Internet Broadcast Systems

Afterburner Patent document

written by
Mayur Jobanputra

started: October 8, 1998
last edited: November 6, 2006

Table of Contents

-

-

INTRODUCTION.. 3

HISTORY.. 3
HOW WEB SERVERS ARE LIKE FTP SERVERS.. 5
How FTP works.. 5
How WWW servers work.. 5

THE COMPETITION **6**
THE INTERNET VIDEO STREAMING INDUSTRY 6
THE CACHE SERVER MARKET 9
How cache-based servers works *9*
Current competitors *10*
HOW THE AFTERBURNER SERVER WORKS **12**
INTRODUCTION 12
AFTERBURNER ADVANTAGES 13
AFTERBURNER PSEUDOCODE 13
AFTERBURNER CONCEPTUAL MODEL 17
BENCHMARK RESULTS 17
APPENDIX 1 - BRIEF DESCRIPTION OF COMPETITORS PRODUCTS **18**
Microsoft Proxy Server 2.0 .... *18*
Netscape Proxy Server *19*
Novell Border Manager *21*
Novell FastCache *22*
Inktomi TrafficServer *24*
Network Appliance Netcache *25*

-

Introduction

Everyone knows about the Internet. Few people, however, understand how it works and the fine details of how computers connected long distances via a common set of protocols and on a shared telecommunication resource works. The Internet has changed rapidly over the past decade and will experience the same level of change for the next decade and beyond. The Afterburner is a revolutionary Internet server, or information provider as another way of looking at it, that can simultaneously 'serve' more visitors to a given website than other servers (which are based on more traditional techniques). An explanation of how it does this first requires a further investigation into how the Internet works and its rapid improvement in technology over the past 30 years.

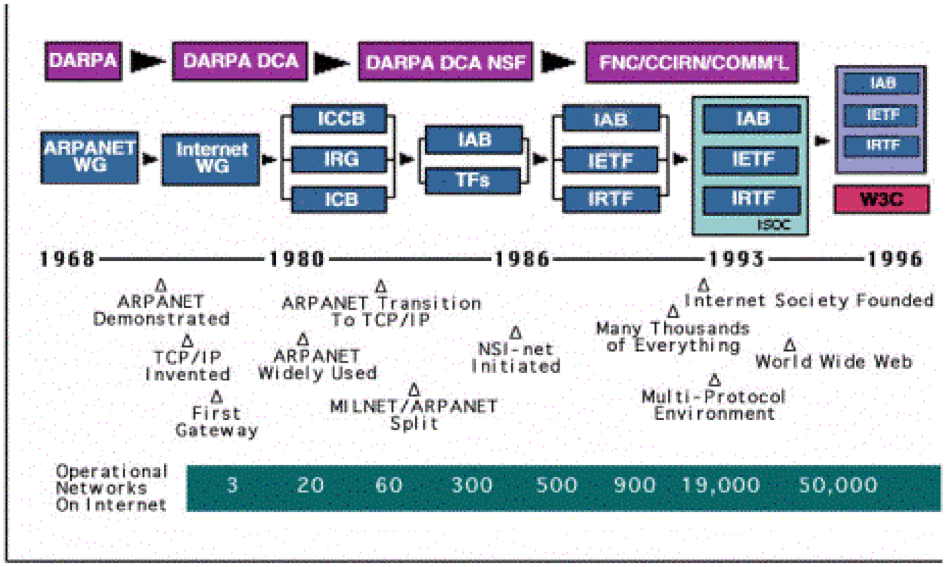
History

(below section is designed for novice Internet users)

The real concrete proof of anything close to the Internet started with ARPANET, a US Defense project to investigate packet networks. ARPANET proved the viability of using packets and packets switching technology using two computers between UCLA and Stanford. The objective was to explore how well multiple, linked computers could communicate transparently over different hardware and communication platforms using packet control.

Packets can be compared to pieces of a jigsaw puzzle. A single, whole picture is formed from putting together many different pieces. When placed together, the picture can be formed, but each piece, or packet, wouldn't make sense on its own. The packets need to be arranged in a unique configuration and only then can you see the whole picture. Now imagine if your friend had a completed jigsaw puzzle but he was passing this entire jigsaw puzzle to you, piece by piece, across a room full of people and each of these people would be required to pass pieces, one by one, over to you. There are numerous different people that a given piece may travel across and the path might not always be the same. The most efficient path would usually be taken but could easily change if someone was too busy to pay attention. Finally, if someone forgets to pass a piece of the jigsaw puzzle on to you, your friend could simply send another copy over. In order for you to make the puzzle, your friend would also have to number and code each piece and there would have to be a way for you to communicate back to your friend if a given packet was completely lost. This in essence is how packets work and is integral to the proper functioning of nearly every type of Internet protocol devised today. The concept of packet theory was first devised by who Leonard Kleinrock at MIT who published the first paper on packet switching theory in July 1961 and the first book on the subject in 1964.

After 1969 when packet experiments were proved with ARPANET, computers were quickly added to the network and work continued on a host-to-host protocol, called the Network Control Protocol. In 1972, ARPANET first demonstrated their work at ICCC (International Computer Communication Conference) which quickly lead to the first 'hot application' for communication, email. For the next decade, many more university and research computers became nodes on this growing network and made e-mail available to professors, students and researchers alike.



