
- Enter shared Failover Gateway address and Click Set
- Repeat above on other Equalizer, but select “Default Backup” for Failover Role.

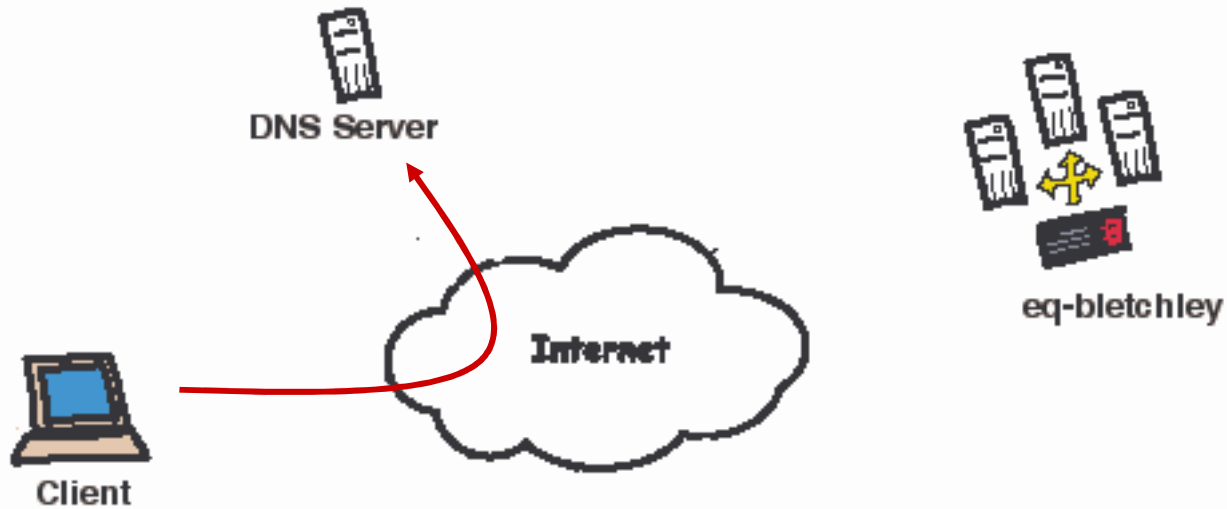


Geographic Load Balancing

- Equalizers use Envoy to do DNS based load balancing
- Nameservers delegate records to Equalizers for domain:
`www.ipcortex.co.uk NS eq-lon.ipcortex.net.`
`www.ipcortex.co.uk NS eq-bletchley.ipcortex.net.`
- Clients query nameserver, get delegation to Equalizer and query it.
- Equalizer returns “best” IP address for client and service...



