Tiscali:
How to Build a Content Delivery Network

Anil Madhavapeddy & Alberto Crivelli | Network Appliance | April 2002 | TR-3152

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.

Network Appliance Inc.

Table of Contents

1. Introduction

This document is aimed at those interested in building a content delivery network to enable the distribution and delivery of rich media over wide area networks, such as the Internet or corporate WANs. It considers a specific large event—the streaming of a U2 concert across Europe. After studying this document, the reader should:

- Recognize the challenges associated with large-scale content distribution
- Realise the inherent benefits in caching and edge delivery of content
- Understand how to construct such a network to solve their own business needs

1.1. U2 Concert

RealBroadcast [1] streamed the U2 concert at Notre Dame, Indiana, live across the Internet. To help them deliver the rich streaming media and web content through Europe, Tiscali [2] provided the pan-European network infrastructure. Tiscali first constructed a content delivery network capable of handling tens of thousands of simultaneous live and video-on-demand streams robustly and securely. Because of the simplicity of the Network Appliance™ content delivery networking technology, this infrastructure was successfully architected, deployed, and tested in only **five days**.

In addition to providing the network, Tiscali had to ensure that their end users enjoyed a high quality-of-service when accessing the streams.

The content to be delivered consisted of:

- Web pages, images, and Flash movies
- Video-on-demand streaming media of preview footage and exclusives
- Live streaming content of the concert itself

There was a live broadcast of the concert in the early hours of the morning (GMT), and then a "live replay" the same evening, which was scheduled at a time more suitable for European audiences.

1.2. Partners and Roles

Partner	Role
RealNetworks	Provide original streams of the concert from the U.S.
Tiscali	Construct a content delivery network to deliver these streams across Europe
Network Appliance	Provide storage, caching appliances, and expertise in building the content delivery network

2. Background

2.1. Problems with Central Distribution

Tiscali has network presence across Europe in over 14 countries. However, if they want to deliver streaming media from a U.S. server, they need to have extremely high-bandwidth, high-reliability international links to each of these countries.

Any problems in the backbone would result in interruption of streams, and a less enjoyable experience for their users. Since there is so much traffic going over the backbone links, the chances of these problems arising also increases.

International burst transit bandwidth is significantly more expensive than local bandwidth. Thus, if the content became extremely popular, Tiscali's costs would rise, thereby penalizing them for serving the most popular content! This is in clear contrast to the traditional television model, where companies can broadcast content without facing additional costs if it proves to be popular.

2.2. The Case for Edge Delivery

To address the challenges described previously, Network Appliance has pioneered the concept of **content delivery networks**. In this model, content is stored centrally at a few key locations, and then pushed out to the edges of the network. Deployed at the network edge, the NetCache® appliance stores local copies of the content and serves it up to local users requesting it.

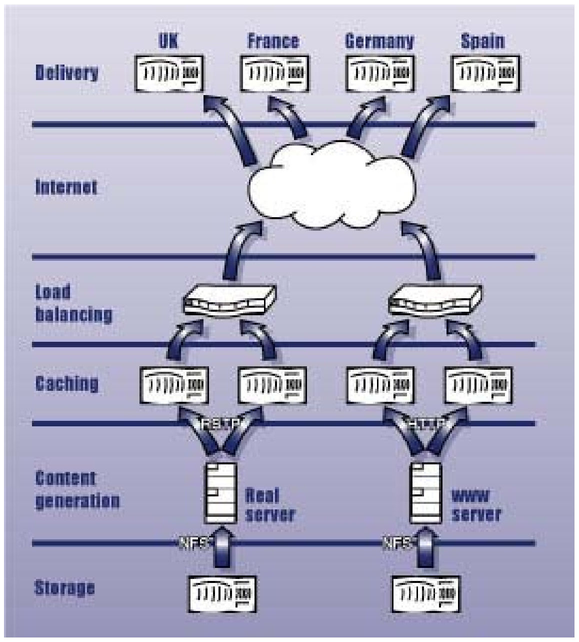
Thus, if a remote location has 1,000 users, the remote NetCache device retrieves the stream or Web page once, and serves that up many times. There are clear bandwidth savings with this approach, and the local users experience much better response times and quality when retrieving the content, since they are just going through their local networks.

