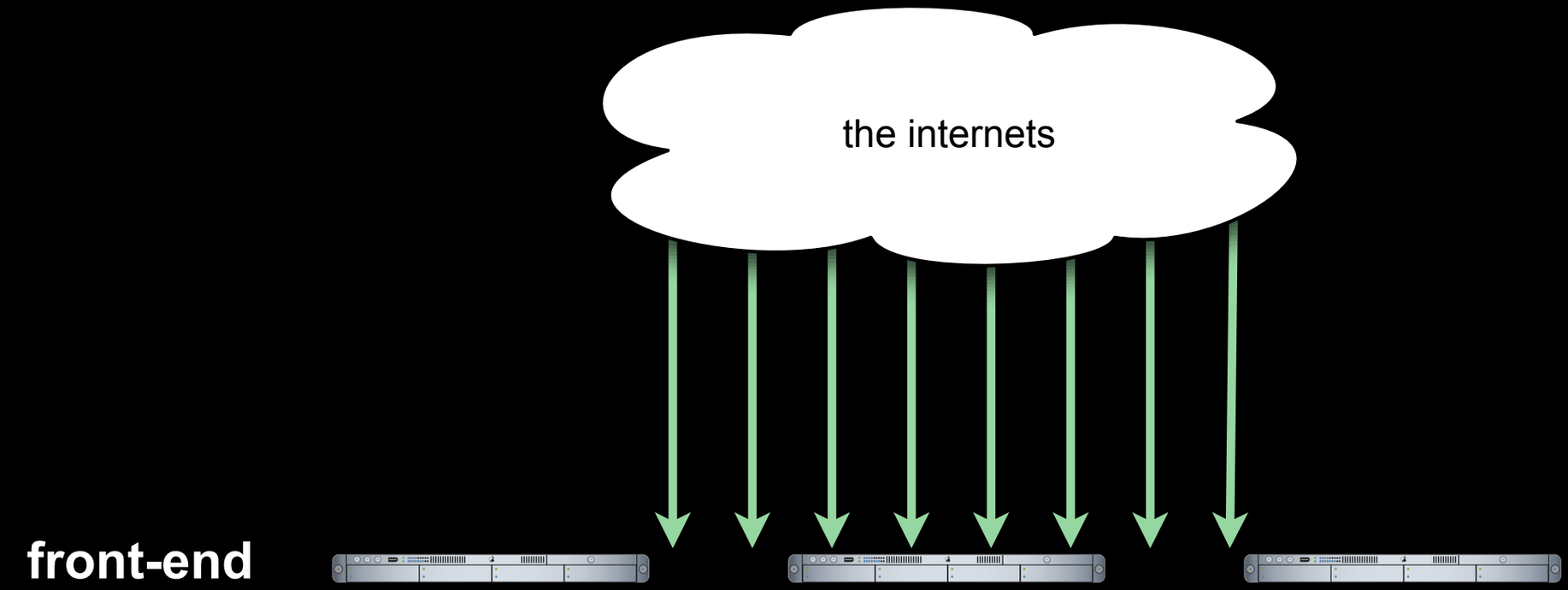
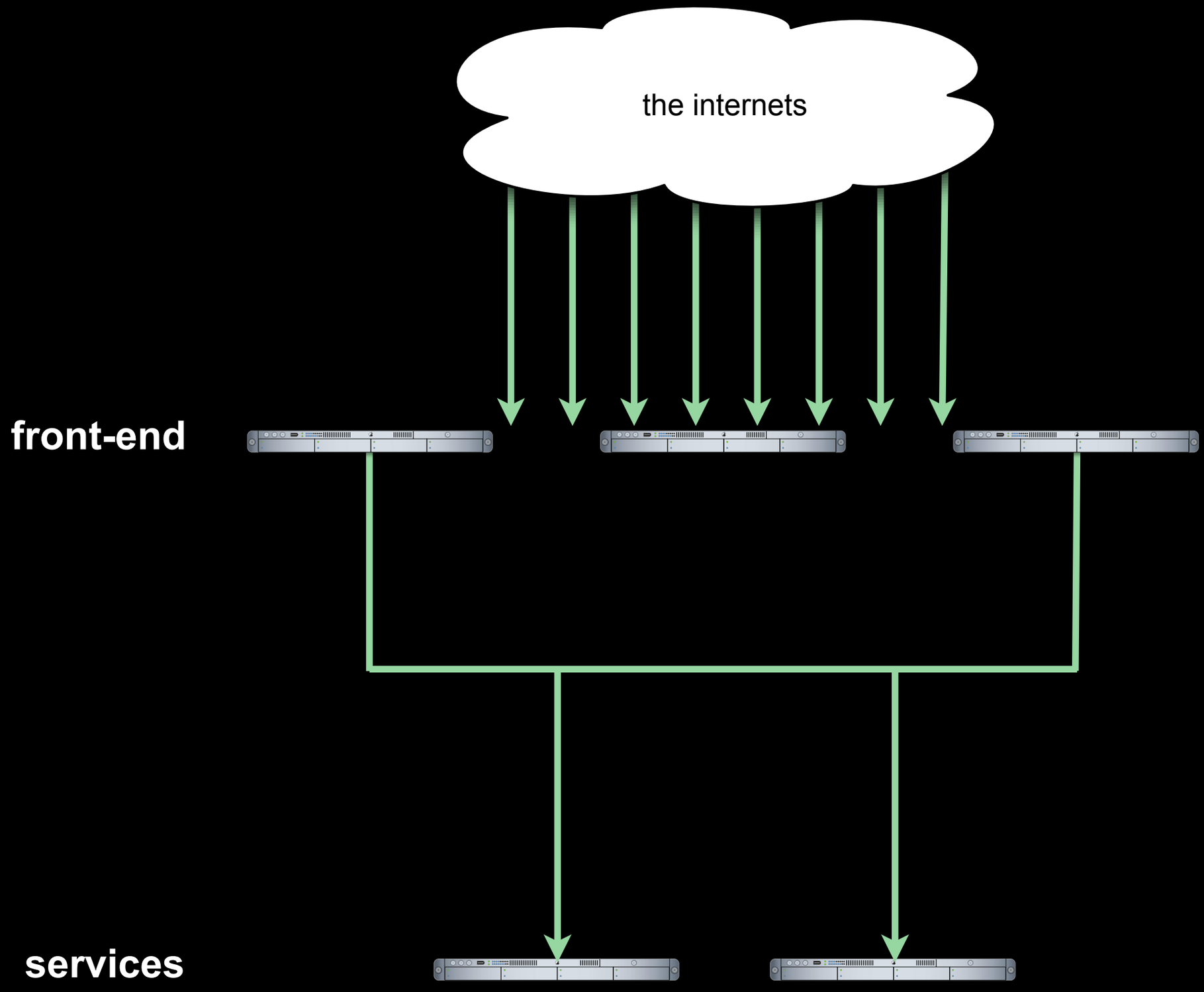
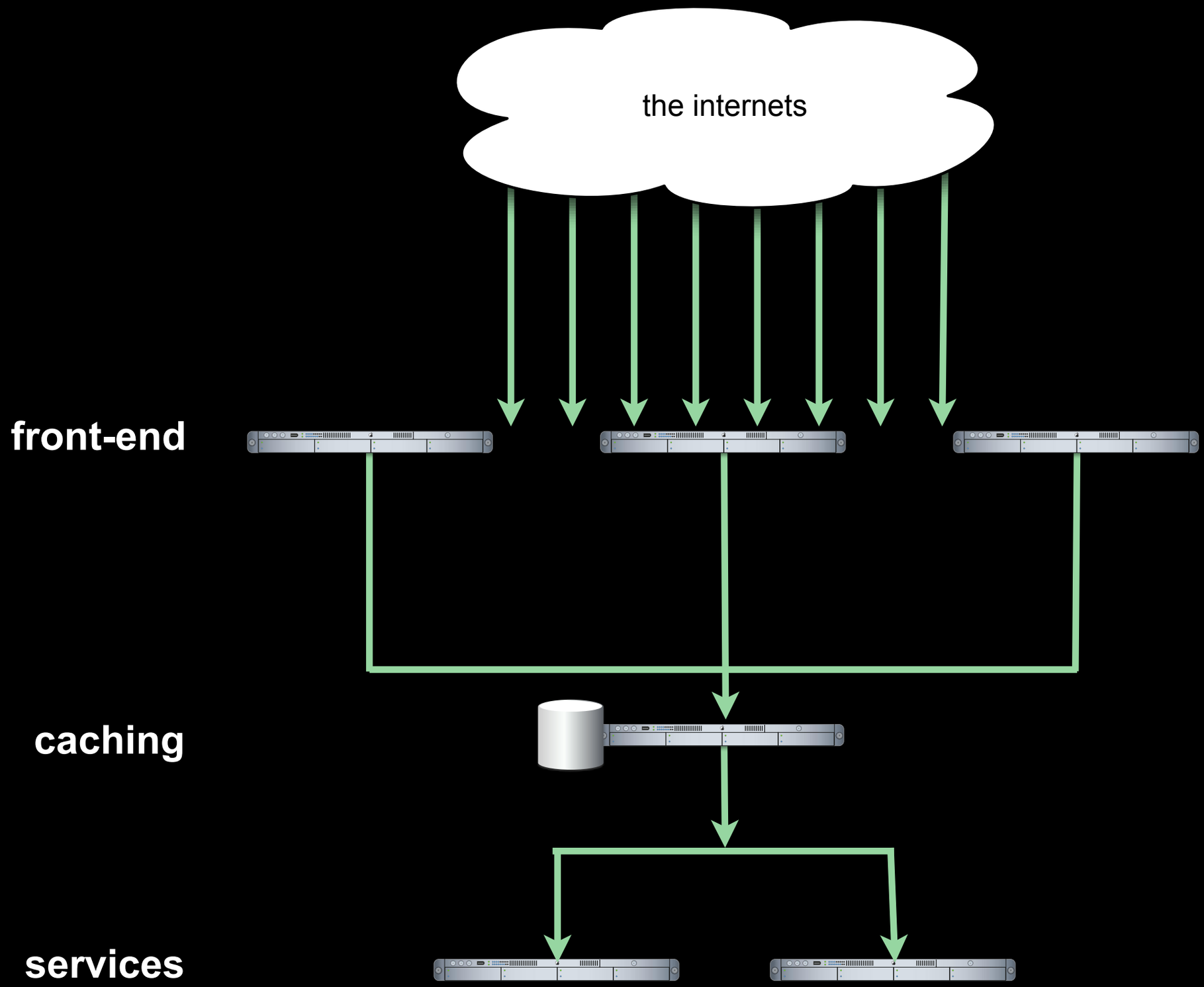
Stupid Web Caching Tricks

Mark Nottingham / mnot@yahoo-inc.com / mnot@mnot.net / @mnot

foo.yahoo.com


the internets
front-end


the internets
front-end
services


the internets
front-end
caching
services

Simple, right?

