NetworkAppliance®
The evolution of storage.™

Understanding and Troubleshooting Routing Loops in NetCache® Appliance Deployments

Afonso Infante, Network Appliance B.V. | May 2003 | TR-3262

TECHNICAL REPORT

Network Appliance, a pioneer and industry leader in data storage technology, helps organizations understand and meet complex technical challenges with advanced storage solutions and global data management strategies.

Abstract

Since version 5.0, the NetCache appliance software has been able to detect and report routing loops in the network where it is deployed. This document explains what routing loops are, how they are detected, and gives some examples and steps on how to handle them.

Table of Contents

1. Executive Summary

The sophisticated Network Appliance™ NetCache appliance can be deployed in complex networks, with complex routing and/or switching configurations. Misconfigurations of these complex networks can cause so-called *routing loops* that may make it impossible for the NetCache appliance to serve requests to Internet clients. A routing loop is a situation where a certain protocol request made by the NetCache appliance is sent back multiple times to that same NetCache appliance instead of going to the destination server.

In NetCache appliance software version 5.0, Network Appliance introduced a mechanism to detect potential evidence of routing loops, to help the customer troubleshoot the problems in the network. This mechanism also prevents routing loops from creating denial of service situations on the NetCache appliance.

With the exception of misconfigured NetCache appliance hierarchies, routing loops are almost always caused by misconfigurations in routers or layer 4/7 switches. They are most common in Transparent proxy deployments, in particular if IP spoofing is also implemented, but can also happen in other scenarios, including (but not limited to) load balancing deployments and when proxy hierarchies are in place.

Troubleshooting routing loops can be a very complex process, and it mainly involves looking very carefully at the network diagram and checking (and correcting) the routing tables and/or switching rules in the network's active equipment.

Experience has shown that routing loops are best prevented, and therefore it is highly recommended to plan the deployment of the NetCache appliance and any necessary changes in routing and/or switching bearing this potential problem in mind, in particular if transparency and IP spoofing are to be used.

2. Introduction

In NetCache appliance software version 5.0, Network Appliance introduced a mechanism to detect potential evidences of a routing loop in the network where it is deployed. Usually, this mechanism will log (hourly, even if there are multiple occurrences during an hour) a warning in the messages log, which, in NetCache 5.3.1 software, can be either:

1. `Connection dropped due to a potential loop: src_ip: <a.b.c.d>, dst_ip: <a.b.c.d>`
2. `Potential routing loop detected for URL("<URL>")in HTTP Via header: <Via: Header>`
3. `Potential routing loop detected in HTTP Via header: <Via: Header>`
4. `Potential routing loop detected for URL ("<URL>") in HTTP X-Forwarded-For header: <X-Forwarded-For Header>`
5. `Potential routing loop detected in HTTP X-Forwarded-For header: <X-Forwarded-For Header>`
6. `Potential routing loop detected in MMS Via header: <Via: Header>`
7. `Potential routing loop detected in MMS-HTTP Via header: <Via: Header>`

8. `Potential routing loop detected in RTSP Via header: <`*`Via: Header`*`>`
9. `Protocol tunnel connection dropped because of a potential routing loop. src_ip:<`*`a.b.c.d`*`> dst_ip: <`*`a.b.c.d`*`> in_port:<`*`xx`*`> out_port:<`*`xx`*`>`
10. `Protocol tunnel connection dropped due to a potential loop: src_ip: <`*`a.b.c.d`*`>, dst_ip: <`*`a.b.c.d`*`>`

FTP is one exception; in the case of this protocol, the FTP client will get this error when the NetCache appliance detects a potential routing loop:

```
421 Service not available, routing loop detected.
```

The following error message will be logged:

```
FTP: routing loop detected.
```

HTTP-based tunneling can also generate another type of routing loop message:

```
[sthread_loop:warning]: Client IP: <a.b.c.d> trying to issue a CONNECT
looping back to Appliance
```

(In other 5.x versions of the NetCache software, only *some* of these messages/situations may be triggered, however, the ones that are will mean the same.)

HTTP requests that the NetCache appliance believes went through a routing loop are, of course, dropped (the connection is closed), since if they are left in a loop they can generate a denial of service-type attack on the NetCache appliance. Therefore, in most cases, the respective Web access log will show `CLIENT_CLOSE` in the *`x-note`* field (if that field is configured to be present—it is by default for the Web access log default format) for that request.