Edge delivery ensures that, as the number of users of a service grow, the bandwidth costs do not, since the caching devices only have to retrieve a stream once and serve it to a large number of local users. Since local networks are generally more reliable than wide area networks, it is significantly easier to guarantee a certain quality of service to end-users.

For further information on distributed Web sites, please also refer to technical report 3071 [3] in the NetApp technical library [4].

3. Architecture and Deployment

3.1. Tiscali's Streaming Deployment



Tiscali operates a major data center in Cagliari, Sardinia. This location is equipped with multiple network connections to various international peers. It received the streaming media feeds from RealBroadcast in the United States.

This central data center has **RealServers** [5] that received the feed and provided licensing and accounting facilities. There are also web servers running **Apache** [6] and **Linux®** [7] that acted as the origin servers for the u2.tiscali.com [8] Web site.

NetApp filers [9] provided all of the fast, reliable central storage for the streaming files, Web content, and applications.

At every European location, clusters of **NetCache** [10] appliances were deployed that received the content from the Cagliari data center, and distributed it to their local clients.

The above figure illustrates the process by which content was moved from the center of the Tiscali network at Cagliari to the edge countries.

3.2. Origin Site

The purpose of the origin site at Cagliari was to provide a **reliable central location** that all the edge NetCache appliances could retrieve the master stream from.

At the data center, there were four RealServers. They received streaming content from RealBroadcast via Tiscali's dedicated international backbone to the United States.

In order to reduce the load on these systems, they were each accelerated by a bank of four NetCache C6100 systems. A cluster of Cisco Catalyst 6500 with content switching module boards performed load-balancing between the NetCache systems and presented a single virtual IP that all external clients could connect to. The NetCache devices and RealServers were connected via switched Gigabit Ethernet to ensure there were no bottlenecks in the local network.

Since the four NetCache systems had a lot of spare capacity and could handle HTTP traffic at the same time as streaming content, they were also configured to accelerate the u2.tiscali.com Web site. The Apache Web servers were deployed on the same gigabit LAN as the RealServers, and the u2.tiscali.com DNS was directed towards the NetCache virtual IP.

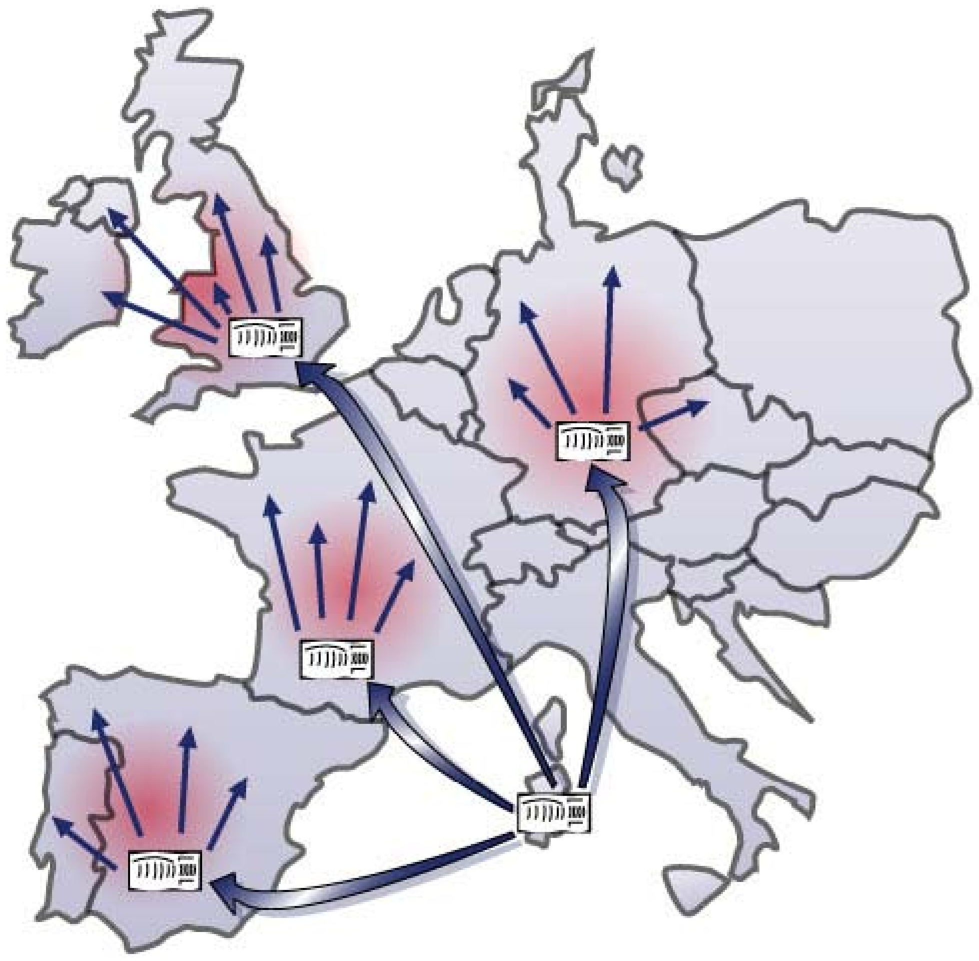
The NetApp filer was used to provide the fast centralized storage facilities for the Apache and RealServers. Each of the servers accessed its content from the filer via NFS over the gigabit LAN. Please refer to NetApp TR3104 [14] for more information on using NetApp filers in a CDN environment.