Well, let’s bring it into rotation...

1 2 3 4 5 6

```
1276007531.061    205 192.168.1.16 TCP_MISS/200 9286 GET /details?ticker=ABC
1276007531.062    205 192.168.1.17 TCP_MISS/200 9287 GET /details?ticker=ABC
1276007531.064    218 192.168.1.16 TCP_MISS/200 9285 GET /details?ticker=ABC
1276007531.065    198 192.168.1.17 TCP_MISS/200 9286 GET /details?ticker=ABC
1276007531.065    215 192.168.1.15 TCP_MISS/200 9286 GET /details?ticker=ABC
1276007531.068    205 192.168.1.15 TCP_MISS/200 9288 GET /details?ticker=ABC
1276007531.072      1 192.168.1.17 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007531.254    398 192.168.1.15 TCP_MISS/200 9288 GET /details?ticker=ABC
1276007531.261    408 192.168.1.15 TCP_MISS/200 9287 GET /details?ticker=ABC
1276007531.289    429 192.168.1.17 TCP_MISS/200 9286 GET /details?ticker=ABC
1276007531.922    852 192.168.1.15 TCP_MISS/504 282 GET /details?ticker=ABC
1276007532.005      0 192.168.1.15 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007532.044    987 192.168.1.16 TCP_MISS/504 283 GET /details?ticker=ABC
1276007532.045      2 192.168.1.16 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007532.068   1001 192.168.1.17 TCP_MISS/000 0 GET /details?ticker=ABC
1276007532.072    998 192.168.1.16 TCP_MISS/504 278 GET /details?ticker=ABC
1276007591.062  60001 192.168.1.16 TCP_MISS/000 0 GET /details?ticker=ABC
```

Oops.


```
1276007531.068    205 192.168.1.16 TCP_MISS/200 9286 GET /details?ticker=ABC
1276007531.068    205 192.168.1.17 TCP_MISS/200 9286 GET /details?ticker=ABC
1276007531.068    205 192.168.1.17 TCP_MISS/200 9286 GET /details?ticker=ABC
1276007531.068    205 192.168.1.15 TCP_MISS/200 9286 GET /details?ticker=ABC
1276007531.068    205 192.168.1.16 TCP_MISS/200 9286 GET /details?ticker=ABC
1276007531.068    205 192.168.1.15 TCP_MISS/200 9286 GET /details?ticker=ABC
1276007531.072      1 192.168.1.17 TCP_HIT/200 9287 GET /details?ticker=ABC
1276007531.072      0 192.168.1.15 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007531.073      1 192.168.1.16 TCP_HIT/200 9285 GET /details?ticker=ABC
1276007531.073      0 192.168.1.15 TCP_HIT/200 9285 GET /details?ticker=ABC
1276007531.074      0 192.168.1.17 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007531.076      1 192.168.1.16 TCP_HIT/200 9285 GET /details?ticker=ABC
1276007531.076      0 192.168.1.17 TCP_HIT/200 9287 GET /details?ticker=ABC
1276007531.077      0 192.168.1.15 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007531.078      1 192.168.1.15 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007531.079      1 192.168.1.16 TCP_HIT/200 9285 GET /details?ticker=ABC
```

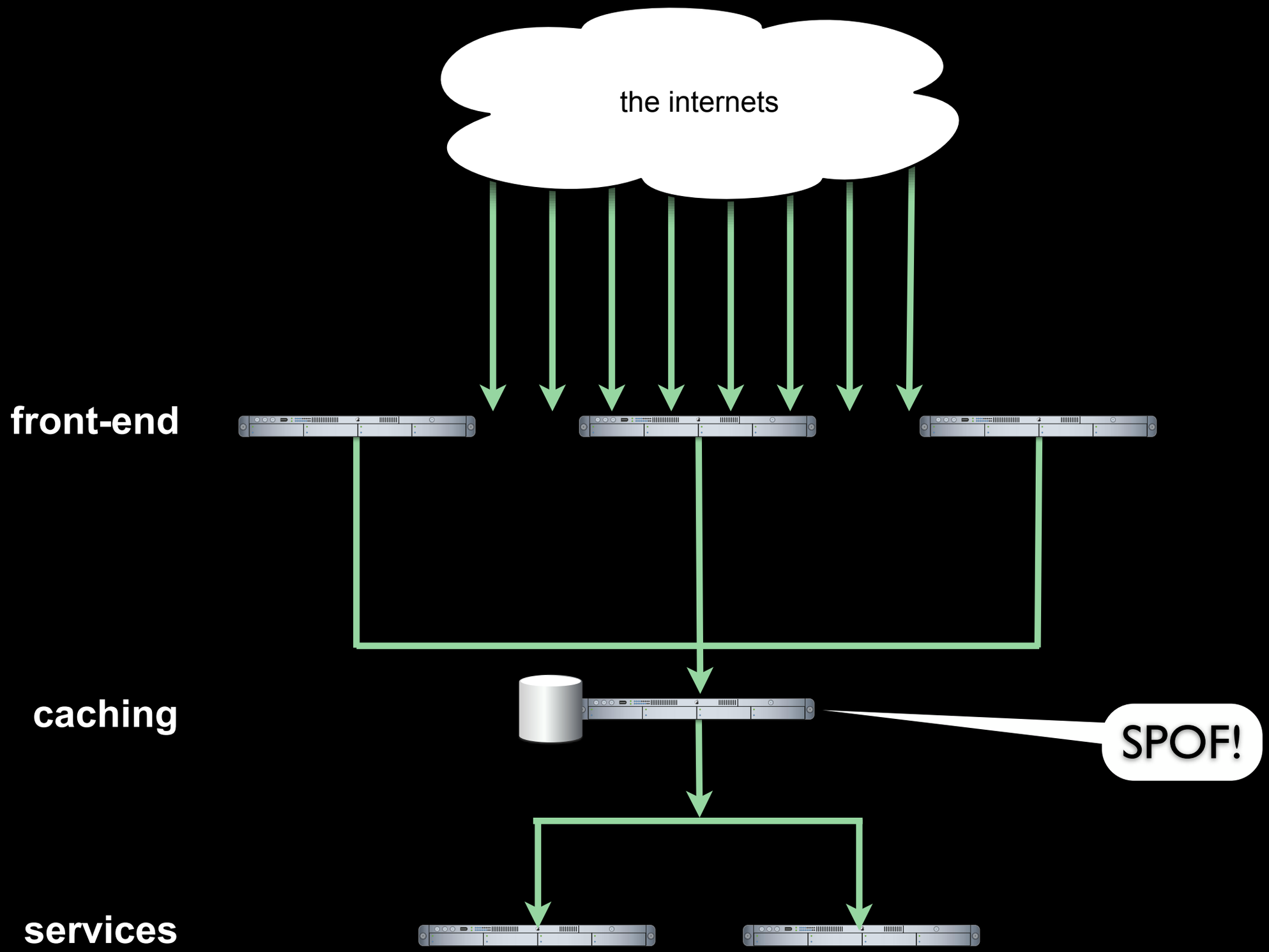
Collapsed Forwarding

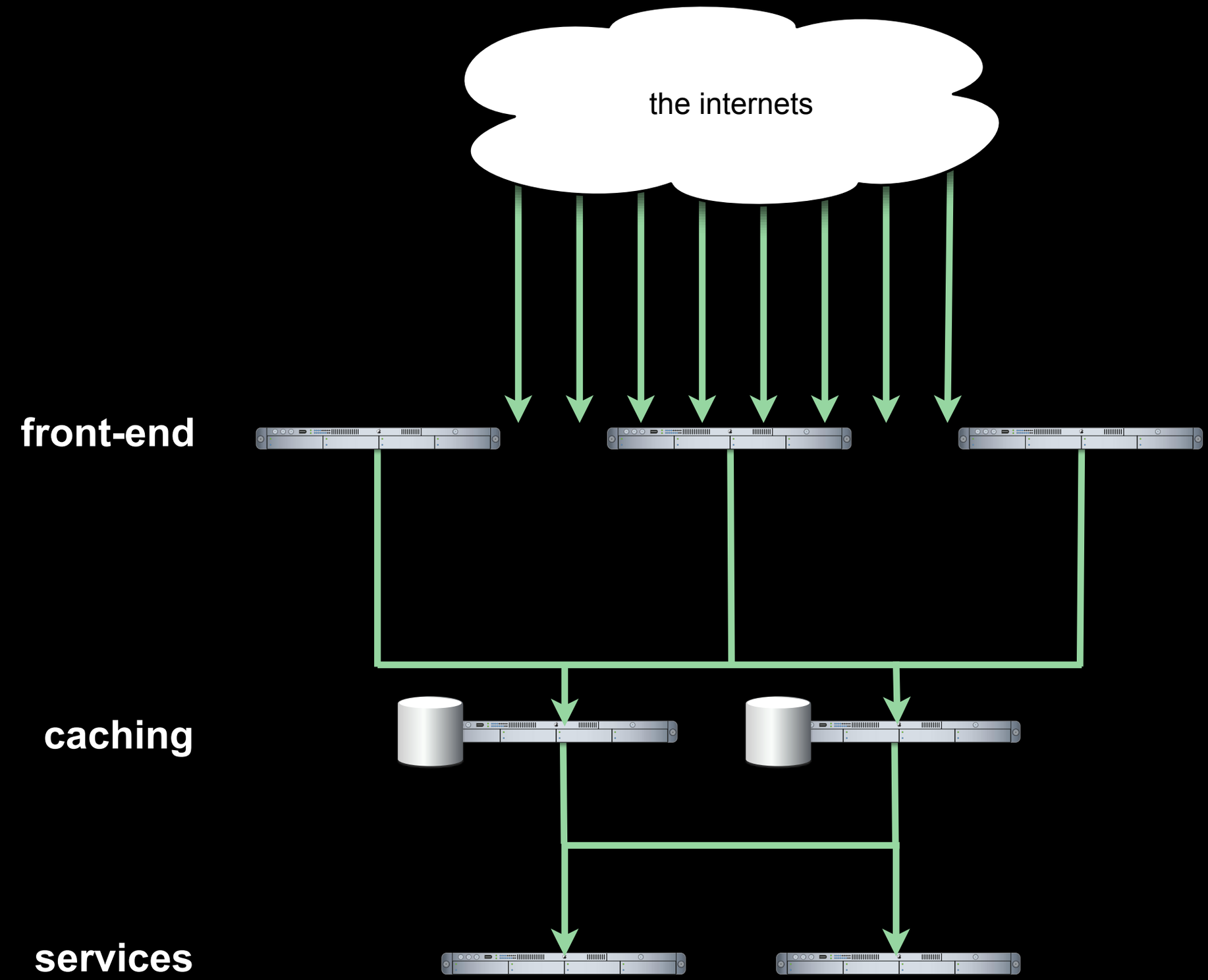
in squid2:

```
collapsed_forwarding on
```