Besides being able to send messages over a more reliable network topology and robust means of communication, the packet network devised by ARPANET allowed "open architecture networking". That is, different network architectures can be seamlessly connected to form a wider area network, we now know as the Internet. Throughout this transition, the primary protocol in use was the Network Control Protocol. NCP lacked in its error control and reliability and was the succeeded by the protocol we know today as TCP/IP (or Transmission Control Protocol/Internet Protocol).

Some key ideas were formed for the design of this protocol:

1. each network would have its own internal mechanisms of communication and wouldn't need drastic changes to become part of the Internet
2. packets that didn't make it to the destination would be re-transmitted by the server
3. black boxes would connect the individual networks to the wider area network or Internet. We know these now as gateways or routers

What started as time-sharing experiments for wide area networks at Xerox, IBM and the like ended up as the basis for how computers interact on the Internet. Widespread development of cheaper and faster PC's in the 1980's allowed users to connect more and more machines to the Internet. Since every host machine on the Internet was sending and receiving packets that were uniquely theirs and didn't belong to another host machine, a domain name system was required to resolve a given host name into an IP number. This system was necessary because the router tables were growing exponentially as more and more computers became nodes on the Internet.

So many new organizations have been formed and re-organized again and again to try and maintain a fair system of participation and standardization in this new economy. In just 20 years, the Internet has grown from several hundred mainframe machines connected via 56 kilobyte per second connections to over 100,000 small and large nodes connected via 45 megabyte per second (and faster) connection on every continent in nearly every country and even in outer space. With the recent formation of the world wide web, commercial activity has exploded at a furious pace as entrepreneurs enter the arena of this new market economy. Purpose-built companies headed by highly technical individuals and a new generation of young, computer-savvy individuals are finding niche markets across every facet of Internet communications.

This rather lengthy discussion brings us to web servers and how the phenomenon of the Internet has fueled growth for this niche market. Ultimately, Internet growth can only grow as fast as new technology is developed to host , translate, and transport information faster and more efficiently. We expect Afterburner to make a significant impact in this competitive market.

Most end users connect to the Internet using dumb terminals, or low-powered personal computers. On their own these end user machines couldn't possible handle the processing power required to connect themselves to the other nodes on the Internet. More often than not, these machines dial a local number or are connected on a local area network to larger 'servers' that serve the information they want and ask for the information they need.

Afterburner can make a significant impact on servers that experience high hit volumes like search engine sites, media intensive sites that serve still and motion images and video, and other large sites. Currently these sites are held on extremely large and powerful machines. Theoretically, these machines can, at best, respond to 5 million hits (a given file or object request), their real capacity is only 500,000 to 1 million hits per day.

How web servers are like FTP servers

How FTP works

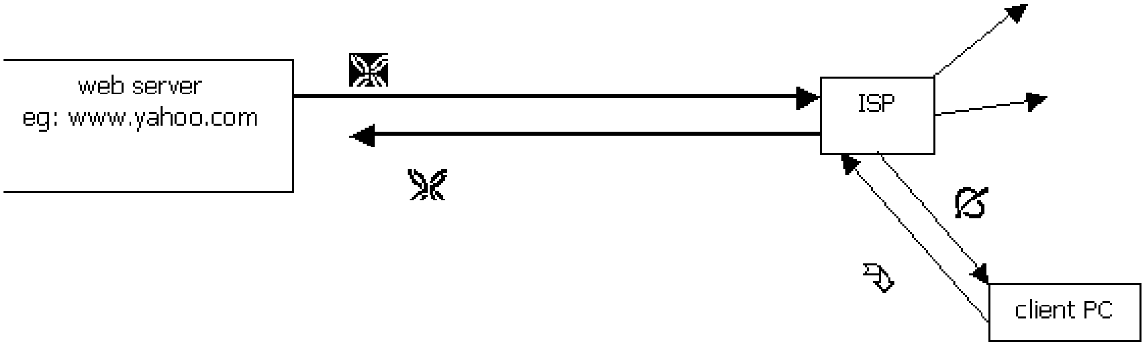
FTP stands for file transfer protocol. During the early phases of development of the Internet, researchers used this system of FTP servers to transfer documents across to other researchers. Because the number of nodes on the Internet at the time was not great, these servers were programmed to using a new thread process for each request made to the server. A user accesses the FTP server and is given permission (read, write, modify) based on their username and password. Anonymous users typically get access to only read (this includes download) the files from the server

How WWW servers work

WWW servers operate a lot like FTP servers and can be said to be a natural extension of FTP servers. Clients still access files but the assumption by default is that the user is anonymous. The WWW server is open to all and because of the ease by which information can be appealingly displayed, WWW servers have been a major influence for the Internet's rapid growth in the past 5 years. The core difference between FTP and web servers are that web servers are required to handle a much higher hit load and must be able to support more concurrent users without any lag time between a user requesting a file and the server sending that file out. Creating a new thread for each user that hits the web server at a given moment isn't efficient. For this reason, web server programming, and in particular socket and packet programming must be much more robust. Afterburner clearly shows attention to these features and in particular to a single threaded concept. Serving out of RAM, the fastest type of memory on your computer, also enhances performance.

Below is a conceptual model for how a user would connect to an ISP and get web pages and graphics from a server.

How clients get files from WWW servers:



1. Client connects to ISP via a phone line or high-speed local area network connection and asks for a file (say an image)
2. ISP (Internet Service Provider like Compuserve or AOL) takes the request and passes it on to the web server

3. Web server processes request and sends an image back to the ISP
4. ISP sends image to client PC

Shown above the web server is only processing one request from one ISP. But typically, a given web server is connected to every single other web server and ISP on the Internet and if popular enough could thousands of hits per second.