The end result of this was a single virtual IP address that was reliably available to any external clients or caches to retrieve the U2 streaming and Web content.

3.3. Remote Sites

At the remote sites, NetCache appliances were configured to act as **streaming accelerators**. In this mode, NetCache accepted requests for streams from clients, retrieved them from the origin site if needed, and then served the stream. So from the client's point of view, they were just connected to a local Real server that had the stream that they requested.

The NetCache device retrieved a single stream from the origin site in Cagliari using RTSP-over-TCP, but served up this content to clients via UDP, TCP, or HTTP-encapsulation. It also automatically thinned streams to local clients who had low-bandwidth connections such as modems.



The figure illustrates how the caches spread around Europe retrieved a single master stream from a central location, and split that to their local clients. Only one stream was requested by each NetCache appliance, and they served it to tens of thousands of local users.

High-end clusters of NetCache (C6100) appliances were deployed in countries of major presence for Tiscali, such as **Italy**, **United Kingdom**, **France**, **Belgium**, **Holland**, **Denmark**, and **Germany**.

For countries with a smaller presence, such as **Switzerland** or **Denmark**, mid range to low-end NetCache (C2100/C1100) appliances were deployed.

3.4. Global Request Routing

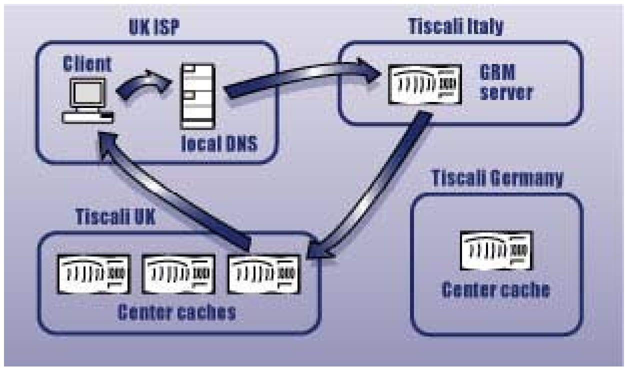
3.4.1. Introduction

The architecture described above has NetCache devices at the most appropriate edge locations. However, one last problem remained: **how to redirect user requests to their nearest healthy NetCache**.

Since requests for the U2 content were coming in from many different networks (Tiscali and external users from other ISPs), a Layer 4 solution was not sufficient. Also, it would prove to be very expensive to deploy a Layer 4 switch beside every remote NetCache.

Network Appliance solves this problem by introducing a global request-routing solution called the **Global Request Manager**, or GRM.

GRM allows a NetApp CDN to assume authority for a DNS domain [11] and to resolve it to the most appropriate NetCache for that requesting client's DNS server.



The figure illustrates how a request from a user in a UK ISP would be received and automatically redirected from Italy to the nearest Tiscali NetCache in the UK.

Since the GRM resolution all happens at the DNS level, it can be used for a variety of services, from streaming to newsgroups to Web pages.

A GRM network consists of two components, a server and an agent. These run on any standard NetCache appliance running version 5.2 or later.

- **GRM Server:**
  This is a NetCache system that can assume authority for a DNS domain and respond to DNS resolution requests for that domain. In this deployment, Tiscali registered cdn-tiscali.com for its CDN, and assigned this to two GRM server NetCache systems. Each of the GRM servers acted as an authoritative DNS server and resolved requests for stream.cdn-tiscali.com into IP addresses.