3. Definition and Detection

A routing loop is a situation where a certain protocol request made by the NetCache appliance is erroneously sent back multiple times to that same NetCache appliance instead of going to the destination server, usually by a misconfigured router or layer 4 – 7 switch. There are several ways the NetCache appliance detects such a potential occurrence:

1. The source IP of an *incoming* request belongs to this same NetCache appliance. In this case, messages 1 or 10 above can be logged, depending if it is a protocol tunnel request (10) or not (1).
2. The source IP of an incoming tunnel request belongs to this same NetCache appliance *and* the destination port of that incoming request (<*in_port*>) is the same as the destination port of an existing TCP connection on the NetCache appliance (<*out_port*>). Message (9) is therefore logged.
3. The *Via:* header is defined in the RFC that defines HTTP[1]:

[1] RFC 2616, available for example in ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt

```
The Via general-header field MUST be used by gateways and proxies to
indicate the intermediate protocols and recipients between the user
agent and the server on requests, and between the origin server and
the client on responses. It is analogous to the "Received" field of
RFC 822 [9] and is intended to be used for tracking message forwards,
avoiding request loops, and identifying the protocol capabilities of
all senders along the request/response chain.
```

If this *Via:* header is very long (over 1,300 bytes) the NetCache appliance will assume the request is bouncing between gateways and proxies (namely, being sent back to the same NetCache appliance) and therefore chances are a routing loop is occurring. Message (2) or (3) shown in the introduction will be logged: (2) if a URL is available (Request), (3) if it is not (Response).
Please bear in mind that if you disabled the *Via:* header from requests (by setting “Suppresses reference to this appliance on the HTML Via header in requests” in “*Setup > HTTP > General > Suppress Writing to Via Header request:*” in the appliance manager interface or setting `config.http.novia = on` in the CLI) the *Via:* header will not grow in routing loops, and therefore this particular detection algorithm will never be triggered (unless there are other proxy /gateways in the loop that are appending to the *Via:* header in requests). It is also important to have the *Via*: header appended to responses, since, although less common, response routing loops may also occur. The CLI option to have them enabled (the default) is `config.http.noviaresp = off`

4. The *Via:* header in an MMS request is over 800 bytes long. As above, but for an MMS request. Message (6) or (7) will be logged, depending on whether it’s pure MMS or MMS-over-HTTP.
5. The *Via:* header in an RTSP request is over 800 bytes long. As above, but for an RTSP request. Message (8) will be logged
6. The *X-Forwarded-For* header of an HTTP request is over 160 bytes long. The *X-Forwarded-For* header contains the IP address of the original clients, followed by the IP addresses of all the proxies it might have been through. For example, it can start as:

   ```
   X-Forwarded-For: 123.321.123.1
   ```

   … and then, every time it goes through a proxy it will be appended, making a routing loop easily visible, similar to what happens with the *Via:* header (which, on the other hand, logs *identification strings*, not IP addresses). For example:

   ```
   X-Forwarded-For: 123.321.123.1, 123.321.120.1, 123.321.120.1,
   123.321.120.1, 123.321.120.1, 123.321.120.1, 123.321.120.1,
   123.321.120.1, 123.321.120.1, 123.321.120.1
   ```

   The above would actually cause the NetCache appliance to log a warning (4) or (5) – depending on whether the URL is available (Request) or not (Response) since it is over 160 bytes long.

   Please bear in mind that if you disabled the *X-Forwarded-For:* header from requests (by setting

"Enables HTTP privacy" in "*Setup* > *HTTP* > *General* > *Client Privacy Enable:*" in the appliance manager interface or setting `config.http.privacy = on` in the CLI) the *X-Forwarded-For:* header will not grow in routing loops, and therefore this particular detection algorithm will never be triggered (unless there are other proxy/gateways in the loop that are appending to the *X-Forwarded-For* header in requests).

4. Examples and Troubleshooting of Possible Routing Loop Scenarios

4.1. Transparency with IP Spoofing Enabled

The single most common scenario where a routing loop may be created occurs in networks where the NetCache appliance is deployed as a transparent proxy/cache, using a layer 4/7 switch or WCCP router or switch, and, additionally, IP spoofing is enabled.

IP spoofing deployments imply that the NetCache appliance will send the request to the server using the original client's IP address as the source IP address. Therefore, unless the layer 4/7 switch or WCCP equipment is configured very carefully, it won't be able to distinguish that request from the original one, and it may "think" that it is a client request and resend it to the same (or other, if there's round-robin load balancing in place) NetCache appliance. That NetCache appliance will again try to send it to the server, through the switch, which will then send it back again to the NetCache appliance, and so on, until the NetCache appliance—in this case, by the growth of the *Via:* or *X-Forwarded-For:* header—finally figures out that a routing loop is happening and it drops the request altogether.

This scenario is not only the most common, but also the most dangerous. If both the *Via:* header and the *X-Forwarded-For:* header were disabled (`config.http.novia = on` and `config.http.privacy = on`), which is extremely common in IP spoofing deployments, the NetCache appliance will actually never be able to detect the routing loop, and not only will a message not be logged, the routing loop will not be broken.