Why do current web servers fail?

Current web server are largely based on research and development completed decades ago. This may not be very old but Internet hardware and software solutions have been maturing so rapidly that what worked 10 years ago, simply can't suffice for the demands placed on servers today. Chief among the problems current web servers face is poor socket programming which leads to limitations in both response times and the maximum number of connections that a web server can provide in a given moment.

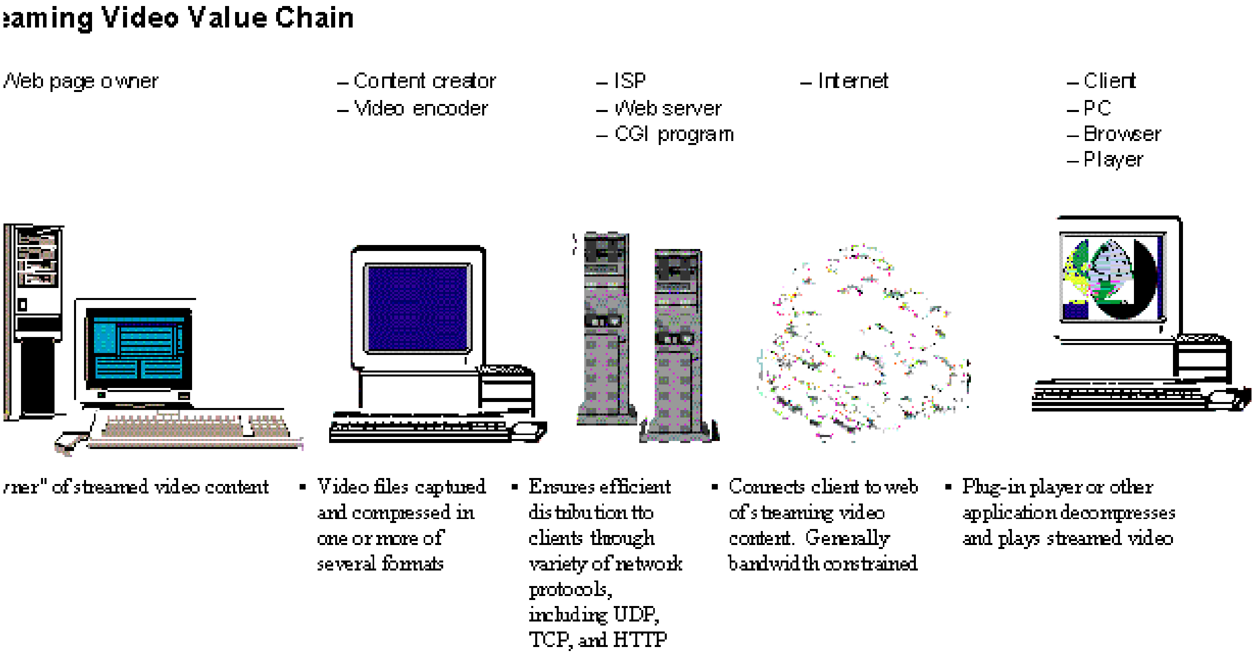
The Competition

The Internet Video Streaming industry

The video streaming industry has created a very high threshold for success by several players all vying for leadership on a relatively clear and defined value chain. Furthermore, the underlying infrastructure of the Internet's backbone network (the high speed pipes that transport data between servers, routers and hubs) will likely open the gates for widespread high-bandwidth transmissions over the Internet in the foreseeable future.

The streaming video market contains competitors from several key areas: Content owners, Content compressors/Encoders, Video servers, and Client players. Realnetworks (www.real.com) has maintained a clear position in the industry in the provision of video streaming solutions over the past two years but falls in some key areas of video compression, quaility, and video server solutions.

In order to establish dominance in this industry it will be essential to provide TV-Quality, VCR-functional, media-rich video content over the Internet in conjunction with an aggressive and sophisticated sales and marketing effort.

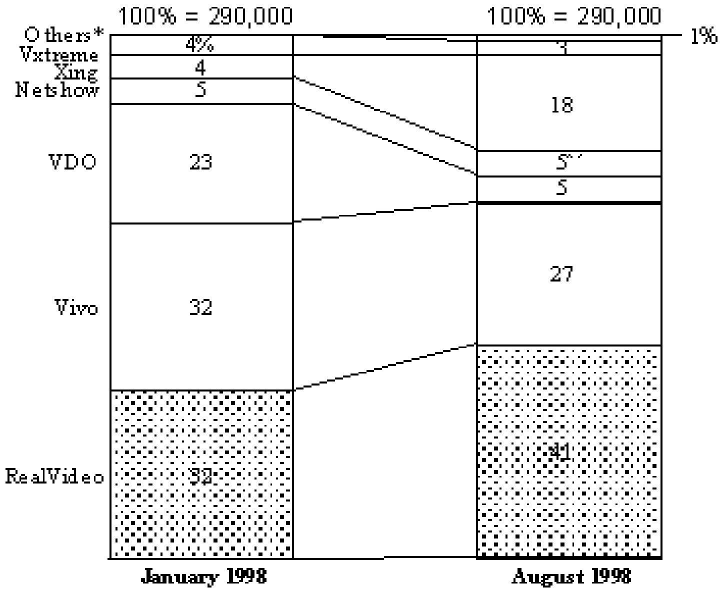

eaming Video Value Chain
Veb page owner
– Content creator
– Video encoder
– ISP
– Web server
– CGI program
– Internet
– Client
– PC
– Browser
– Player
ner" of streamed video content
▪ Video files captured and compressed in one or more of several formats
▪ Ensures efficient distribution tto clients through variety of network protocols, including UDP, TCP, and HTTP
▪ Connects client to web of streaming video content. Generally bandwidth constrained
▪ Plug-in player or other application decompresses and plays streamed video