- **GRM Agent:**
  Every remote NetCache runs a GRM agent, which communicates with the GRM servers every 30 seconds to establish a heartbeat. In this heartbeat, it communicates information such as supported protocols and current load to the GRM servers.

3.4.2. Typical Operation

A streaming URL was encoded in the form:

```
rtsp://stream.cdn-tiscali.com/broadcast/u2_live/concert.rm
```

When a client attempted to resolve stream.cdn-tiscali.com, this is what happened:

1. Client made a recursive DNS request [12] for stream.cdn-tiscali.com to their local DNS resolver.

2. The ISP DNS resolver started to make iterative resolution requests [13]. First of all it contacted the Internet *root name servers* to find out the authority for .com, and then it contacted those to find out who controlled cdn-tiscali.com.

3. The ISP DNS resolver made a request to one of the GRM servers (which were the authority for cdn-tiscali.com) and asked for a set of IP addresses for stream.cdn-tiscali.com.

4. The GRM server looked in its database to check if it had proximity information about the requesting IP address.

5. If it had no information, then it returned the IP address of a random cache, but with a very low Time To Live (TTL).

6. Meanwhile, the GRM Server instructed every GRM Agent cache to attempt to discover proximity to the requesting ISP DNS resolver by using ICMP pings, trace routes, or DNS lookup time.

7. When a certain number of agent caches returned back information that appeared consistent, the GRM server stored this in its database. The next time the resolver made the request for the domain, it returned the set of caches that were closest to it.

Over time as the GRM server ran, it built up a database of DNS resolvers for major ISPs and eventually became very accurate about calculating proximity to remote client requests. It also coped with transient network errors very well, by ensuring that the different agent replies were consistent with each other.

After three days of operation, the Tiscali GRM server had approximately 4,000 local DNS entries in its database, which represented the result of over half a million resolution attempts. Please refer to Section 3.5.4 for further analysis.

3.4.3. Proximity Checks

The GRM server can instruct GRM agents to perform three different kinds of proximity checks to the requesting DNS server. Each agent performs one of these checks and reports its results to the GRM server, which builds up a knowledge database.

1. **ICMP Echo:**
   The agent cache attempts to "ping" the target and measures the latency in the ICMP echo response. This is the preferred method of proximity detection, and the other two are used only if this fails.

2. **DNS RRT:**
   The agent measures the amount of time it took to perform a reverse DNS lookup on the target.

3. **Traceroute:**
   Each agent attempts to perform a traceroute to the target, and records the latency to the last hop that was reached. If at least three agents agree about the location of the last hop, the proximity check is accepted as valid.

These three methods are necessary due to various firewalls and packet filters that drop ICMP packets, hence not allowing agents to succeed with their ping test. However, the traceroute method will allow the agents to measure as far as the firewall itself, as long as at least three agents agree that the last hop was a single IP address. This ensures that transient routing failures do not poison the GRM database if a traceroute failed temporarily, while still allowing firewalls to be detected.

3.4.4. Center and Edge Caches

The caches described above are known as **center caches**, since they sat at central network peering points and performed proximity checks to incoming requests from various external clients.

In addition to these center caches, it is possible to define **edge caches**, which are assigned to fixed blocks of IP addresses. This is particularly useful for an ISP environment, where a point of presence (POP) consists of a fixed block of addresses that should always be served by a locally installed NetCache. By defining that NetCache as an edge cache, the GRM server will always reply back to requests from that block of IP addresses with the addresses of the local NetCache appliances.

3.4.5. Load-Balancing and Failover

Since most remote sites had more than one NetCache devices, it was useful to define **edge groups** and **center groups** that represented groups of NetCache devices in a single location.

When the GRM server returned information to a DNS resolver, it didn't just return a single IP address. Instead, it included multiple A records in the DNS response with other caches from the same edge or center group. If there were only two caches in an edge group, the server attempted to include some center caches if they were in close proximity.