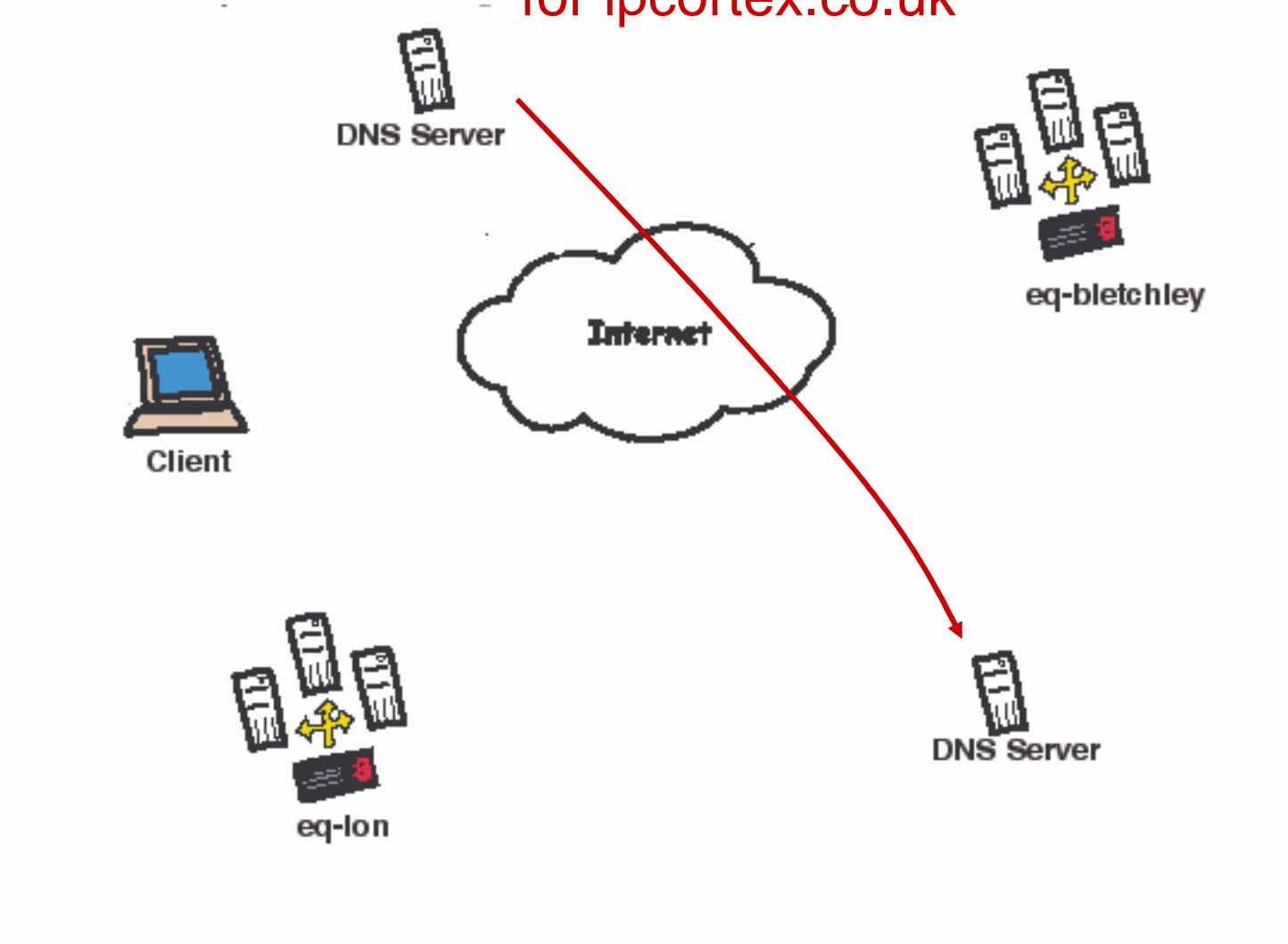
Equalizers exchange information about service status