in squid2.HEAD:

```
collapsed_forwarding_timeout
```

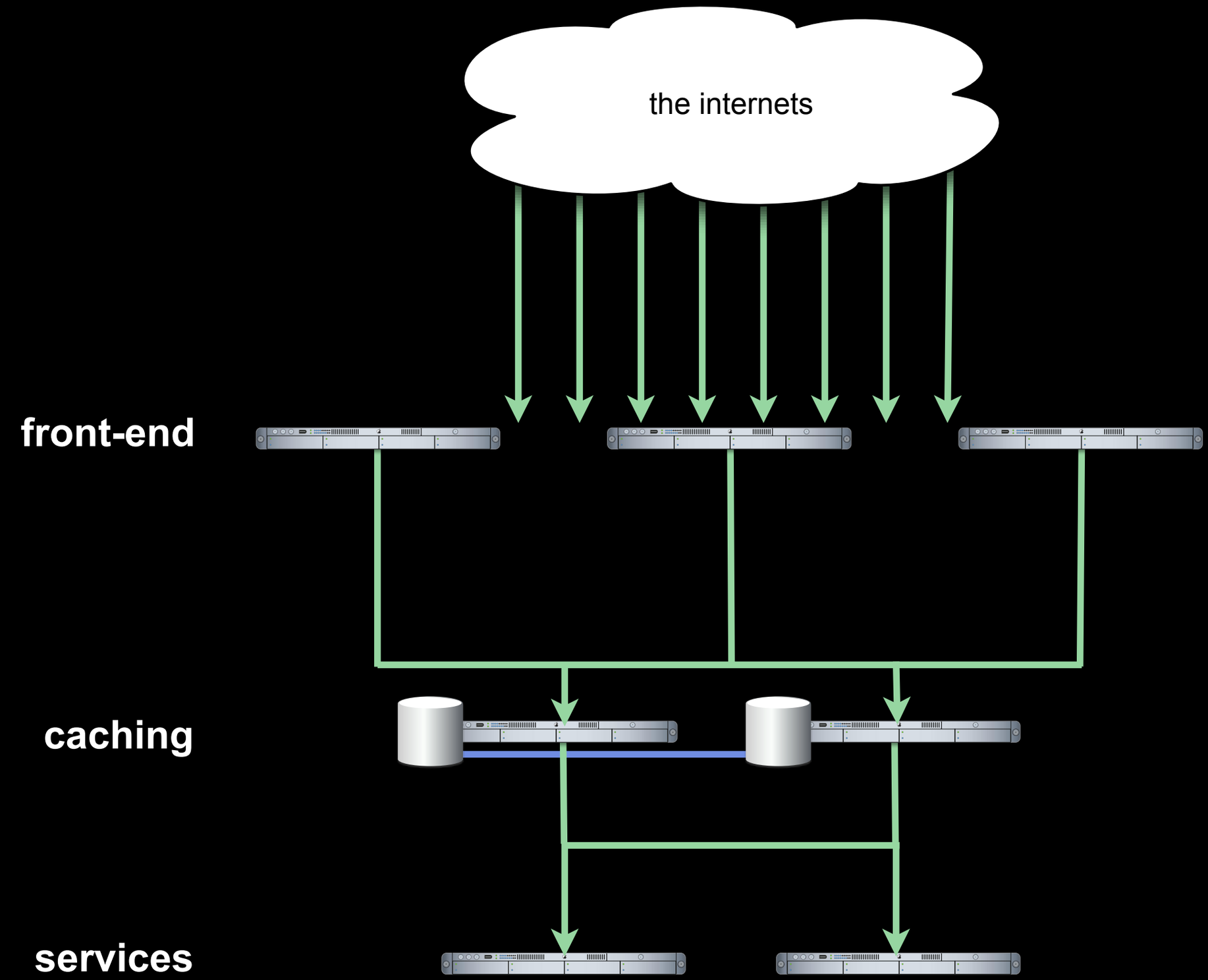

the internets
front-end
caching
services
SPOF!


the internets
front-end
caching
services

\+ good business continuity
more flexible

\- worse hit rate
high load when new caches come online
caches can come out of sync