There are several players vying for competition within each of the individual market segments for this industry:

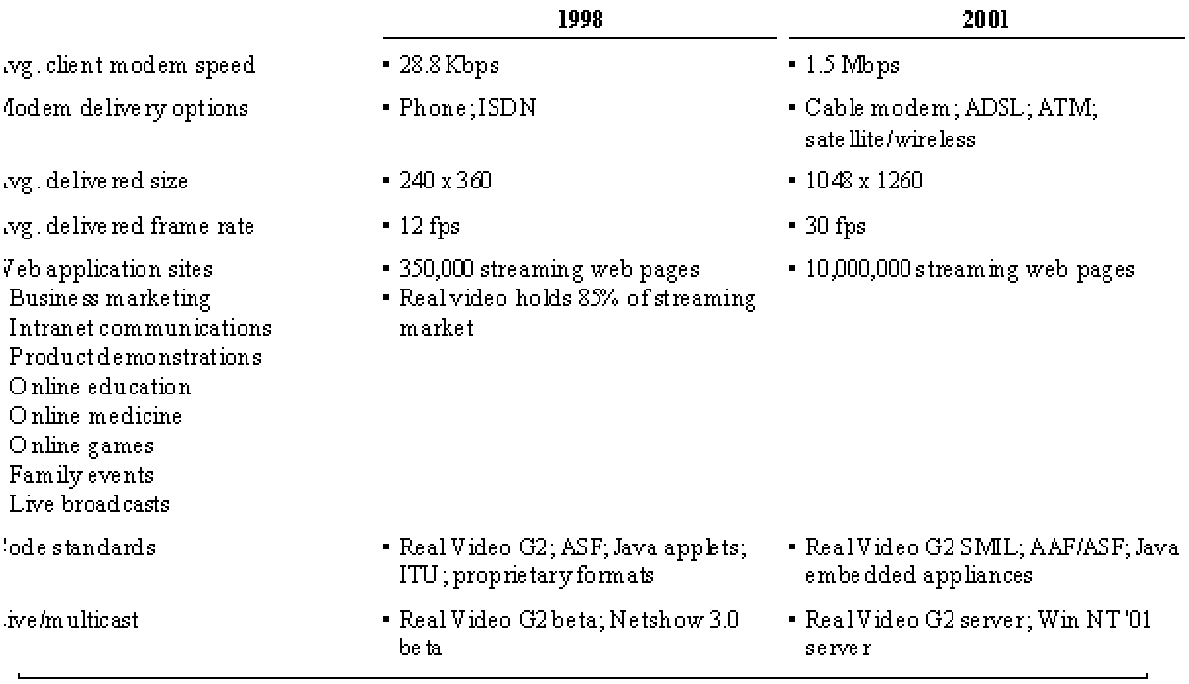
Page Owner	Streaming Video Solution Providers	Encoder	Video Server	Client Player
000 players in: ainment casting on	Real Networks	• Real Publisher; Proprietary, Fractal; Creates HTML; SMIL	• Low-end & high-end solution; HTTP streaming available	• Real Player/Plus; content/ad insertion
; rate Training	Microsoft	• Several formats including RealVideo; cumbersome	• Intelligent streaming	• Ad insertions; watermarks; banner; hyperlinks; several file formats
	VDOnet	• Wavelets; save to disk real-time while broadcast	• Scalable live/on-demand; UDP/HTTP; Clip Concatenation	• Reads VDO format; Active X
	Vosaic	• Several formats plus proprietary; encoding wizard; creates HTML; batch compression	• Works with any web server; cross-platform; full VCR control when used with Media Client	• Media client player or Java applet; dynamic adaptation to available bandwidth
	Xing	• MPEG Live Encoder	• Software solution live and on-demand. Co-exist with web/network server	• Full-screen, full motion*
	Cubic	• Wavelets; converts AVI to CVideoNow files	• Software solution over TCP/IP networks	• CVideoNow Player
	GEO	• Batch compression; scalable; full size estimation; predetermined settings	• Writes HTML; preview in browser after compression	• Java applet frame size, VCR controls
	Vivo	• VIV; ASF; supports URL flipping and real time	• Not needed	• Player or Java applet

In the context of the Internet and high-tech related industries, the streaming media market is relatively sophisticated and has undergone substantial and rapid consolidation in the past year. RealNetworks has been the clear market leader in the industry since its inception in spite of Microsoft's efforts to aggressively enter this market.

URLs with Streaming Video



Within the foreseeable near future, end user connections to the Internet will rapidly improve and Internet backbone connection speeds will have increased substantially.