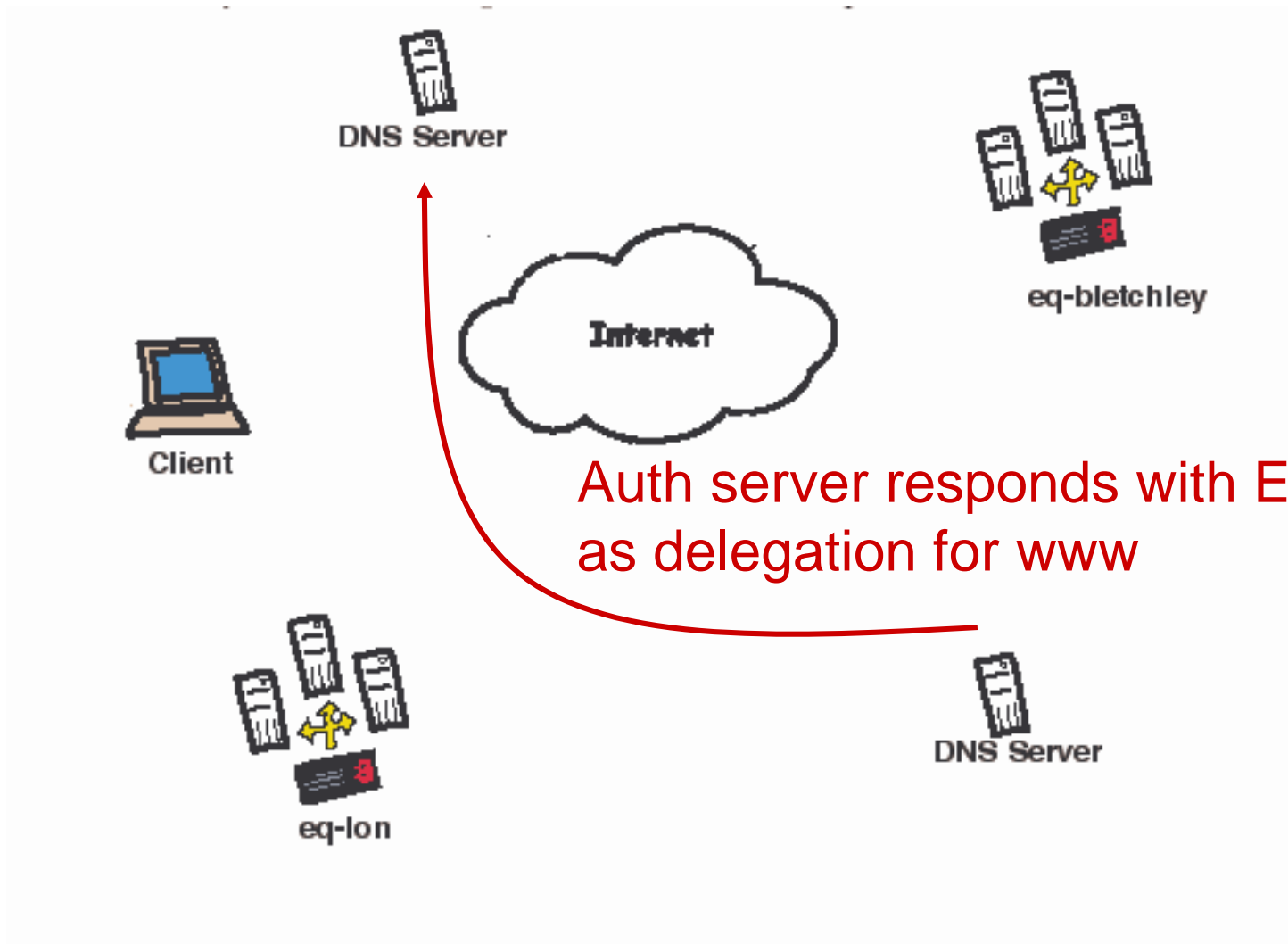


Client requests www.ipcortex.co.uk
from local DNS server