If this happens for every single request, the effects of such a situation will be quite obvious because the NetCache appliance will be unable to serve any request.

However, the switch may be configured in such a way that this only happens for *some* requests, coming to or from particular networks, for example. These will still go undetected if the *Via:* header and the *X-Forwarded-For:* header are both disabled, although it may still manifest itself by hanging requests, overloads (DoS) on the NetCache appliance, and requests logged with an *x-note* field of `RESOURCE_RECLAIMED`.

If this type of situation is suspected, the first step is to make sure that either (or both) the *Via:* header and the *X-Forwarded-For*: header are being appended to by the NetCache appliance (`config.http.novia = off` and / or `config.http.privacy = off`). This will make routing loops visible and stop them from looping infinitely.

By looking at the messages log it will be possible to see the path that the requests are going through. In this specific scenario, odds are that you will see the same NetCache appliance identification string repeated consecutively in the *Via:* header or the NetCache appliance IP repeated consecutively in the *X-Forwarded-For:* header.

If that is the case, the problem will be identified. The next step will be going through the configuration of the switch or WCCP router and trying to identify in which situations the requests are being sent back to the NetCache appliance. It may involve a complete reconfiguration of the switch, and it will depend on the features of its firmware or software. IP spoofing deployments will usually require two sets of filters – one for the outgoing client requests (based on the *destination* port – 80 for HTTP, 21 for FTP, 1755 for MMS, 554 for RTSP, etc.) to be redirected to the NetCache appliance, and another set of filters for the incoming server responses (based on the *source* port – again 80 for HTTP, 21 for FTP, 1755 for MMS, 554 for RTSP, etc.), to be redirected also to the NetCache appliance.

For example, if the first set is applied on the switch/router port that is connected to the default gateway (router)—sometimes called an *outbound* or *egress* filter—there is enormous potential for a routing loop to occur, because only a very complicated set of rules can distinguish an initial request coming from the clients from a spoofed request from a NetCache appliance. On the other hand, if that first set is applied on the switch port(s) that is/are connected to the clients—an inbound or ingress filter—that distinction is easy and clear.

For the second set, the same rule applies, although to the opposite ports—filtering for server responses should also be inbound, but on the default gateway port. If an *outbound* filter is applied to the client port(s), a routing loop will be generated with the response (it will be continuously re-sent to the NetCache appliance until it is dropped, and will never get to the clients).

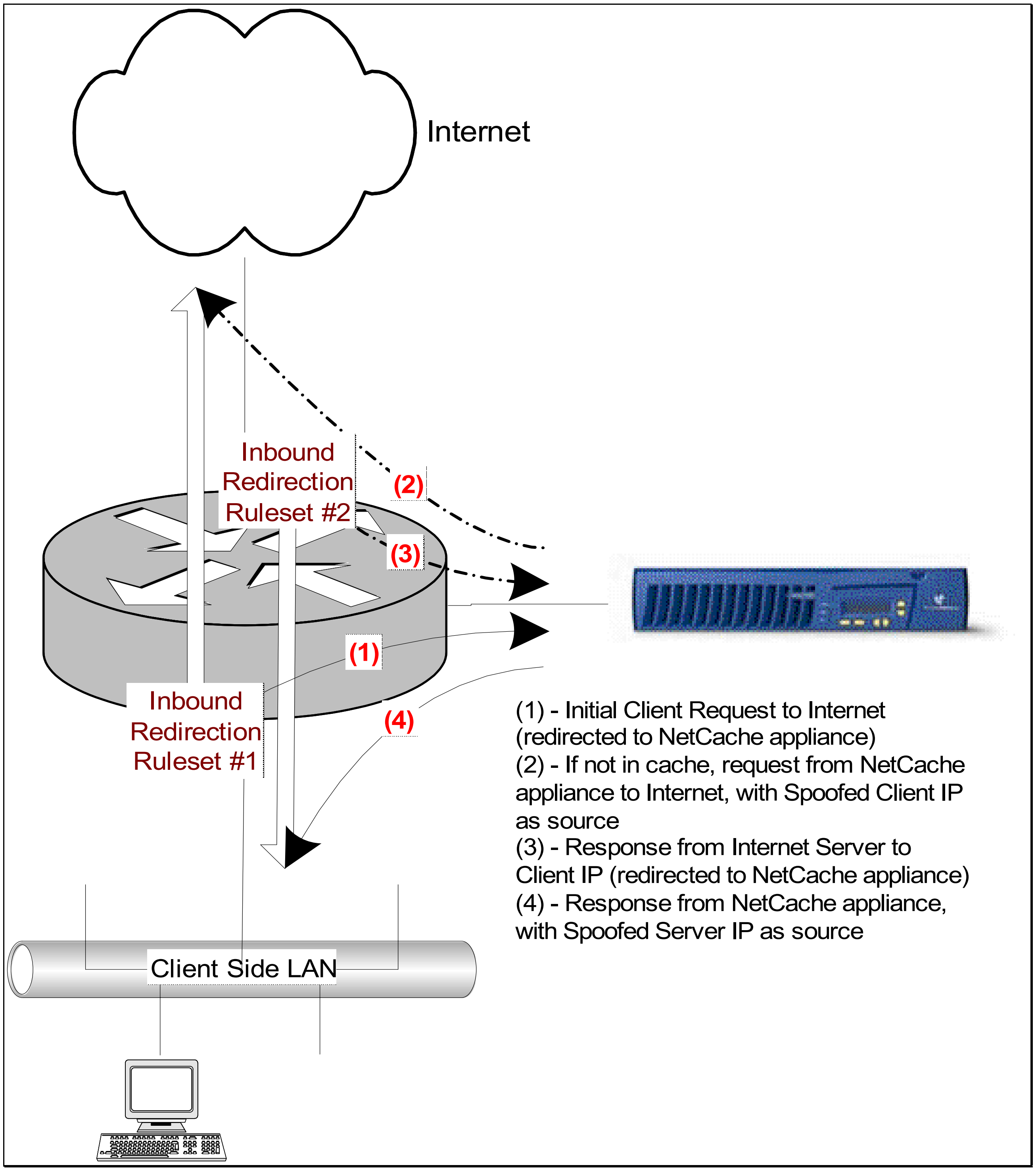
Internet
Inbound Redirection Ruleset #2
(2)
(3)
(1)
(4)
Inbound Redirection Ruleset #1
Client Side LAN
(1) - Initial Client Request to Internet (redirected to NetCache appliance)
(2) - If not in cache, request from NetCache appliance to Internet, with Spoofed Client IP as source
(3) - Response from Internet Server to Client IP (redirected to NetCache appliance)
(4) - Response from NetCache appliance, with Spoofed Server IP as source