	1998	2001
vg. client modem speed	▪ 28.8 Kbps	▪ 1.5 Mbps
Iodem delivery options	▪ Phone; ISDN	▪ Cable modem; ADSL; ATM; satellite/wireless
vg. delivered size	▪ 240 x 360	▪ 1048 x 1260
vg. delivered frame rate	▪ 12 fps	▪ 30 fps
Veb application sites Business marketing Intranet communications Product demonstrations Online education Online medicine Online games Family events Live broadcasts	▪ 350,000 streaming web pages ▪ Realvideo holds 85% of streaming market	▪ 10,000,000 streaming web pages
ode standards	▪ Real Video G2; ASF; Java applets; ITU; proprietary formats	▪ Real Video G2 SMIL; AAF/ASF; Java embedded appliances
ive/multicast	▪ Real Video G2 beta; Netshow 3.0 beta	▪ Real Video G2 server; Win NT '01 server

rce: Video Software Laboratory

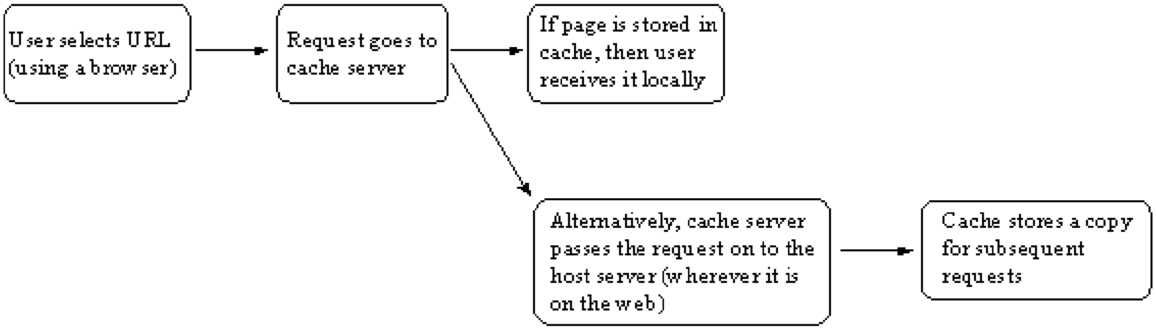
The Cache server market

Cache servers use highly robust content delivery techniques and algorithms to reduce the amount of redundant traffic over the Internet. They use RAM (random access memory) to store repetitive data locally, therefore, reducing the amount of demand on the parent server. The result is faster response times to requests, a problem mentioned earlier in the Introduction.

How cache-based servers works

A basic understanding of how computers work is essential to understanding the behaviour of cache-based servers. In computers, you have two basic types of memory: your hard drive and floppy drive, and your RAM. The hard drive or floppy drive is your long-term memory that stores your data even when your computer is turned off. RAM, on the other hand holds your most frequently accessed data because it operates faster than your hard drive. When your computer is turned off, any memory in RAM is emptied onto your hard drive. The concept of caching is to keep your most frequently accessed data in RAM so that the CPU of your computer can retrieve this data and pass it to other functions and areas of your computer.

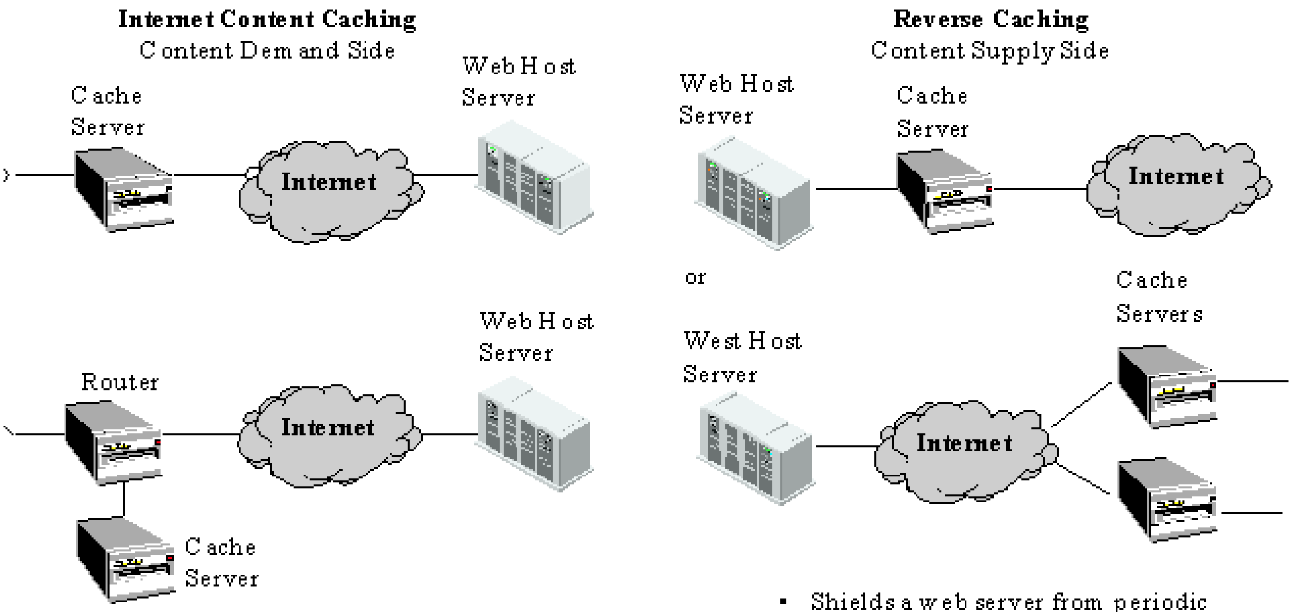
Our Afterburner server is a cache server that stores the most frequently requested files into the high speed memory areas in order to speed up content delivery to outside requests for the files. The Afterburner server serves as a bridge between the request for a file and the slower web server that offers the content. If content is requested that isn't stored at the cache-server, that request is passed further on to the web server.