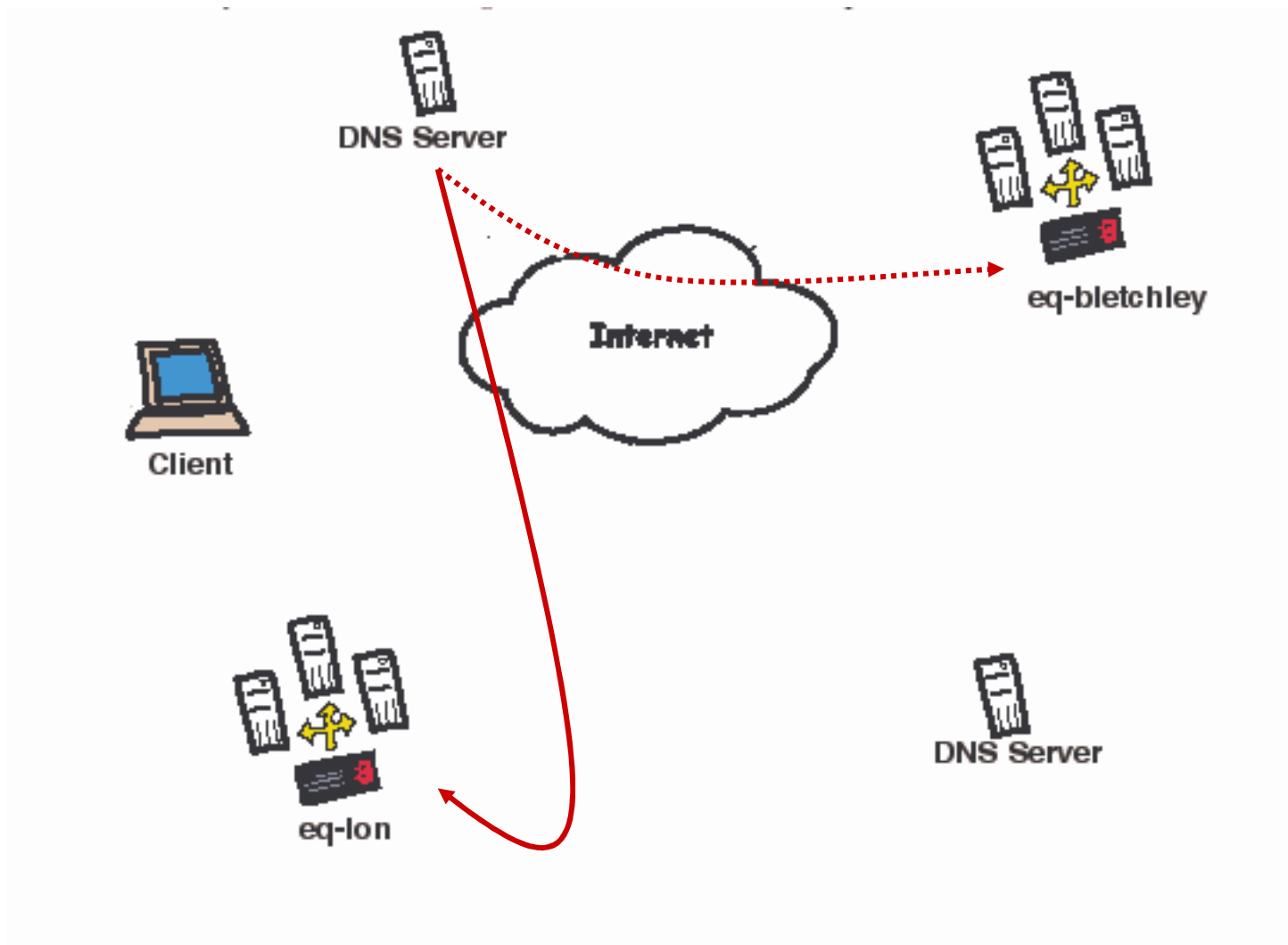


Local DNS server queries auth server for ipcortex.co.uk

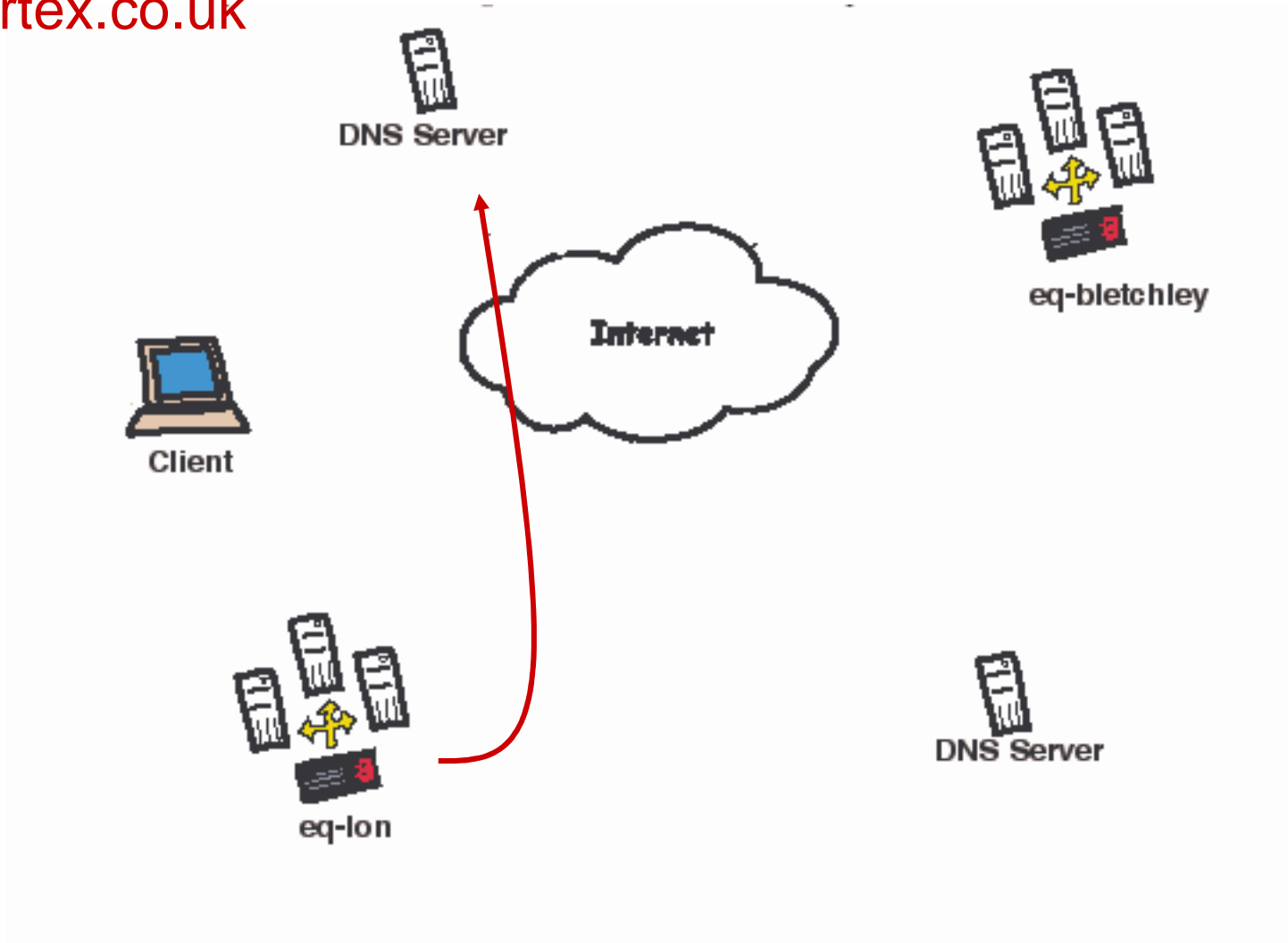