Figure 1) IP Spoofing Designed in a Way That Prevents Routing Loops

This is a typical IP spoofing deployment, using inbound filters. Applying filters in the inbound interface (the interface connected to the source of packets that are to be redirected to the NetCache appliance) simplifies rules and avoids routing loops. Although a router symbol is used in the diagram, the same idea can be applied to a layer 4/7 switch. In either case, one interface/ port should be used for the client side LAN, another (separate) one for the NetCache appliance(s), and a third one for the Internet/WAN/next-hop side.

Unfortunately, not all layer 4/7 switches and WCCP equipment have the possibility of being configured like this (with inbound filters), and that is the main reason there is so much potential for routing loops with this type of deployment. However, for example, WCCP deployments that use Cisco IOS versions 12.1(3)T or later support inbound filtering and are generally the most reliable way of implementing transparency with IP spoofing exactly because of that.

Here are basic steps on how to configure a WCCP router in an IP spoofing deployment; in this example, interface *A* is the one connected to the NetCache appliance(s), *B* is the one connecting to the Internet/WAN/next-hop, and *C* is the one connecting to the client-side LAN:

1. Define two service groups (say 10 and 11) on the router globally.

```
#conf t
#ip wccp 10
#ip wccp 11
#end
```

2. Use one—10—for destination port 80 redirection (1) in figure 1 and another—11—for source port redirection (3) in figure 1. Essentially, the idea is to use 10 for inbound client traffic and 11 for inbound server traffic. So, it will be necessary to apply inbound redirection filters to corresponding interfaces:

```
#conf t
#interface C
(config-if)#ip wccp 10 redirect in
#exit
#interface B
(config-if)#ip wccp 11 redirect in
#end
```

3. Exclude redirection for traffic coming in from the interface connected to the NetCache appliance(s):

```
#conf t
#interface A
(config-if)#ip wccp redirect exclude in
#end
```

4. On the NetCache appliance, you need to make sure that characteristics of group 10 and 11 are as follows:

```
config.wccp.service_groups = \\
"client-spoof-traffic" 10 dynamic off password= tcp dst 80 dst-ip
"server-spoof-traffic" 11 dynamic off password= tcp src 80 src-ip
\\
```

If inbound filtering is not a possibility (e.g., with Catalyst 6500 switches), then it will be necessary to apply exceptions on the outbound filters, based on the source switch port (exclude from redirection, on both sets, if the frame is coming from the switch port where the NetCache is connected). In WCCP routers, this involves the usage of ACLs to build these exclusion filters.

4.2. Routing Issues in Transparent Deployments

In many cases, it may be a router—even the default gateway—to send a request (or response) back to the NetCache appliance either because, for some unknown reason, it thinks the NetCache appliance is the next hop in the route (which should never happen, since the NetCache appliance is not a router, nor does it have most of the router functionality) or, more commonly, because it sends a request back to a layer 4/7 switch as if it were a client request (even if the filters for requests are inbound in the client ports there may be situations where the router sends back a request through a client switch port, directly or indirectly, by way of other router or switch). In complex networks the combinations are endless.