This ensured that requests for the content were evenly spread over the multiple caches at a single POP. Intermediate DNS caches rotated the multiple A records with every request, further ensuring even balancing. Here is an example of this behavior being tested from a UNIX® prompt:

```
openbsd> nslookup stream.cdn-tiscali.com
Server:  localhost
Address:  127.0.0.1

Name:    stream.cdn-tiscali.com
Addresses:  212.35.2.195, 212.35.2.193, 212.35.2.197

openbsd> nslookup stream.cdn-tiscali.com
Server:  localhost
Address:  127.0.0.1

Name:    stream.cdn-tiscali.com
Addresses:  212.35.2.193, 212.35.2.197, 212.35.2.195
```

Note that the order of the list of IP addresses rotates to put a different one at the head.

If a NetCache lost network connectivity, it failed to issue its regular heartbeat to the GRM server and was removed from the responses until it came back online again. Its IP address remained active for up to 60 seconds, but this was not a problem as most modern clients automatically failed-over to the next IP on the list of responding caches.

When the NetCache device rejoined the GRM network, it synchronized itself automatically with a GRM server, learning any new activity that occurred while it was offline.

This failover mechanism was observed to work successfully during the streaming of the U2 concert when a remote NetCache system in the UK lost its network connection, and its partners in the data center picked up the extra load and continued to offer service to UK customers.

3.5. Management and Monitoring

With around 50 NetCache systems spread throughout Europe, central management capability was very important to ensure that all of the appliances had consistent configuration and setup. Also, all of the NetCache appliances had to be monitored for problems and overloading.

3.5.1. Log Files

Each of the NetCache systems maintained a set of log files that contained detailed information about what content it had served to local clients. These included:

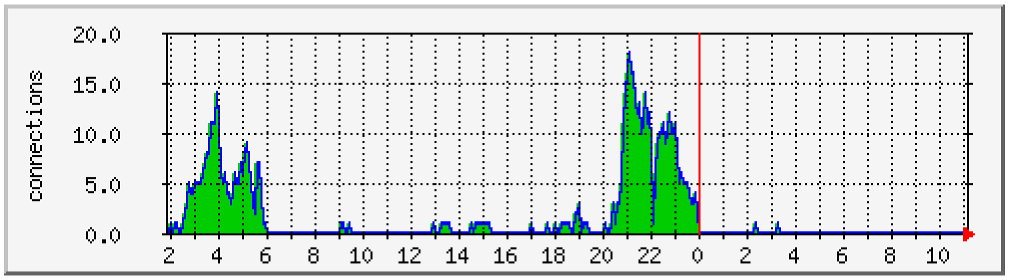
- Streaming access log
- Streaming details log
- HTTP log
- Messages log

There were also other log files for other protocols, but these were not relevant for this deployment.

Since the log files provided an accurate audit of every action NetCache took, it was very important to have them available centrally for post-processing and analysis. The NetCache were configured to **automatically push log files** to a central NetApp filer when they reached a certain size, or after a period of time. So after the event, all of the log files were already stored reliably on the Filer, waiting to be processed by ContentReporter software.

3.5.2. SNMP

Network Appliance publishes an SNMP MIB for the filer and NetCache product, so that they can be monitored via any package that understands SNMP. Tiscali deployed the excellent freeware tool MRTG at the Cagliari data center, and used it to poll all of the remote NetCache devices and give accurate graphs back about bandwidth savings, total throughput, and so on.



The above graph was generated via MRTG, and plots the number of thousand streaming and HTTP connections from one of the central NetCache systems to remote clients. There are two spikes since the U2 concert was replayed after the initial live broadcast.

3.5.3. Configuration Control

NetCache appliances control their configurations via a **registry** that contains a hierarchial set of key/value pairs. For example, the hostname is determined by:

```
config.system.hostname
```
or the RTSP values are all under
```
config.rtsp.*
```

These configuration values can be edited by the Command-Line Interface (CLI) of the NetCache, or via the more user-friendly web interface. Here is an example of the CLI:

```
netcache-milan-1> show config.rtsp.*
config.rtsp.real.max_connections = 0
config.rtsp.rtcp_port = 2001
config.rtsp.udp_port_range = 6970-65000
config.rtsp.transparency = off
config.rtsp.ip_spoofing = off
config.rtsp.ui.enable = on
config.rtsp.enable = off
config.rtsp.ports = 554
netcache-milan-1> set config.rtsp.enable on
```

In the above example, RTSP was turned off on the NetCache, but the command was entered to enable it by modifying the appropriate registry key. However, instead of specifying this command manually for every NetCache, a remote URL can be sourced. To illustrate:

```
netcache-milan-1> set < ftp://filer.tiscali.it/streaming.cfg
```

This caused NetCache to contact the FTP site specified in the URL, retrieve the configuration file, and execute all the commands it found in there. Configuration files were created for the streaming, HTTP, and access control lists settings. As a result, the only manual changes each NetCache needed were the network IP address, netmask, and other specific local details.