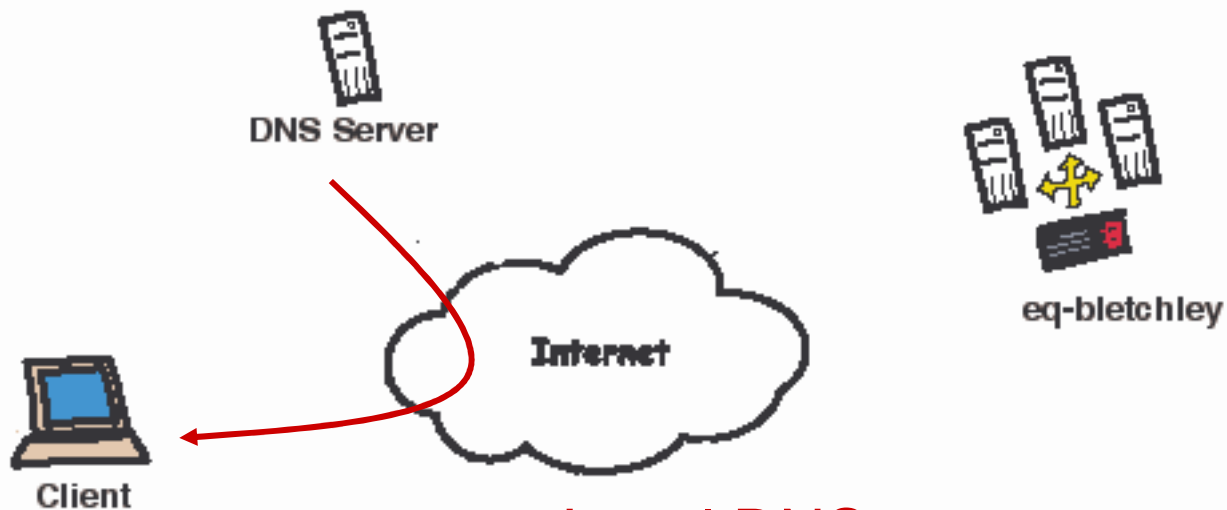


Local DNS server queries Equalizers for www.ipcortex.co.uk