The troubleshooting steps for this type of situation start in exactly the same way as the previous scenario. If the redirection rules are proven to be correct on the layer 4/7 switch, WCCP switch or WCCP router, then the next place to look is at the default gateway's routing table. A traceroute can be a very useful tool in determining problems, although it is usually not enough, because:

a) It doesn't use the same protocol and port as does, for example, HTTP (so it may be being treated differently by the network equipment)

b) The problem may be limited to certain source or destination subnets, which may be unknown

A good look at the NetCache appliance logs (looking for clues in the logged messages and in the x-note field of the Web access log), plus a thorough analysis of the switch/router filters and the routing tables in the default gateway will be the next steps. A good, detailed network diagram will be a definite requirement, so that for each filter/route request can be manually "traced," hop by hop. This may be a slow and repetitive process, so it is best avoided by prevention. When designing the deployment, bear in mind the potential for this type of problem and avoid it. For example, deployments that use WCCP routers that have a dedicated interface for clients, another for the NetCache appliance(s), and a third one to connect exclusively to next hop(s) (for *any* route) are usually the most reliable and easy to troubleshoot. If there are intranet servers, these should be also connected to the "next hop" interface in the router and completely isolated—meaning, in a different Ethernet segment/VLAN – from the client(s) and NetCache appliance(s).

An example of a network that, although complex, is designed to prevent routing loops and be easy to troubleshoot is represented in figure 2 (as an expansion of the concept presented in figure 1).