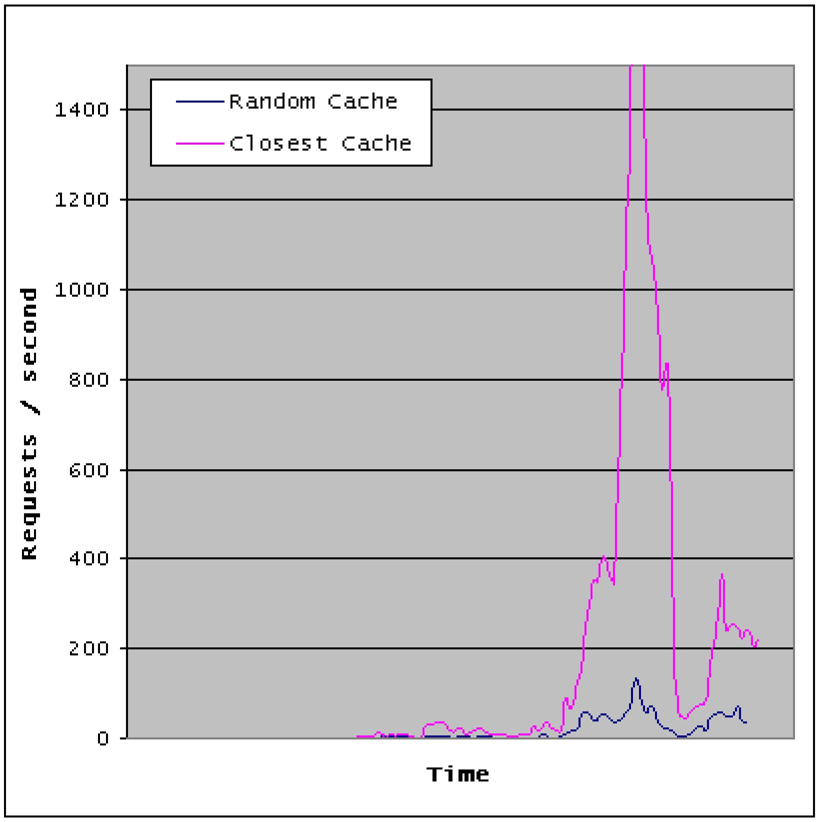
This process was automated, so that each remote NetCache polled a remote URL via HTTP, HTTPS, or FTP regularly to determine if its configuration had changed or not. So by altering a single file on the central filer at Cagliari, every remote NetCache could automatically be updated to new settings.

Since multiple URLs can be specified, it was easy to separate out the settings for different protocols or security settings, and mix and match them between NetCache appliances that had different requirements.

3.5.4. Global Request Manager

In addition to MRTG, the GRM server NetCache also provided a graphical snapshot of the current activity over the caching network. This view included the status of each cache, its current load, and what protocols it had active. This enabled the administrator at the GRM Server to quickly determine at a glance how well the network was functioning, and if any of the regions were overloaded and required additional resource.

The GRM server also provided detailed log files which were later analyzed to determine how well it returned accurate results.



The above graph illustrates an analysis of the GRM logs during the U2 concert. The purple line represents the closest cache response, when the GRM server authoritatively replied back to the user with its nearest point of service. The blue line represents when the GRM server replied back with a best guess reply while instructing the agents to send out proximity probes. Note that the number of random-cache responses are relatively stable when compared to the number of closest-cache responses, proving that the GRM system stabilized very quickly and successfully redirected the majority of users to the right NetCache.