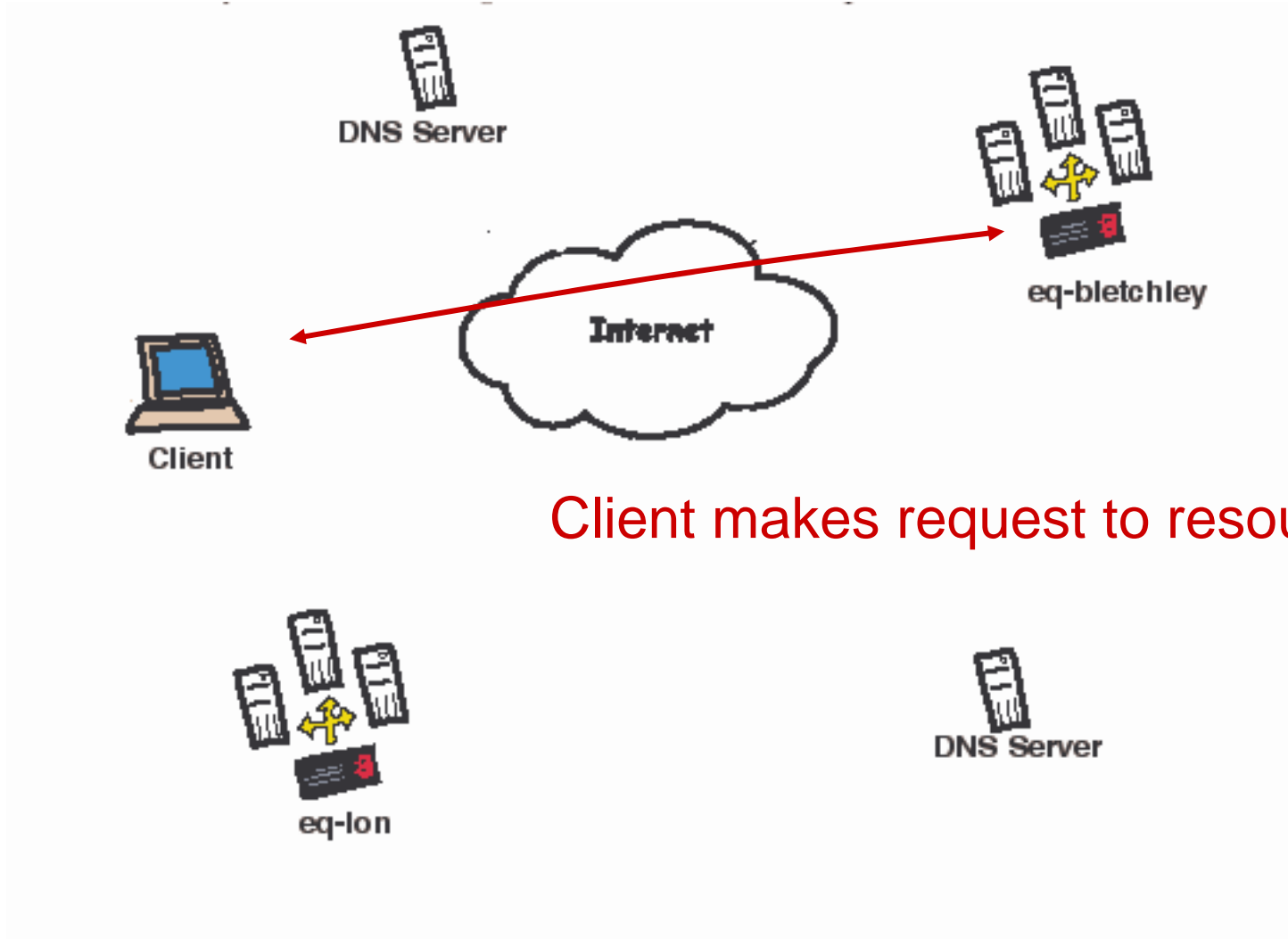


Equalizer responds with IP address of best service location for www.ipcortex.co.uk