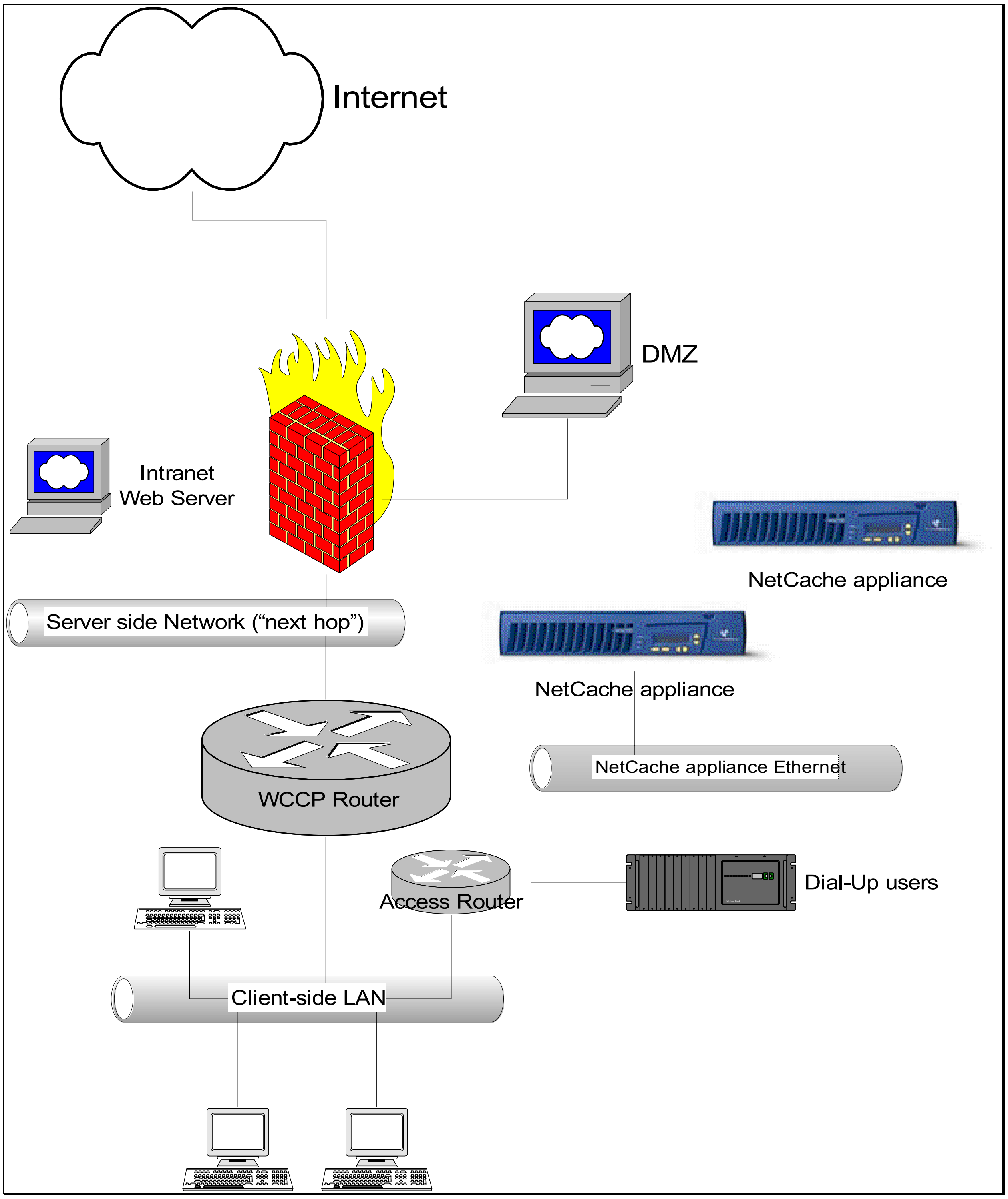

Figure 2) Network Designed in a Way That Prevents Routing Loops

This network implements all the recommendations on how to *prevent* routing loops. It allows for several possibilities, while remaining easy to follow and troubleshoot. It is essentially an expansion on the example presented in figure 1.

4.3. External Routing Loops That Are Detected When They Go through a NetCache Appliance

It can also happen that there is a routing loop in a completely different area of the Internetthat gets detected by a NetCache appliance when it finally goes through one.

If a request (or response) is sent when going through a large routing loop before getting to a NetCache appliance, this can trip triggers 3, 4, 5, and 6, even if the IPs in the *Via:* or *X-Forwarded-For:* headers do not belong to the NetCache appliance. The NetCache software will still report a loop warning irrespective of the headers containing its own name or IP address.

4.4. Looped Hierarchy

In a proxy/cache hierarchy where child/parent relationships exist, the children should be configured to send requests to their parents, but those parents MUST NOT be configured to send requests back to their children. If they are configured to send requests back to their children, a routing loop will, of course, occur.

If a layer 4/7 switch or WCCP router is involved in a hierarchy, great care needs to be taken to ensure the above—parents' requests coming back totheir children—doesn't happen due to the transparent redirection. Requests coming from the parents should be excluded from any redirection rules (as should any requests coming from a NetCache appliance, of course).

Neighbor hierarchies are even more dangerous, due to the nature of the protocol (ICP) that is used in such deployments, in particular if layer 4/7 switches or round-robin DNS are involved.

Network Appliance strongly advises against neighbor hierarchies, since they are slow and inefficient, and prone to problems such as these. If such a deployment is deemed necessary even in spite of that, extra care is required, especially with DNS entries and any transparent redirection rules. Ideally, such deployments should require manual redirection (proxy settings in the client browsers), and not transparent (layer 4/7, WCCP, or DNS-based) redirection, especially since all types of transparent redirection, if properly designed, make a neighbor hierarchy redundant and therefore unnecessary.

4.51 Requests to the Loopback IP

In nontransparent environments, a type of situation that will easily create a routing loop is if a buggy or malicious client tries to request from the NetCache appliance a URL with a server IP of 127.0.0.1 or a hostname that is resolved to 127.0.0.1. This address (127.0.0.1) is standardized as the IP address of the *loopback* interface, a virtual interface that all TCP/IP stacks provide, that points to the machine itself. This means that the NetCache appliance will try to get the object from itself, irrespective of the IP of any actual physical interface.

As an example, here are some Web access log entries of such requests:

```
bash$ grep 127.0.0.1 cache.access.log1

1049070383.11 1 10.10.10.10 TCP_MISS/500 406 GET http://127.0.0.1/myfile.html - - ""

1049075276.05 10 10.10.10.11 TCP_MISS/500 406 GET http://127.0.0.1/myotherfile.html - - ""

1049077365.56 1 10.10.20.30 TCP_MISS/500 406 GET http://loopback/myfinalfile.html - - ""
```

Clients `10.10.10.10`, `10.10.10.11` and `10.10.20.30` are accessing URLs that have `127.0.0.1` (or loopback, which is resolved to `127.0.0.1`) as destination server.

This type of request is usually impossible in transparent environments, since the client will just try to connect to the server directly, which means to itself in the case of a server IP or `127.0.0.1`.

You can add a `deny server-ip 127.0.0.1` entry in the global ACL section of *Setup* | *Access Control - Access Control Lists* to deny this kind of request, and solve this type of routing loop permanently.

4.6. Round-Robin DNS Configurations

A routing loop also occurs in certain conditions—transparent or nontransparent, in this case—when multiple NetCache appliances are associated with a DNS entry for a proxy. For example, if `proxy.mycorp.com` has IP addresses IP1, IP2, and IP3 for NetCache appliances NC1, NC2, and NC3 (respectively), and a request is made with `proxy.mycorp.com` as server (e.g., `GET http://proxy.mycorp.com/ HTTP/1.0`), a routing loop may occur.