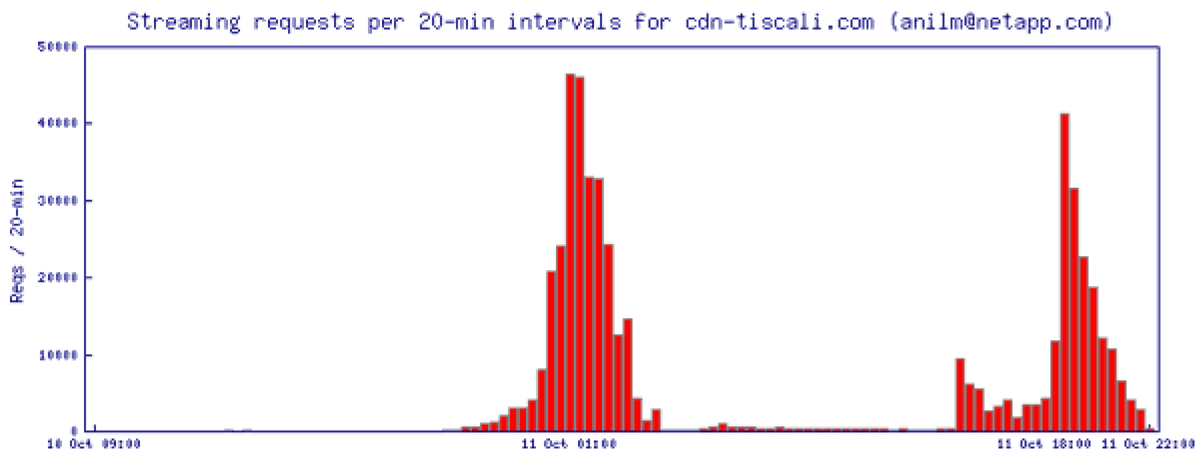
In future, Tiscali plans to accelerate its Web site via GRM as well. This provides a major benefit to the streaming portion, since by the time user finishes clicking through the Web site, the GRM would have had time to determine accurate proximity information for the user, and can immediately offer every user an initial stream from its local NetCache.

The longer the GRM system runs, the bigger a knowledge base it builds up, making it progressively more accurate and confident as time goes on.

4. Summary

4.1. U2 Concert

The U2 Webcast was declared a complete success. The entire caching network across Europe was architected in one week, which illustrates the simplicity and robustness of the NetApp CDN model.



The overall CDN handled approximately 500,000 streaming requests and 8,000,000 HTTP requests over a 12 hour period, with over 15,000 simultaneous high-bandwidth streaming connections at the peak time. The individual NetCache devices never exceeded a 15% load average, illustrating that the network could handle many more connections if required.

More importantly, Tiscali's bandwidth usage across its backbone did not register any significant spikes, proving that edge delivery successfully spread the load of the concert across its European network.

4.2. Sting Concert

Ten days after the U2 concert was Web cast, Tiscali received details of a Sting concert that was taking place in New York City. This concert was different from the U2 concert since it was only encoded in Windows Media Format. However, Tiscali successfully broadcast this concert to its users without any changes to the caching infrastructure, since the NetCache devices were all multiprotocol and supported Windows Media as well as Real and QuickTime™.

4.3. Future Developments

Tiscali plans to use this content delivery network to help it deliver more successful events to their customers in the future.

Since the NetCache product implements all three major streaming protocols natively, it is capable of providing an authentication framework that is protocol-independent.

This enables Tiscali to deliver pay-per-view content from a common authentication and accounting framework independently of whether Real, Microsoft® or QuickTime formats are used. This gives them maximum flexibility when choosing content, and grants consumers more choice about which players they want to use.

The content adaptation protocol ICAP is being examined to enable intelligent modification of Web content at the local level. This would open new revenue opportunities in areas such as targeted advertisement insertion into Web pages: encoding into WML, GPRS, and other wireless formats; or even language translation from English into the local language.

5. References

[1] **RealBroadcast**: *http://www.rbn.com/*

[2] **Tiscali**: *http://www.tiscali.com/*

[3] **"Application of a Distributed Web site: Mars Polar Lander"**: *NetApp Tech Report 3071*

[4] **NetApp Technical Library**: *http://www.netapp.com/tech_library/*

[5] **RealServer**: *http://www.real.com/*

[6] **Apache**: *http://httpd.apache.org/*

[7] **RedHat Linux®**: *http://www.redhat.com/*

[8] *u2.tiscali.com*

[9] **NetApp Filers**: *http://www.netapp.com/products/filer/*

[10] **NetCache**: *http://www.netapp.com/products/netcache/*

[11], [12], [13] **DNS and BIND, 3rd Edition**

[14] **"Filers in a CDN Environment"**: *NetApp Tech Report 3104*