Local DNS server responds to client with IP address



- DNS records have a Time to Live (TTL)
- After TTL expires, client re-queries and whole process starts again
- “New” clients will always get “best” address
- Existing clients with records which have unexpired TTL will wait until this decrements to zero before re-querying
 - Short TTL improves failover
 - Also less efficient as client and servers have to do *lots* of querying
 - Typical tradeoff in region of 1-15 mins
- Some clients (and ISPs) don't honour standards so records will be retained for too long

Domain name to respond to

Add Geographic Cluster

Geographic Cluster Name: (FQDN)

TTL: (seconds)

MX Exchanger:

Load Balancing Method:

Load Balancing Response:

Add Geographic Cluster

Geographic Cluster Name: (FQDN)

TTL: (seconds)

MX Exchanger:

Load Balancing Method:

Load Balancing Response:

TTL

Add Geographic Cluster

Geographic Cluster Name: (FQDN)

TTL: (seconds)

MX Exchanger:

Load Balancing Method:

Load Balancing Response:

Mail Exchanger
Probably not required
unless mail to
name@www.domain is
used

Add Site

Geographic Cluster: www.ipcortex.co.uk

Site Name:

Site Address:

Peer Default Site

Keepalive: (seconds)

Static Weight:

Resource Address:

Port:

Agent Address:

External address of site, as served by DNS server.

NAT: this is the Internet IP address of this site.

Add Site

Geographic Cluster: `www.ipcortex.co.uk`

Site Name:

Site Address:

Peer Default Site

Keepalive: (seconds)

Static Weight:

Resource Address:

Port:

Agent Address:

Peer = site is same preference as all others

Default Site = site will be used if no other site is clearly better

Add Site

Geographic Cluster: `www.ipcortex.co.uk`

Site Name:

Site Address:

Peer Default Site

Keepalive: (seconds)

Static Weight:

Resource Address:

Port:

Agent Address:

Keepalive: how often to probe the resource (in seconds)

Add Site

Geographic Cluster: `www.ipcortex.co.uk`

Site Name:

Site Address:

Peer Default Site

Keepalive: (seconds)

Static Weight:

Resource Address:

Port:

Agent Address:

Static weight of site (higher is greater preference)

Add Site

Geographic Cluster: www.ipcortex.co.uk

Site Name:

Site Address:

Peer Default Site

Keepalive: (seconds)

Static Weight:

Resource Address:

Port:

Agent Address:

Address and port this resource is known by *to its local Equalizer* – used to identify which cluster is being monitored

Add Site

Geographic Cluster: `www.ipcortex.co.uk`

Site Name:

Site Address:

Peer Default Site

Keepalive: (seconds)

Static Weight:

Resource Address:

Port:

Agent Address:

Agent address: address of Equalizer on which the resource resides

NAT: Local IP address if this Equalizer, public IP address if remote.

- Equalizers communicate on UDP port 5300, 5301 between sites to poll resource status
- Site address is IP address to be published in DNS for this site
- Resource address and port number is identifier for cluster on site's Equalizer
- Agent address is the Equalizer on which the resource resides and must be addressable from config Equalizer
 - Local NATed IP address of local agent (Equalizer)
 - Public IP address if remote agent (only way to reach it)



Accessing Services from Within the Server Network



Accessing Services from Within the Server Network

- All packets between client and server for a conversation need to flow through the Equalizer for a load balanced service to work
- Accessing services from machines on the server network will break this
- Problem is that *response* packets are usually routed back direct to client if they are on same subnet and Equalizer can't re-write headers
- Solution is to add a /32 route on all servers for all potential clients:
server2% route add -host server1 equalizer
C:\ route add server1 equalizer



Active Content Verification

- By default up/down probe simply verifies server is accepting connection
- ACV is a powerful way of probing to ensure that the server is delivering meaningful results not just accepting TCP connections
- At poll time, ACV probe string is sent and Equalizer waits 200ms for response
- If first 1000 bytes contains response pattern then server is marked as up, otherwise marked as down

- HTTP
 - Probe: “GET /index.html HTTP/1.0\r\n\r\n”
 - Response: “200 OK”
 - Probe: “GET /status.jsp HTTP/1.0\r\n\r\n”
 - Response: “<body>Status: OK</body>”
- SMTP
 - Probe: “HELO Equalizer.foo.bar\r\n”
 - Response: “250 mailserver.foo.bar Hello”