answer: Cache Peering

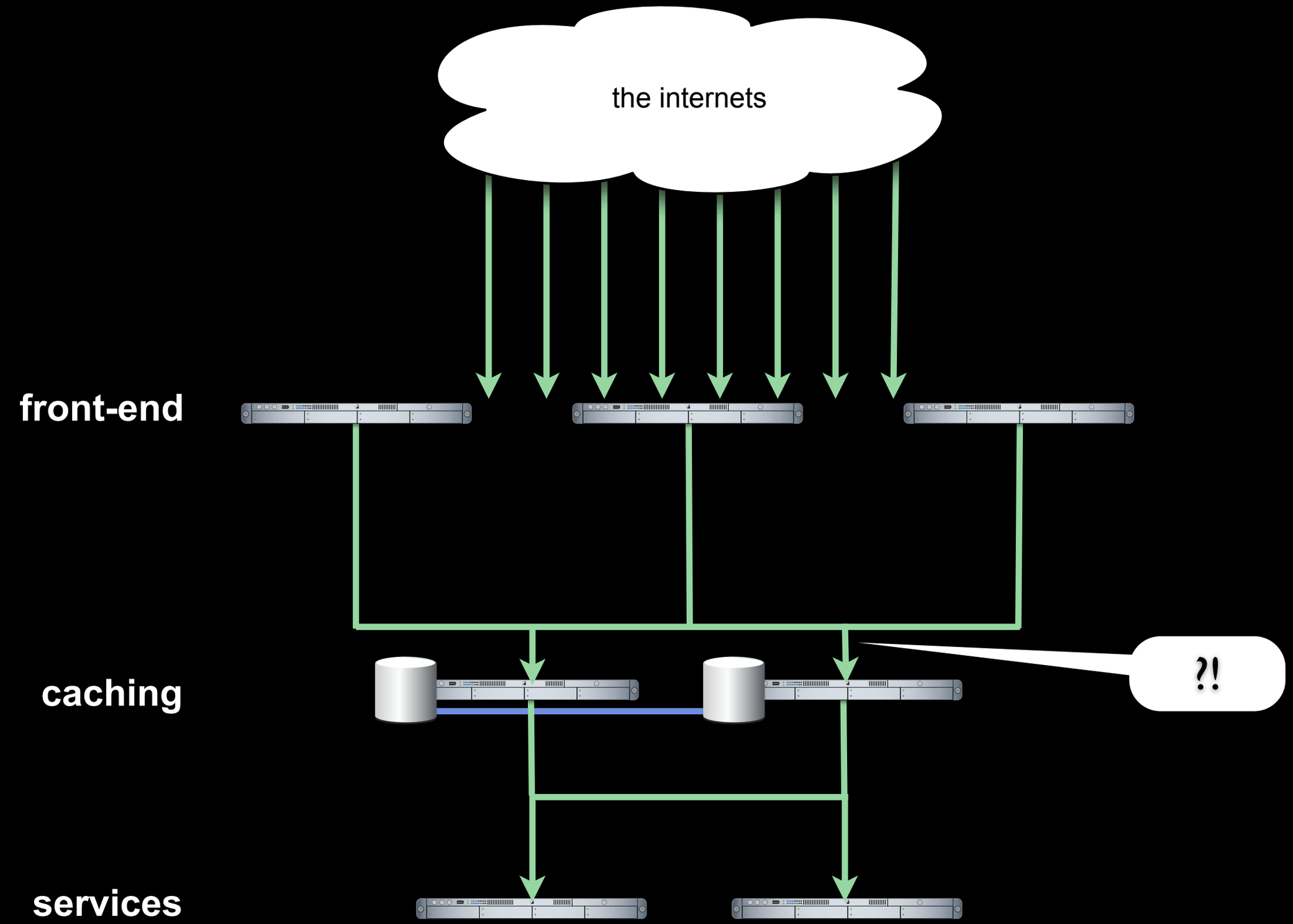

the internets
front-end
caching
services

RFC 2186 - Internet Cache Protocol

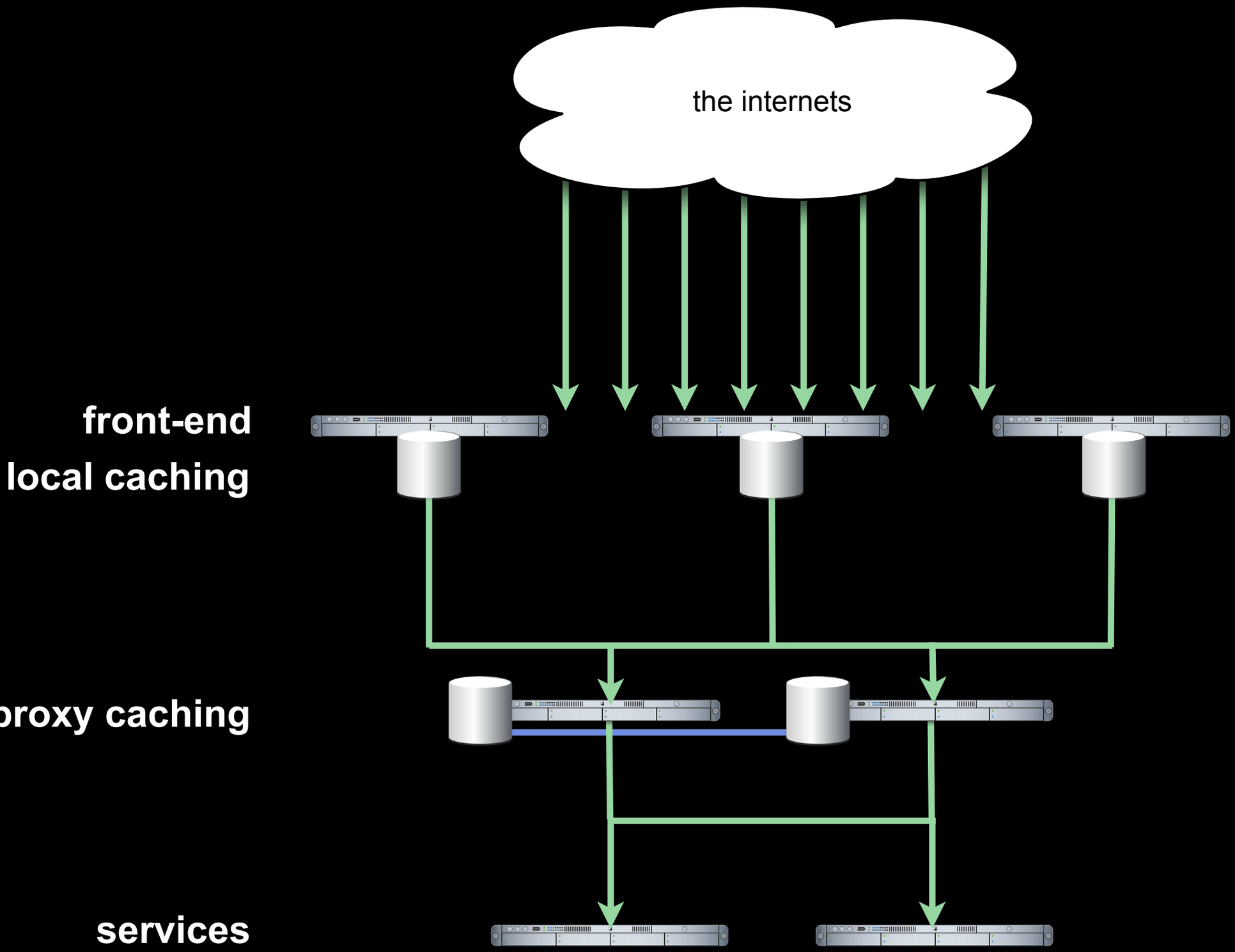
UDP-based
Just the URI
Query only
in Squid / Traffic Server

RFC 2756 - Hyper Text Caching Protocol

UDP-based (option for TCP in spec)
Includes URI + Headers
Query, CLR operations
in Squid


the internets
front-end
caching
services
?!

24 front-end servers x
24 Apache children x
5 pages / second x
8 service requests / page x
10k / service response /
2 cache servers =
11,520 req/sec
900 Mbits/sec
/cache server

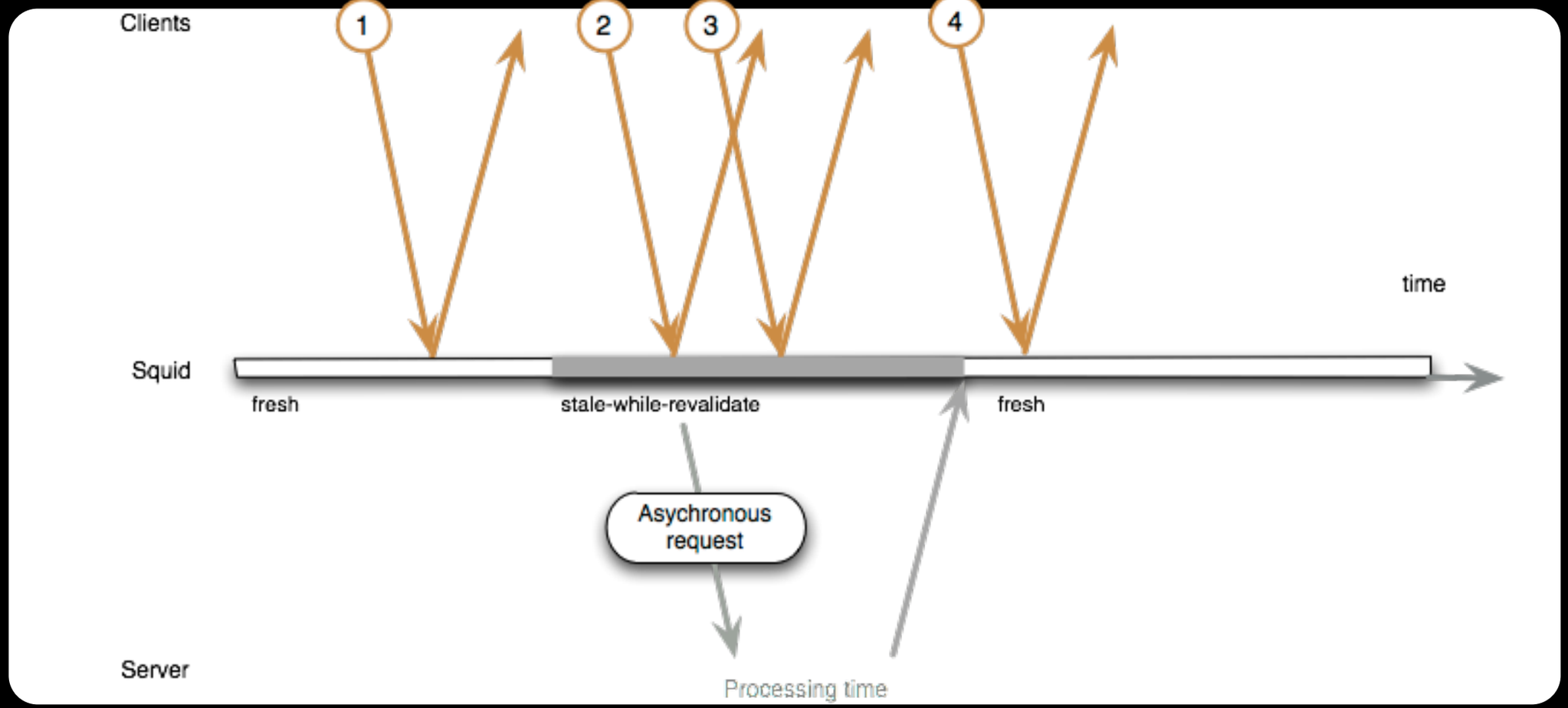

the internets
front-end
local caching
proxy caching
services

Hierarchy

1 2 3 4 5 6

```
1276007530.037      0 192.168.1.17 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007530.057      1 192.168.1.16 TCP_HIT/200 9285 GET /details?ticker=ABC
1276007530.083      0 192.168.1.17 TCP_HIT/200 9287 GET /details?ticker=ABC
1276007530.119      0 192.168.1.15 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007530.141      1 192.168.1.15 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007530.179      0 192.168.1.16 TCP_HIT/200 9285 GET /details?ticker=ABC
1276007530.397    205 192.168.1.15 TCP_REFRESH_MISS/200 9285 GET /details?...
1276007530.401      1 192.168.1.17 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007530.414    201 192.168.1.17 TCP_REFRESH_MISS/200 9285 GET /details?...
1276007530.418      1 192.168.1.15 TCP_HIT/200 9285 GET /details?ticker=ABC
1276007530.434      0 192.168.1.16 TCP_HIT/200 9287 GET /details?ticker=ABC
1276007530.442    198 192.168.1.17 TCP_REFRESH_MISS/200 9285 GET /details?...
1276007530.372      0 192.168.1.15 TCP_HIT/200 9285 GET /details?ticker=ABC
1276007530.494    201 192.168.1.16 TCP_REFRESH_MISS/200 9285 GET /details?...
1276007530.525      1 192.168.1.17 TCP_HIT/200 9284 GET /details?ticker=ABC
1276007530.548    201 192.168.1.17 TCP_REFRESH_MISS/200 9285 GET /details?...
1276007530.563      1 192.168.1.15 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007530.594      0 192.168.1.16 TCP_HIT/200 9285 GET /details?ticker=ABC
```