The workaround is, similarly, to use an ACL entry to deny requests to the hostname that point to the NetCache appliance.

For the example above:

```
deny server-name proxy.mycorp.com.
```

4.7. Acceleration Request Gets Treated as Proxy Request

One particularly complex scenario that can create a routing loop involves a request that is intended to be a Web acceleration request to be treated as a proxy request. This can occur in very particular load-balanced Web acceleration situations.

The first symptom will be that requests from a NetCache appliance will start showing up in the logs of other NetCache appliances, when they should go to the accelerated Web server. For example, here is the log of a NetCache appliance called `netcache05`:

```
10.15.16.56 - - [29/Oct/2002:10:37:51 -0800] HEAD http://192.168.12.34/kicchomu? "" "" 500 -
10.15.16.56 - - [29/Oct/2002:10:37:51 -0800] HEAD http://192.168.12.34/kicchomu? "" "" 500 -
10.15.16.60 - - [29/Oct/2002:10:37:51 -0800] HEAD http://192.168.12.34/kicchomu? "" "" 500 -
10.15.16.56 - - [29/Oct/2002:10:37:51 -0800] HEAD http://192.168.12.34/kicchomu? "" "" 500 -
10.15.16.56 - - [29/Oct/2002:10:37:51 -0800] HEAD http://192.168.12.34/kicchomu? "" "" 500 -
10.15.16.56 - - [29/Oct/2002:10:37:51 -0800] HEAD http://192.168.12.34/kicchomu? "" "" 500 -
10.15.16.60 - - [29/Oct/2002:10:37:51 -0800] HEAD http://192.168.12.34/kicchomu? "" "" 504 -
10.15.16.60 - - [29/Oct/2002:10:37:51 -0800] HEAD http://192.168.12.34/kicchomu? "" "" 500 -
10.15.16.56 - - [29/Oct/2002:10:37:51 -0800] HEAD http://192.168.12.34/kicchomu? "" "" 500 -
```

Let’s suppose there are five NetCache appliances involved, load-balanced by a layer 4/7 switch, each with the following default/main IP addresses in their e5 interface:

```
10.15.16.56     netcache01-e5
10.15.16.57     netcache02-e5
10.15.16.58     netcache03-e5
```

```
10.15.16.59     netcache04-e5
10.15.16.60     netcache05-e5
```

So in the example log, requests coming from `netcache05` (itself) and `netcache01` are ending up in `netcache01`.

The server IP—`192.168.12.34`—is the front-end IP of an accelerated Web server:

```
config.http.acceleration.rules = \\
192.168.12.77 80 * 10.13.10.131 80 *
192.168.12.32 80 * 10.13.10.212 80 *
192.168.12.33 80 * 10.13.10.212 82 *
192.168.12.34 80 homepage.netapp.com 10.13.10.212 81 *
192.168.12.35 80 * 10.13.10.212 87 *
\\
```

In the messages log, the following will be observed if client privacy is not enabled:

```
Wed Oct 23 19:03:35 PDT [sthread_loop:warning]: Potential Routing loop
detected in HTTP Via header: 1.1 netcache01 (NetCache NetApp/5.2.1R2D2), 1.1
netcache03 (NetCache NetApp/5.2.1R2D2), 1.1 netcache01 (NetCache
NetApp/5.2.1R2D2), 1.1 netcache05 (NetCache NetApp/5.2.1R2D2), 1.1
netcache01 (NetCache NetApp/5.2.1R2D2), 1.1 netcache04 (NetCache
NetApp/5.2.1R2D2), 1.1 netcache03 (NetCache NetApp/5.2.1R2D2), 1.1
netcache01 (NetCache NetApp/5.2.1R2D2), 1.1 netcache05 (NetCache
NetApp/5.2.1R2D2), 1.1 netcache02
```

This is the full interface configuration of `netcache01`:

```
config.system.interface.e5 = \\
# default incoming and outgoing gateway interface (via the L4/7 switch)
up 10.15.16.56 255.255.254.0 10.15.17.255 1500 n/a send
\\

config.system.interface.e2 = \\
# outgoing requests to the accelerated Web Server
up 10.13.10.56 255.255.254.0 10.13.11.255 1500 100tx-fd n/a
\\

config.system.interface.e0 = \\
# admin interface
up 10.8.5.140 255.255.0.0 10.8.255.255 1500 100tx-fd n/a
\\

config.system.gateways.ip = 10.15.16.1
```

In an initial observation, one may feel that the acceleration rule that the request shown in the extract of the Web access log would hit is the fourth:

```
192.168.12.34 80 homepage.netapp.com 10.13.10.212 81 *
```

The problem is that the rule specifies that the accelerator host name (homepage.netapp.com) is required, but the request is for the IP address. So the NetCache appliance is treating this request as it would forward proxy requests, not accelerator requests, because it doesn't match any of the rules 100%.