User selects URL (using a browser)
Request goes to cache server
If page is stored in cache, then user receives it locally
Alternatively, cache server passes the request on to the host server (wherever it is on the web)
Cache stores a copy for subsequent requests

Current competitors

Currently, the market is growing but is estimated to be a $1 to $2 billion market by the year 2000. Existing providers (Sun, Microsoft, Cisco) are entering this market as are a number of new firms (Inktomi). All companies offer either software or appliance solutions. Software solutions are only a partial solution compared to full service appliance solutions which offer an entire integrated package of both new hardware and new software. Our Afterburner server is of the latter type.

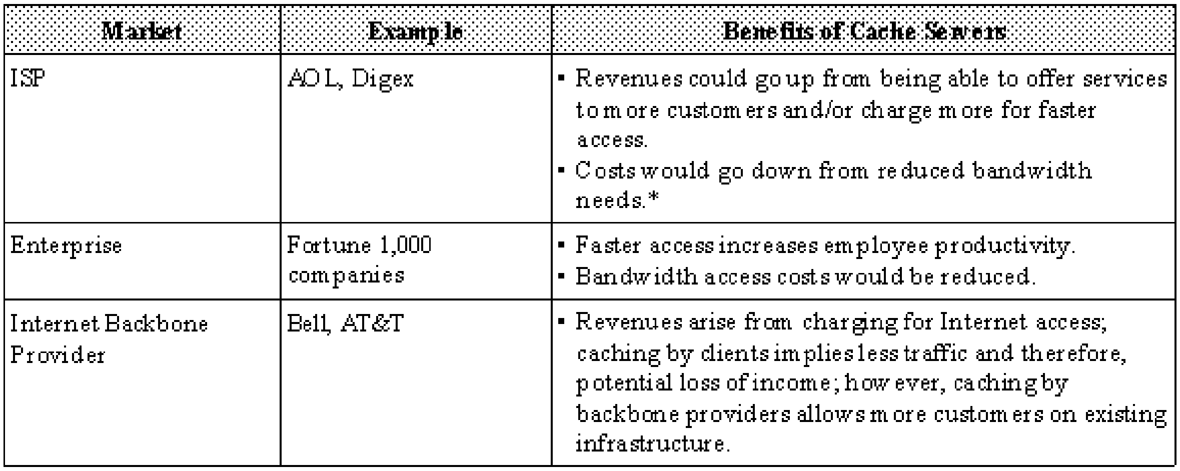
ative Approaches to Caching



- Speeds user access
- Reduces Internet backbone traffic
- Reduces bandwidth access costs

- Shields a web server from periodic overwhelming traffic demands
- Provides an isolated access point for host server security
- Reduces bandwidth and server demand costs

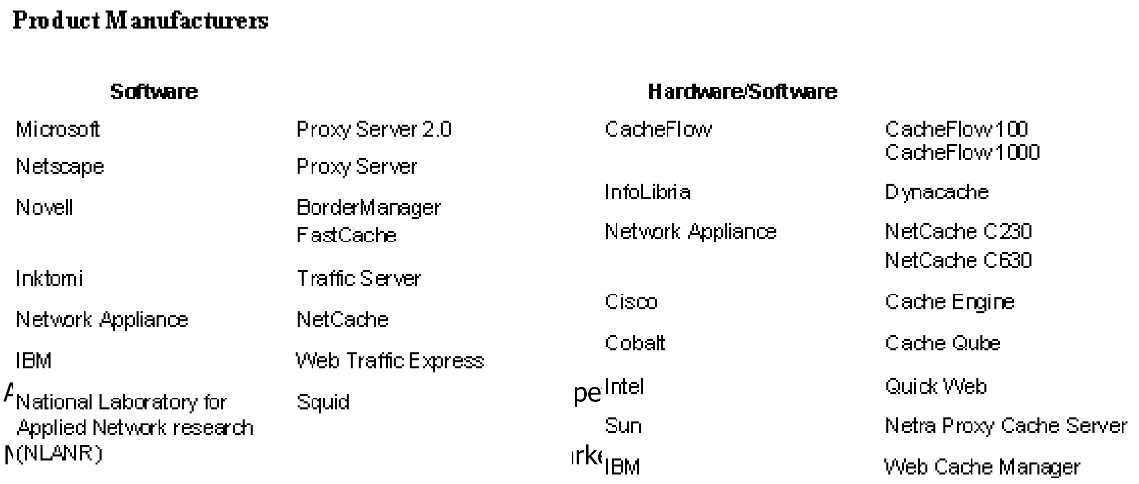
ISP's, Enterprises, and Backbone providers would benefit from the user of cache servers because of the reduced bandwidth requirements and the related benefits; lower costs, faster access, and room for more users.