Content Becomes Stale




```
Cache-Control: stale-while-revalidate=30
```

RFC 5861

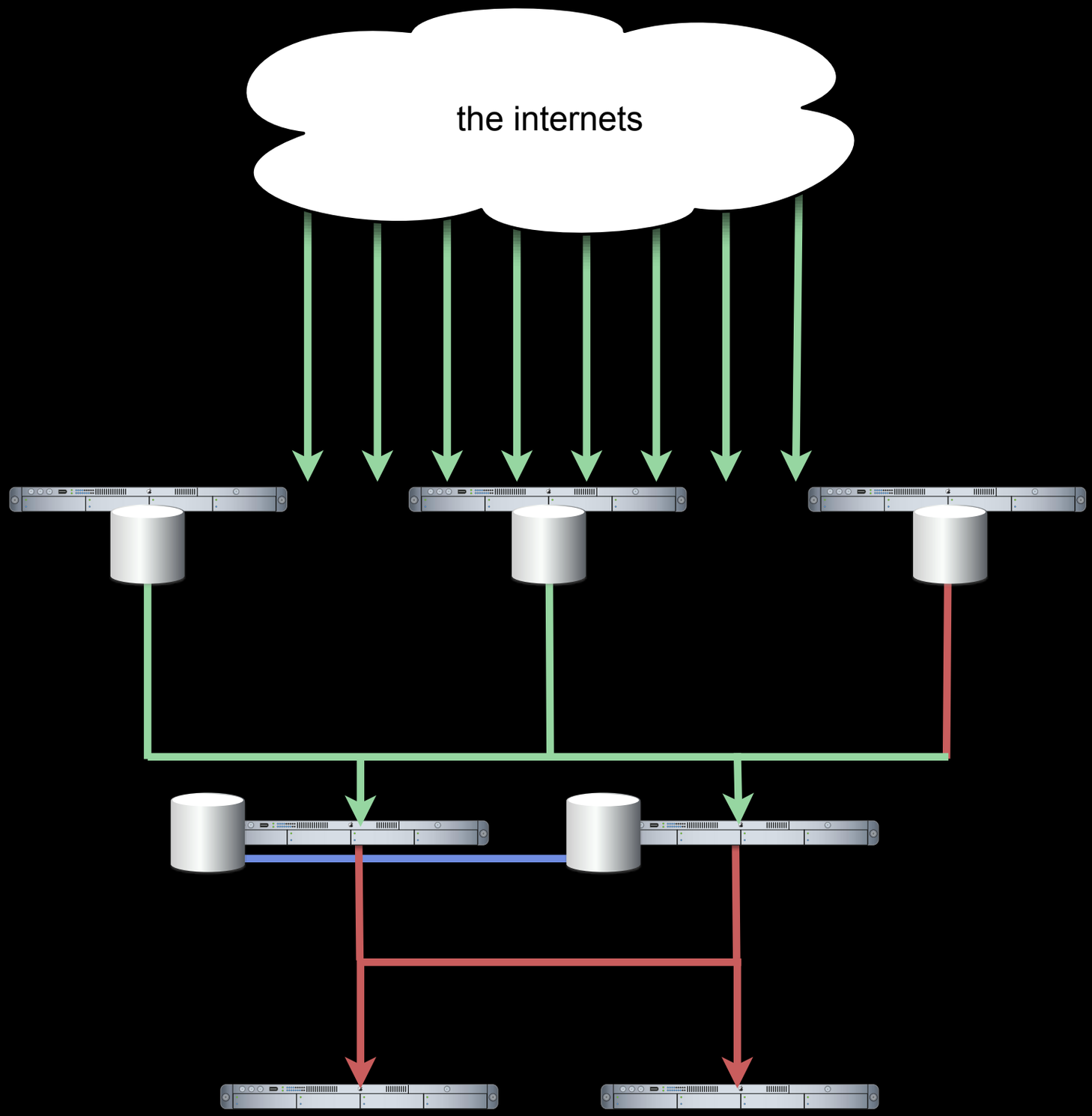
implemented	Squid 2.7
coming soon	Squid 3.2
	Apache Traffic Server

1 2 3 4 5 6

```
1276007530.037      0 192.168.1.17 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007530.057      1 192.168.1.16 TCP_HIT/200 9285 GET /details?ticker=ABC
1276007530.083      0 192.168.1.17 TCP_HIT/200 9287 GET /details?ticker=ABC
1276007530.119      0 192.168.1.15 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007530.141      1 192.168.1.15 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007530.179      0 192.168.1.16 TCP_HIT/200 9285 GET /details?ticker=ABC
1276007530.192      0 192.168.1.15 TCP_STALE_HIT/200 9285 GET /details?...
1276007530.213      1 192.168.1.17 TCP_STALE_HIT/200 9286 GET /details?...
1276007530.243      0 192.168.1.17 TCP_STALE_HIT/200 9285 GET /details?...
1276007530.294      0 192.168.1.16 TCP_STALE_HIT/200 9287 GET /details?...
1276007530.347      0 192.168.1.17 TCP_STALE_HIT/200 9285 GET /details?...
1276007530.384    219 0.0.0.0 TCP_ASYNC_MISS/200 9285 GET /details?...
1276007530.401      1 192.168.1.17 TCP_HIT/200 9284 GET /details?ticker=ABC
1276007530.418      1 192.168.1.15 TCP_HIT/200 9286 GET /details?ticker=ABC
1276007530.434      0 192.168.1.16 TCP_HIT/200 9285 GET /details?ticker=ABC
```

stale-while-revalidate

front-end
local caching



proxy caching

services

```
Cache-Control: stale-if-error=3600
```

RFC 5861

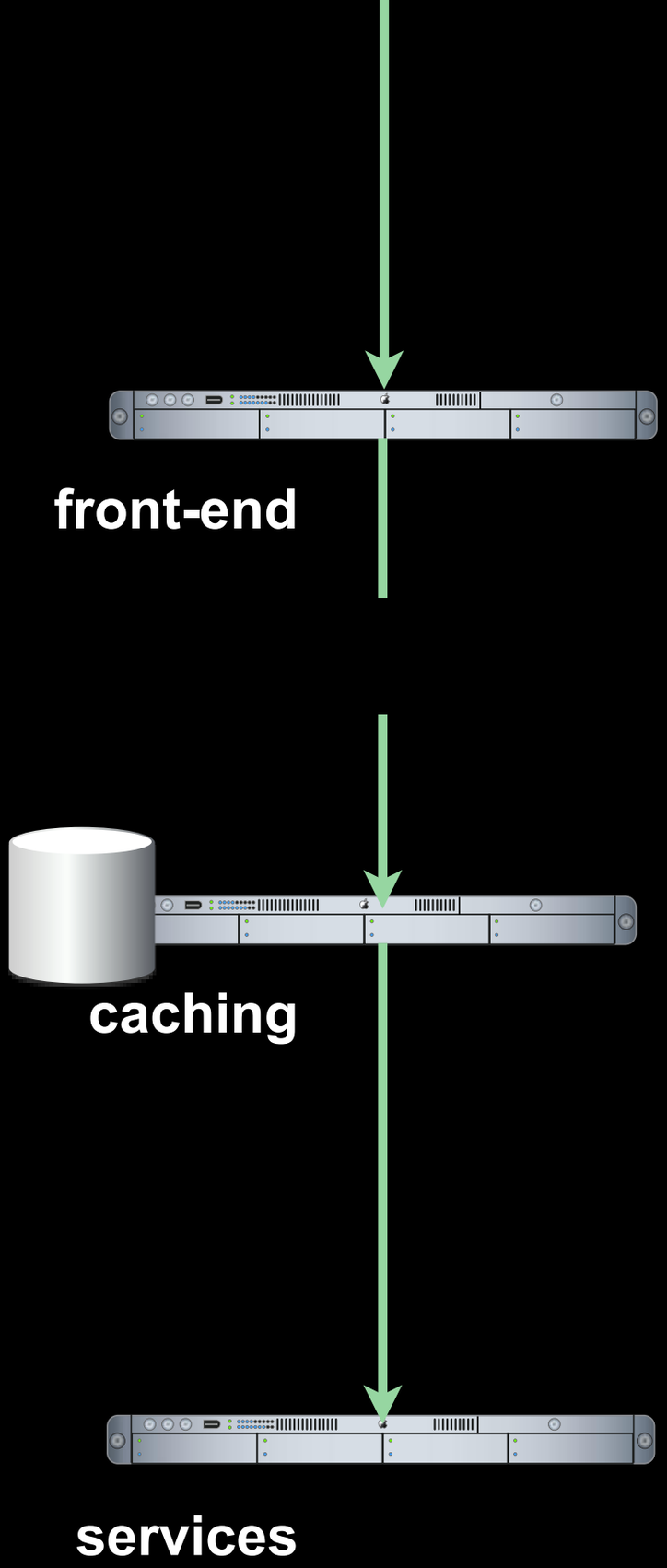
implemented — Squid 2.7

coming soon — Squid 3.2

Apache Traffic Server

Dealing with

Aborted Requests



front-end timeout: 500ms

dropped client connection

slow service = no cached response

not cached = always slow

Squid `quick_abort`

Apache Traffic Server `background_fill`

Getting an Immediate Answer

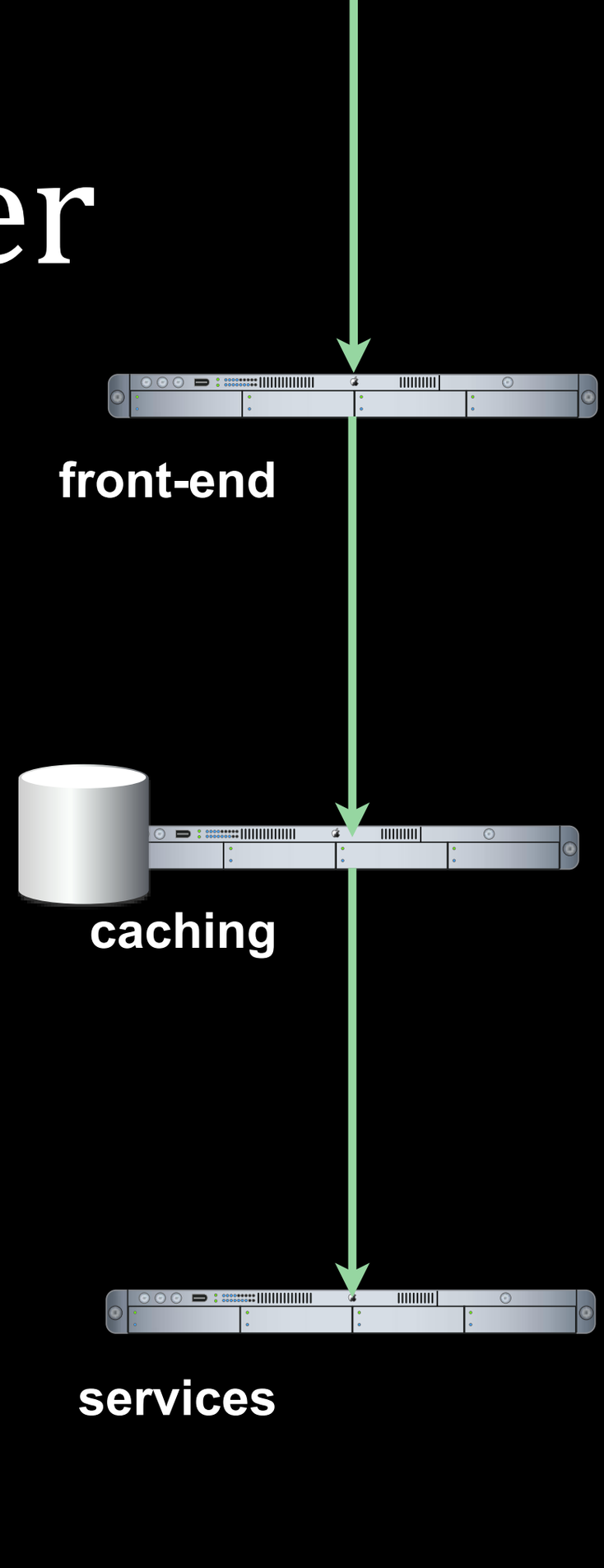
```
Cache-Control: only-if-cached
```

```
504 Gateway Error
```