Since the destination IP address of the request is not in the subnet of any of the interfaces of the NetCache appliance, it will be sent via e5 to the default gateway though the layer 4/7 switch, which will, unfortunately, send it back to the NetCache appliance, since it believes it's an external client request.

The request will therefore get back to the NetCache appliance, which again won't match any rule, and the loop will repeat itself until the NetCache appliance detects it and drops the request altogether.

There are at least three ways to prevent this routing loop:

1. Configure the layer 4/7 switch so it does not redirect back to the NetCache appliance requests coming from it, effectively preventing a loop from occurring.

2. Assuming the NetCache appliance is *only* used as a Web accelerator, add this entry in the HTTP ACL (not the acceleration ACL):

```
deny not accel
```

This will tell the NetCache software to deny any request that doesn't match an acceleration rule (like the request causing the loop).

There is one thing to bear in mind with this option: if your load balancer is doing HTTP health checks of the NetCache appliance or the backend servers, it is necessary to allow these as proxy requests (because they are not matched by acceleration rules). If the load balancer's IP is `10.15.16.2` that would mean that you'd need to add two entries in the HTTP ACLs, instead of just one:

```
allow client-ip 10.15.16.2
deny not accel
```

3. Change the existing acceleration rule(s) so they won't require the host name in the request. In the example above, that would mean changing the fourth rule to:

```
192.168.12.34 80 * 10.13.10.212 81 *
```

4.8. CONNECT Request Looping Back to NetCache Appliance

This type of routing loop is actually detected and logged differently. This is the type of entry in the messages log that you'll see:

```
[sthread_loop:warning]: Client IP: 10.64.23.50 trying to issue a CONNECT
looping back to Appliance
```

What triggers this is a `CONNECT` (HTTP-based tunneling) request that was sent to the NetCache appliance with a destination with a server name equal to the one of the NetCache device or a destination IP of one of the IP addresses of the NetCache appliance. This could be the result of a client software bug or someone attacking the NetCache appliance.

Another possibility is that an end user is connecting to a malicious Web site where there is a link that refers back to the name or IP address of the NetCache appliance. This can be there statically or be added dynamically (asin an `.asp` or `.php` page), obtained from the source IP of the request (which would be the IP of the NetCache appliance unless IP spoofing is being used).

There are several ways of fixing this:

1. In a transparent environment, if the sites that are causing this (links pointing back to the NetCache appliance on non-HTTP ports) are known, just go to *Setup* | *Transparency - Request Forwarding* in the NetCache manager interface, enable IP forwarding, and add those sites in the server IP list for IP forwarding.
2. In a nontransparent environment, that bypass (for the "guilty" servers) can be accomplished by adding the server names to the exception list in your Web browser's proxy settings, or, if you're using a `proxy.pac` file, by adding rules to that file so it won't use a proxy for those servers.
3. Also, in a transparent environment, another possibility (if the server or, in this case, client IPs causing the problem are known) is to add a router/switch ACL for such sites and/or clients.
4. Finally, if the NetCache appliance has `10.64.23.50` and `10.64.23.51` as IP addresses (for example), entries in the global ACLs section such as

   ```
   deny server-ip 10.64.23.50
   deny server-ip 10.64.23.51
   deny server-ip 127.0.0.1
   ```

   … will deny the request and hence stop the routing loop from happening.

5. Findings – Frequency of Occurrence of Routing Loop Situations

It has been the experience of the customer satisfaction department of Network Appliance that routing loop cases are not particularly frequent in general; they represent less than 0.5% of the support cases. However, they tend to reoccur on the same deployments, since, as shown in this document, the style of the design of a network can play a huge role in making it routing loop-prone or, on the contrary, routing loop-resilient.

6. Conclusions

Unfortunately, since they are caused by design flaws of networks, fixing routing loops is usually extremely time-consuming, especially in remote troubleshooting scenarios. Fixing them quickly and permanently requires deep, detailed knowledge of the networks' designs, and usually a physical presence in the network's premises by a skilled network engineer, capable of finding the source of the problem and redesigning the network if necessary, cannot be replaced by a remote one.

This document contains most of the scenarios Network Appliance customer satisfaction has encountered, therefore helping the local network engineer’s task of troubleshooting and redesigning.

7. Acknowledgements

This technical report could not have been written without the invaluable help of many people, namely:

- Fabricio Santos, NetCache technical advisor, who drove the proposal for this document, contributed with a list of cases where routing loops were observed, and provided feedback on the draft document
- Robert Richardson and Kentaro Ebisawa, NetCache escalation engineers, who provided valuable feedback
- NetCache technical support engineers, who over the years carefully documented the cases, providing us with precious scenario information