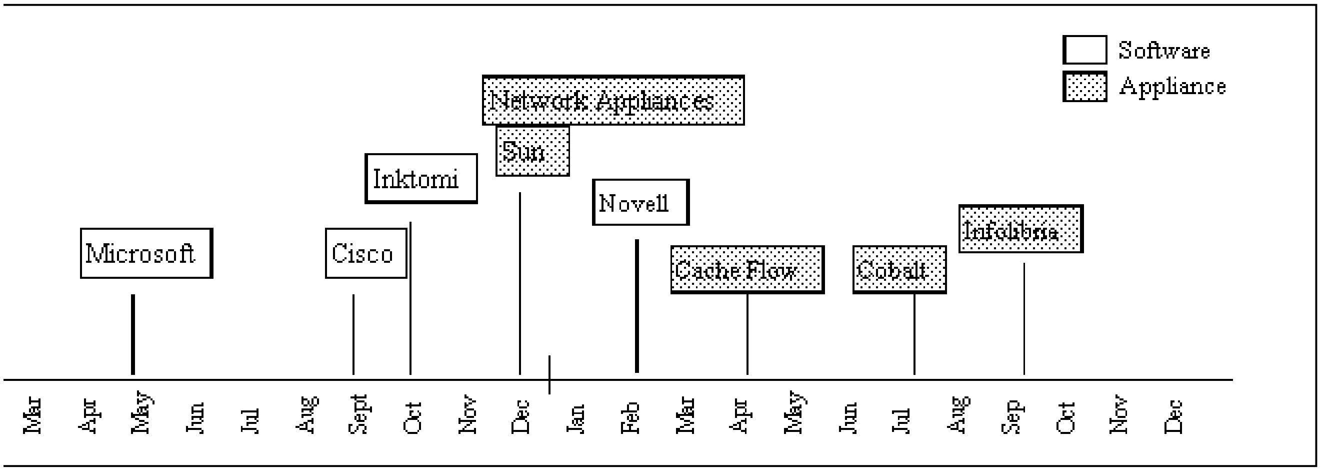
Market	Example	Benefits of Cache Servers
ISP	AOL, Digex	▪ Revenues could go up from being able to offer services to more customers and/or charge more for faster access. ▪ Costs would go down from reduced bandwidth needs.*
Enterprise	Fortune 1,000 companies	▪ Faster access increases employee productivity. ▪ Bandwidth access costs would be reduced.
Internet Backbone Provider	Bell, AT&T	▪ Revenues arise from charging for Internet access; caching by clients implies less traffic and therefore, potential loss of income; however, caching by backbone providers allows more customers on existing infrastructure.

Currently, there are a number of competitors offering cache-based servers either as a software only or as an appliance solution (embedded hardware/software)

Product Manufacturers



Software		Hardware/Software	
Microsoft	Proxy Server 2.0	CacheFlow	CacheFlow100 CacheFlow1000
Netscape	Proxy Server	InfoLibria	Dynacache
Novell	BorderManager FastCache	Network Appliance	NetCache C230 NetCache C630
Inktomi	Traffic Server	Cisco	Cache Engine
Network Appliance	NetCache	Cobalt	Cache Qube
IBM	Web Traffic Express	Intel	Quick Web
National Laboratory for Applied Network research (NLANR)	Squid	Sun	Netra Proxy Cache Server
		IBM	Web Cache Manager



Increased interest by all the above companies has not come without warrant. Various studies have indicated that backbone congestion is increasing rapidly and a large time lags between data transfers across the backbone are becoming more common. Within the last few years, many companies have joined the market to solve this problem in various ways, although focusing mainly on cache servers.

How the Afterburner server works

Introduction

Our Afterburner server runs on top of the FreeBSD operating system on a Pentium class computer. FreeBSD is a UNIX based operating system that is freeware and fully supported by volunteer programmers worldwide. FreeBSD was chosen because of its rock solid performance, reliability, scalability and ease of use and installation. Programmers familiar with UNIX will find FreeBSD just as easy to use.

A quick summary of FreeBSD obtained from the freebsd website is attached below:

- Preemptive multitasking with dynamic priority adjustment to ensure smooth and fair sharing of the computer between applications and users.
- Multiuser access means that many people can use a FreeBSD system simultaneously for a variety of things. System peripherals such as printers and tape drives are also properly SHARED BETWEEN ALL users on the system.
- Complete TCP/IP networking including SLIP, PPP, NFS and NIS support. This means that your FreeBSD machine can inter-operate easily with other systems as well act as an enterprise server, providing vital functions such as NFS (remote file access) and e-mail services or putting your organization on the Internet with WWW, ftp, routing and firewall (security) services.
- Memory protection ensures that applications (or users) cannot interfere with each other. One application crashing will not affect others in any way.
- FreeBSD is a 32-bit operating system and was designed as such from the ground up.
- Binary compatibility with many programs built for SCO, BSDI, NetBSD, Linux and 386BSD.
- Demand paged virtual memory and `merged VM/buffer cache' design efficiently satisfies applications with large appetites for memory while still maintaining interactive response to other users.
- Shared libraries (the Unix equivalent of MS-Windows DLLs) provide for efficient use of disk space and memory.
- A full complement of C, C++ and Fortran development tools. Many additional languages for advanced research and development are also available in the ports and packages collection.

Currently, many web servers spawn a new child for each request made to the server. This traditional socket programming has been passed on from the design legacy of FTP servers. Normally, this isn't a problem since the hosting machine can handle this new process adequately. However, if a web server is designed on the same principle, the machine will a) run out of memory space, and 2) the processor will be overworked to process each child process.

The Afterburner model does not spawn children. It simply scans all available connections for requests and/or readiness to receive the response.

Afterburner advantages

1. Speed - Since Afterburner does not require the huge overload of heavy or light weight multitasking and it sends all responses from preloaded files it is lightning fast. Tested performance exceeds 10 million hits a day which was extrapolated from a 30 minute test.
2. Response time - Since Afterburner uses preloaded files a typical response is much faster than under NCSA or Apache.
3. Binary Log Files - With NCSA, Apache, and most other servers, logs are in ASCII and grow in size with great speed. Afterburner log files are binary (5 bytes per hit). This technique can hold 20 times more data than other servers.