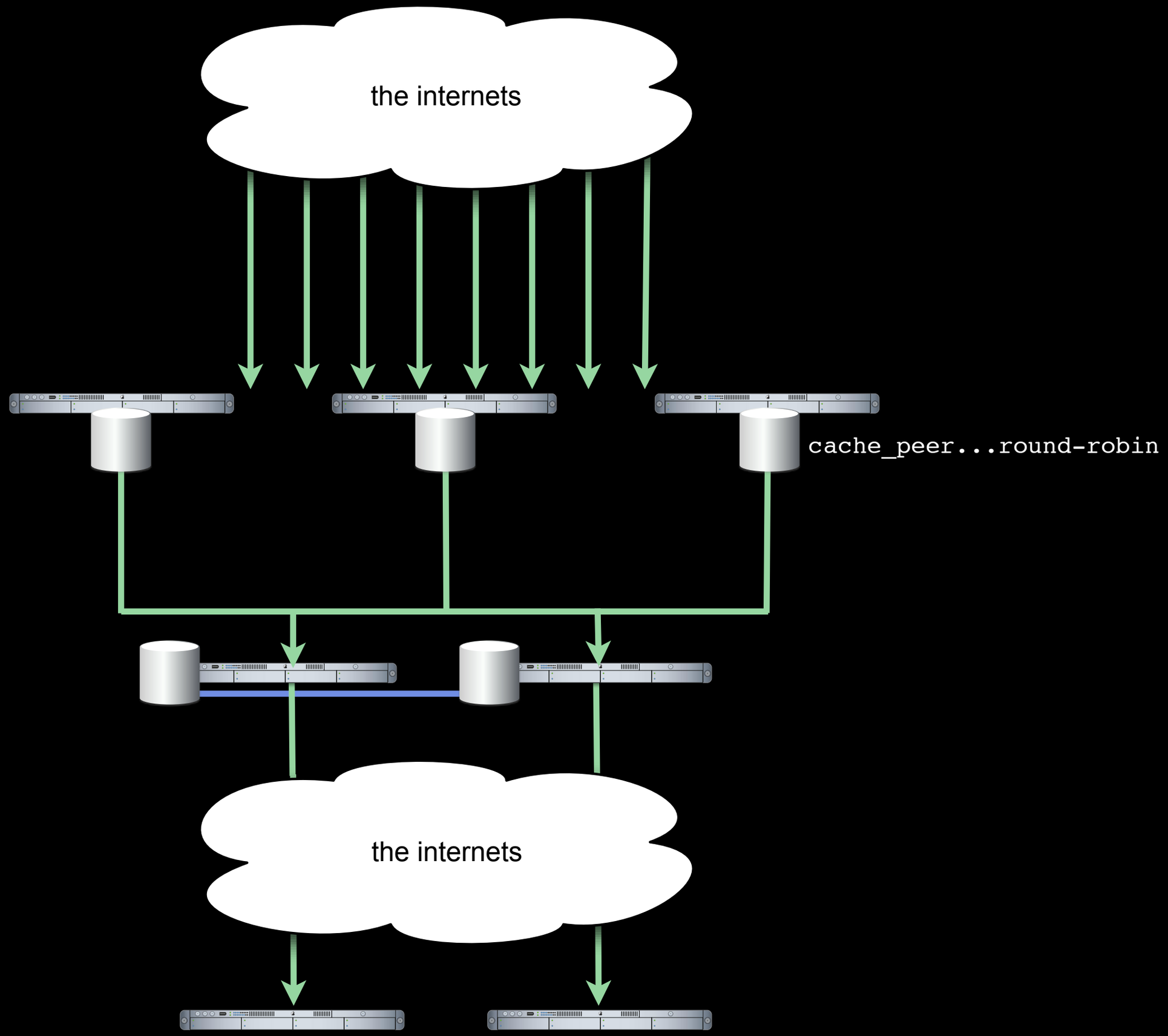
```
Cache-Control: max-age=3600,
               max-stale
```

Squid (soon)

```
fetch_only_if_cached_access
```



front-end
local caching

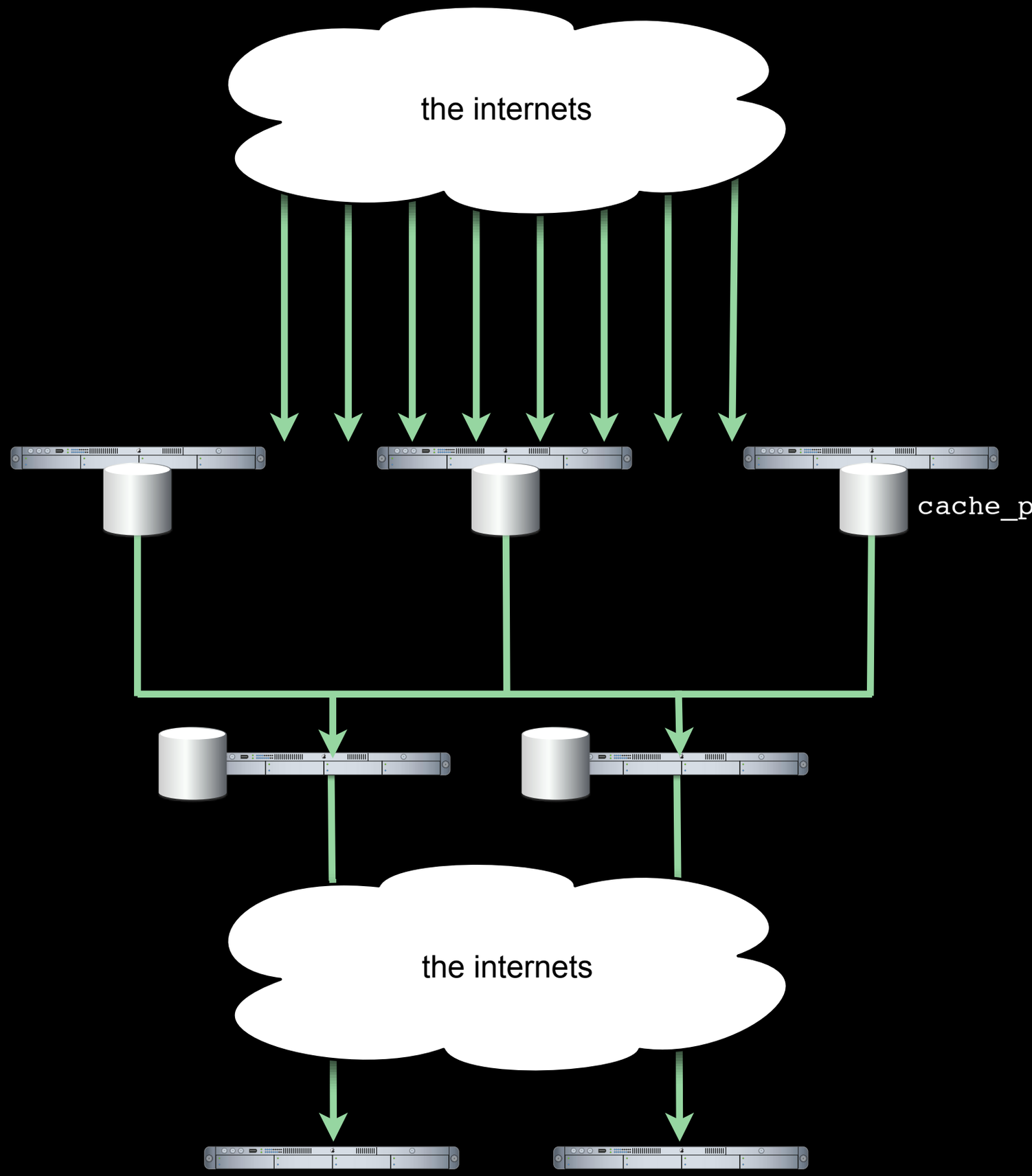


proxy caching

services

front-end

local caching



proxy caching

services

Why won't Squid cache that response?

Easy Answers

request Cache-Control

response Cache-Control

authentication

unfriendly freshness information

lack of LM/ETag

...in Squid

request Cache-Control

ignore-reload

response Cache-Control

ignore-[no-cache, no-store, must-revalidate, private]

authentication

ignore-auth

unfriendly freshness information

override-[expire, lastmod]

lack of LM/ETag

store-stale

```
refresh_pattern . 10 100% 10 [options]
```

...in Traffic Server

request Cache-Control

proxy.config.http.cache.ignore_client_no_cache

response Cache-Control

proxy.config.http.cache.ignore_server_no_cache

authentication

proxy.config.http.cache.ignore_authentication

unfriendly freshness information

proxy.config.http.cache.when_to_revalidate

lack of LM/ETag

proxy.config.http.cache.required_headers

```
dest_domain=example.com method=GET pin-in-cache=2d
```