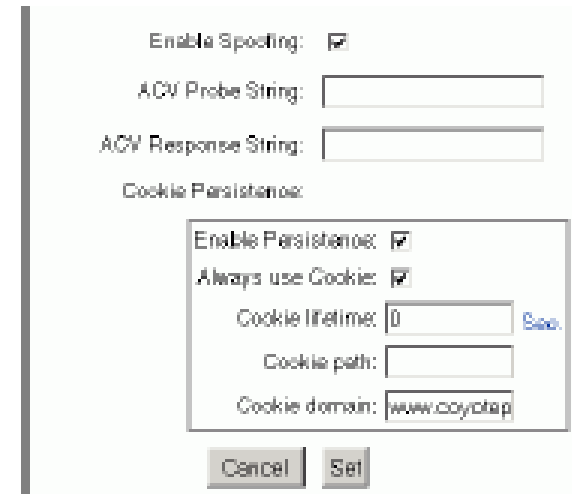


Session State

- Problems when one server holds application state and transactions spread over multiple TCP connections (e.g. User login status)
- Transactions may go to different servers and state lost, two solutions:
 - share state between servers – best as it allows perfect load balancing, but affects application architecture and implementation
 - use stickiness to direct multiple connections to same server – simple, requires no application support but not perfect
- Subnet stickiness can be used to cover proxies etc

- Cookie Persistence

- Optionally inserts Cookie into HTTP responses if it sees at least one from server
- Always inserts cookie if option set (even if none from server)
- Cookie domain, path and lifetime values just go into cookie header and are honoured by client
- All TCP sessions on servers now originate at Equalizer, “Enable Spoofing” setting allows retention of original client IP address (server logs!).



The screenshot shows a configuration window for 'Cookie Persistence'. It contains the following fields and options:

- Enable Spoofing:
- ACV Probe String:
- ACV Response String:
- Cookie Persistence:
 - Enable Persistence:
 - Always use Cookie:
 - Cookie lifetime: [See...](#)
 - Cookie path:
 - Cookie domain:
- Buttons: Cancel, Set



SSL

Two ways of dealing with SSL

- Treat as opaque TCP traffic to port 443
 - Certificates reside on servers
 - L7 Load Balancing not possible
- Install SSL certificate on Equalizer
 - Only one certificate required (on Equalizer)

-
- Layer 7 functionality is useless for SSL connections unless Equalizer actually acts as end-point of SSL session.
 - Without SSL decryption, Equalizer just sees opaque encrypted traffic stream so only L4 load balancing decisions are possible.
 - By terminating SSL at Equalizer, full HTTP headers etc are available for L7 forwarding decisions.
 - Equalizer uses HTTPS (port 443) SSL encapsulated towards client, and plain HTTP towards servers.
 - CA signed SSL certificate installed on Equalizer not on web servers
 - Side effect is that this can save cost with per server certification policies of most commercial CAs

- Create cluster in same way as HTTP.
- Red warning text indicates that cluster will not operate until certificate is uploaded.
- PEM is easy if you originally created cert using OpenSSL based systems (e.g. Apache, mod_ssl etc) as this tends to be default.
- If using IIS enrolment then need to export to PKCS#12
- Ensure private key can be passphrase protected but this is stripped on upload (use XCEL card for additional certificate security)
- *Test with browser and check certificate details.*



Match Rules

- Powerful facility that allows requests to be sent to subset of servers based on attributes of request header:
 - Hostname or component (*virtual server based match rules*)
 - Pathname or directory (e.g. *all accesses below /images/ go to static page server etc*)
 - Filename (e.g. *.php goes to one group of servers, .html to another within same site*)
- *Rules are flexible and powerful (including regexp on any HTTP header), but use with care as a complex expression with regexp can sap LB throughput.*
- *New virtual cluster always has a default match rule.*
- *Single server can only be present in one rule.*

- Questions...

- Formal support:
support@ipcortex.co.uk
<http://www.ipcortex.co.uk/>
01908 276654