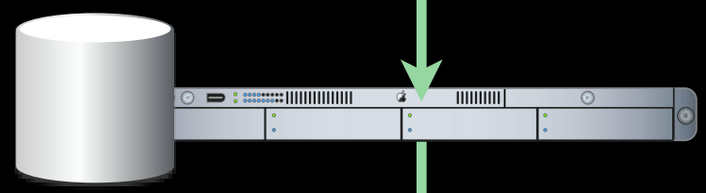
Not So Easy: Wandering URLs

```
http://srv254.dctr.example.com/foo/image.gif

http://example.com/thing.xml?uselessToken=abc123

http://example.com/endPointforEverything
```

storeurl_rewrite



http://a

http://b

http://a

No Answers*

non-GET methods

Protocol Errors

Vary: *

*without hacking

Your

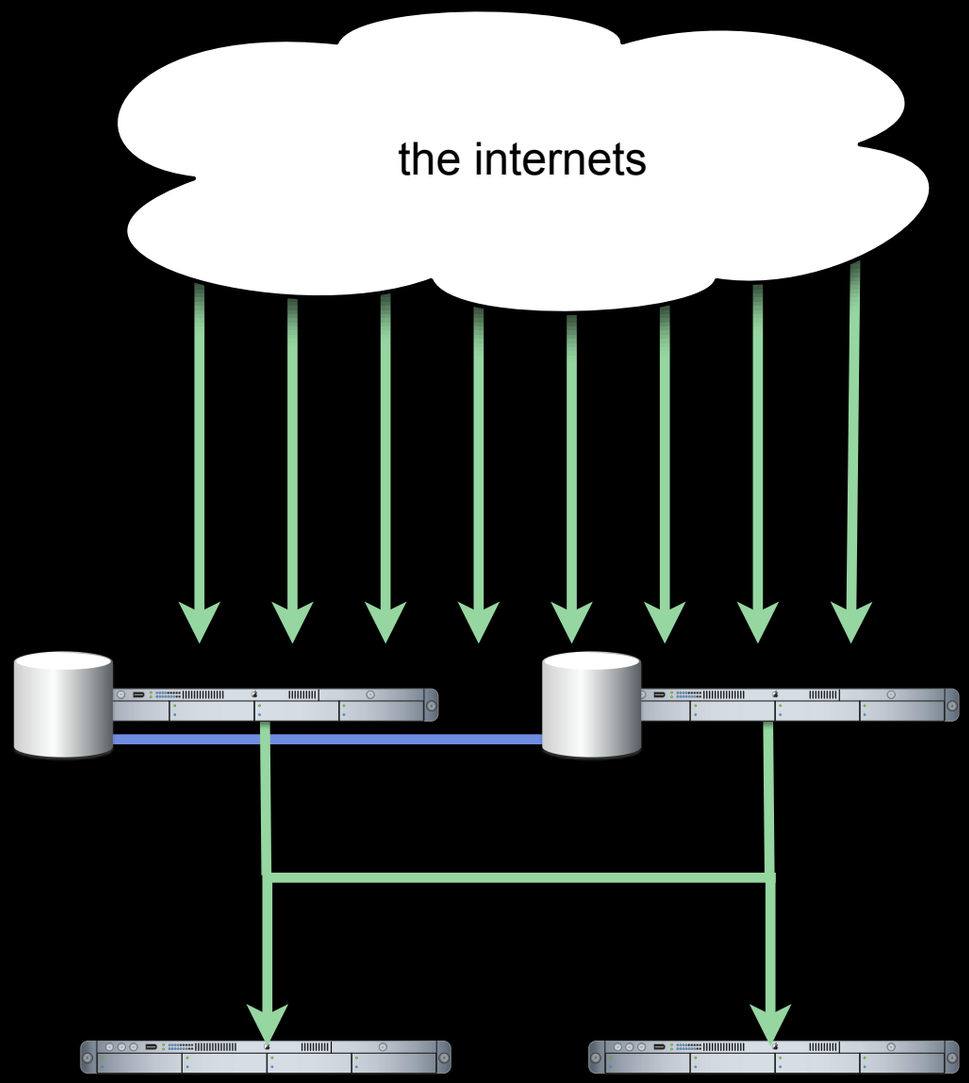
API

will

be

cached.

Accelerator Caching



proxy caching

services

/people?name=Britney_Spears&page=2

/people?name=Britney_Spears&page=2&

/people?NAME=Britney_Spears&page=02

/people?page=2&name=Britney_Spears

/people?name=Britney_Spears&page=02

/people?name=britney_spears&page=2

/people?name=Britney_Spears&page=2&token=abc

/people?name=Britney_Spears&page=2&user=jane

non-canonical URLs = low cache hit rate

```
<map base="http://example.com/">
   <path seg="images">
      <rewrite path=”pix”/>
   </path>
   <path seg="people">
      <query lower_keys="true" sort="true" delete="true">
          <page type="bool"/>
          <name type="lower"/>
      </query>
   </path>
</map>
```

XML format

local in-cache / fetched from site

Director

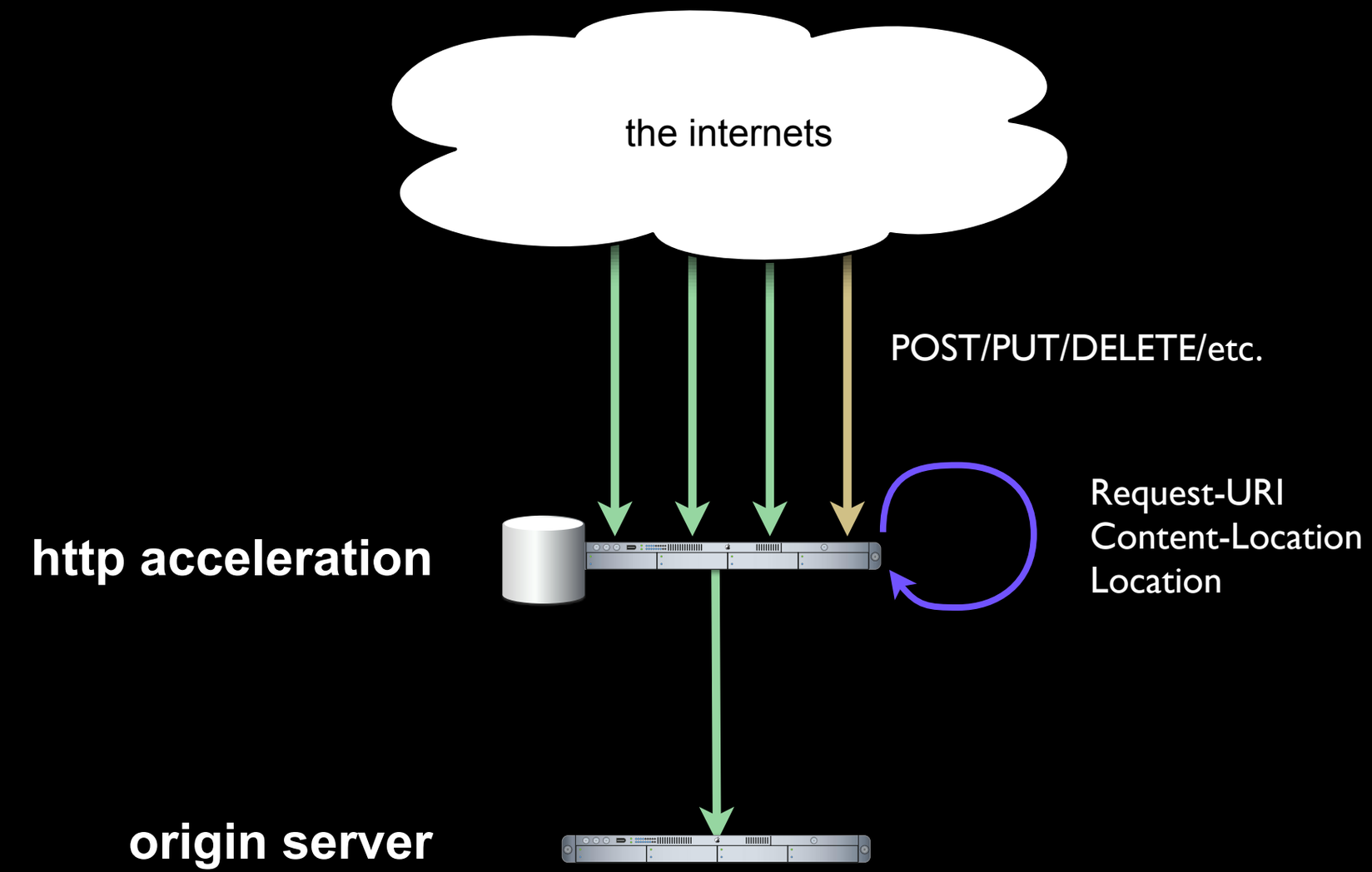
There are only 1 two hard things in CS:

cache & invalidation & naming things.

Phil Karlton

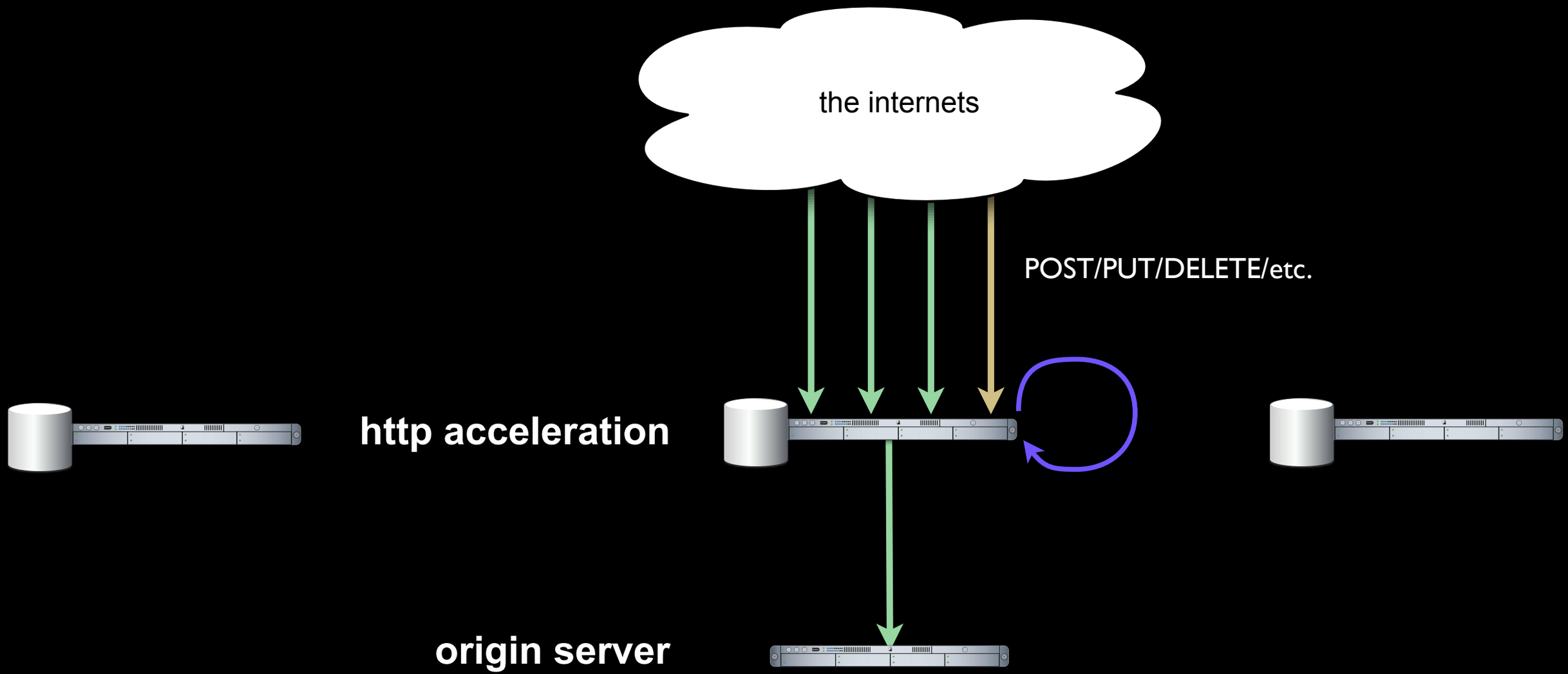
reliability,
scalability,
immediacy.

Choose two. Or maybe one.

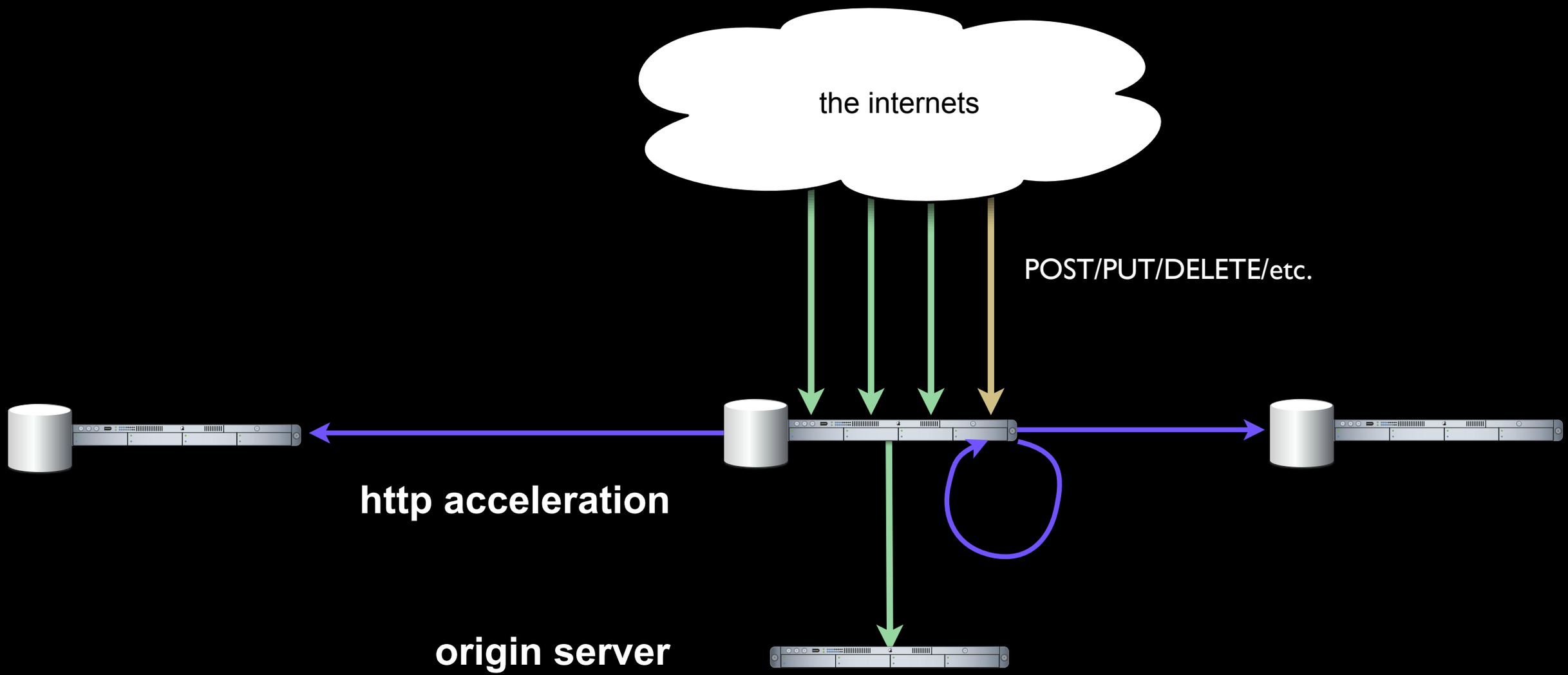


RFC 2616:

Invalidations after Updates or Deletions



Problem 1: Peered Caches



Sharing Invalidations with HTCP CLR

POST /articles/123/new_comment

/newest_comments

/articles/123/comments

/comment_feed

Problem 2: Related Responses

POST /articles/123/new_comment

/newest_comments

```
Link: </articles/123/new_comment>; rel="invalidat
```

/articles/123/comments

```
Link: </articles/123/new_comment>; rel="invalidated-by"
```

/comment_feed

```
Link: </articles/123/new_comment>; rel="invalidated-by
```

Link: rel=invalidated-by

POST /articles/123/new_comment

/cat/vuvuzela

/newest_comments

```
Link: </articles/123/new_comment>; rel="invalidat
```

/bob/comments

/articles/123/comments

```
Link: </articles/123/new_comment>; rel="invalidated-by"
```

/comment_feed

```
Link: </articles/123/new_comment>; rel="invalidated-by
```

Problem 3: Dynamic Relations

POST /articles/123/new_comment

```
Link: </cat/vuvuzela>; rel="invalidates"
Link: </bob/comments>; rel="invalidates"
```

/cat/vuvuzela

/newest_comments

```
Link: </articles/123/new_comment>; rel="invalidat
```

/bob/comments

/articles/123/comments

```
Link: </articles/123/new_comment>; rel="invalidated-by"
```

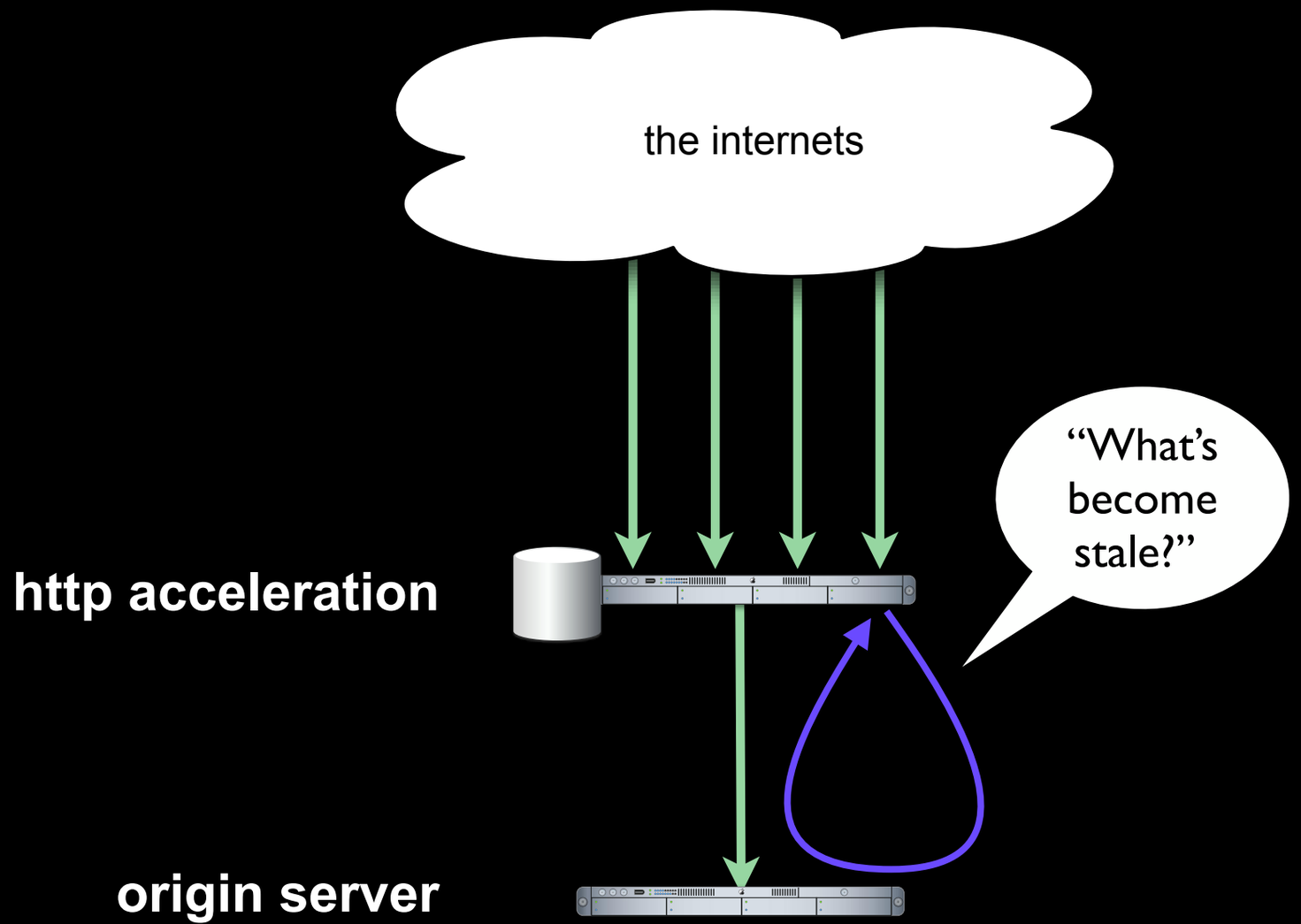
/comment_feed

```
Link: </articles/123/new_comment>; rel="invalidated-by
```

Link: rel=invalidates

“side effect” invalidation + link relations =

Linked Cache Invalidation



Cache Channels

Cache Channels

Good for: occasional tight control

Caveat: ~10-30s lag; not immediate

Bottleneck: number of events in channel

Linked Cache Invalidation

Good for: user-generated content

Caveat: not 100% reliable

Bottleneck: complexity of relationships

The whole point of using a Web cache is that you're

not

writing code.

http://www.squid-cache.org/

http://trafficserver.apache.org/

http://www.mnot.net/cache_docs/

http://redbot.org/

http://github.com/mnot/