Afterburner pseudocode

```
Congifurations:

 *    DATA_DIR    Where we keep the real images
 *
 *    REF         What pages are allowed to refer to us
 *
 *    MAXCONNS    Max number of open connections (probelly should be
 *                OPEN_MAX-2)
 *
 *    MAXLINE           Our max read/write buffer (probelly should be 1/4 of
 *                the machines scoket buffer sizes)
 *
 *    MAXNAME           The longest file name allowed
 *
 *    MAXFILELEN  The longest file we can read (should be really big)
 *
 *    MAXWAIT           Max wait for request befor we close in seconds
 *
 *    DEFAULT_IMG The default image to give back if the request doesn't
 *                exist and/or we have a bad referer
 *
 *    LOG         The log file
 *
 *    REFTAB            The referer table
 */

#include <sys/types.h>
#include <sys/syslimits.h>
#include <sys/socket.h>
#include <sys/filio.h>
#include <sys/time.h>
#include <sys/stat.h>
#include <netinet/in.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <signal.h>
#include <dirent.h>

/* user configs */
#define MAXCONNS OPEN_MAX-2   /* Max number of connections */
#define MAXLINE 1024          /* Max read or write size */
#define REF ""    /* Who is allowed to ask for files */
#define MAXNAME 80            /* Max file name length */
#define MAXFILELEN 1024*1024  /* Longest file allowed -- 1 meg */
#define MAXWAIT   60          /* How many seconds to wait for request, max */
#define DEFAULT_IMG ""  /* our default image */
#define DATA_DIR "/data/afterburner/data" /* where the files really live */
#define LOG "/data/afterburner/log"       /* the log file */
#define REFTAB "reftab"       /* the reftab */

/* global vars */
      /* Last system error we got */
      /* Our log file */
      /* Did we get a SIGPIPE -- 1 if so, 0 if not */

/* the data we toss back */
      /* next file */
      /* length of this file */
      /* name of this file */
      /* the contents of this file */
      /* make it a linked list */

/* who is allowed to ask for what */
      /* next entry */
      /* file name */
      /* referer */
      /* make it a linked list */

/* keep track each connection */
      /* the file descriptor of this connection */
      /* the data that is being sent back to them */
      /* where we are in the file */
      /* the request */
      /* status -- 0 not open, 1 waiting for request
```

```
          2 sending the results */
        /* when we became avaible for reading */
        /* give us MAXCONNS slots */

main()
{
        /* number of files ready for r/w */
        /* the file desc of our listen port */
        /* dummy so listen and few other finds don't
           blow up */
        /* the highest fd for select to look for */
        /* the read and write sets for select */
        /* don't wait for timeout */
        /* the length of our last read or write */
        /* how many connections we have open */
        /* make us a daemon and decide what to do with SIGPIPE */
        /* init conn (i..e empty each slot and make it avaible for accept()) */
        /* build the reftab list */
        /* build the linked list for files */
        /* open the log file */
        /* make a socket for the listen port */
        /* set the internet parameters of the listen port */
        /* make our selves avaible for new connections */
        /* main loop */
                /* add listen port to our select list */
                /* add the right slots to the right select lists */
                /* scan all the slots */
                        /* if connected and has a good socket added it to the
                           right list */
                                /* if we have not reached MAXWAIT and it is
                                   has not gotten a request mark it as avaible
                                   for rading.  if the above is not true and
                                   it is still marked for reading close it
                                   (i.e. MAXWAIT was reached) */
                                /* if we have a request and/or in the middle
                                   of sending continue to */
                                /* if it is open and the file desc is larger
                                   then current max file desc in the select
                                   set make that file desc the same as ours */
                /* test all the sets upto for being either read or write
                /* if select gave an error close everything and start  over*/
                /* if there is a new connection and there are open slot(s)
                        /* scan all slots */
                                /* is it open, if so use it */
                                        /* accept the new connection and make
                                           its file desc equal to the systems
                                           file desc for it */
                                        /* if a valid connection make it
                                           wait for a rquest */
                                        /* no need to find an other slot */
                                        /* do not try to read or write this pass, may cause some havoc
                                        */
                                        /* scan all slots and if ready for reading or writting do it
                                        */
                        /* if ready for reading and waiting for request get it*/
                        /* try to read MAXLINE bytes from it */
                                /* if EOF of SIGPIPE close it */
                                /* figure out what they asked for and make it
                                    ready to be sent */
                                /* don't try to read and write in the same pass
                                   may be dangerous */
                        /* if the socket all the sudden became bad close it*/
                        /* if we have the request and are ready to send then do it */
                                /* if there are more then MAXLINE bytes left
                                   to be written then write them if less then
                                   write the remander of the file */
                        /* if write error or wrote the last of the file
                           or got SIGPIPE then close the connection */
                        /* move the write ptr past what we just wrote */
/* figure out what the user asked for and what page refered them */
/* what the client sent to us */
/* the file desc of the socket */

      /* zero out ref */
      /* figure out what they are asking for */
            /* isolate the next line */
            /* if it is the request store it */
            /* if it the referer store it and break */
            /* if we don;t have both and reached EOM then bail */
      /* write the log entry and do the real write every 10th hit */
      /* notify everyone we had a broken pipe */
/* clear this slot out */
/* which slot */
      /* if the connection is still open close it */
      /* clear out various status varibles */
/* read all the avaible requestable files into mem */
      /* the dir we are scanning */
      /* the current file in the dir */
      /* the file we are reading */
      /* the current file desc */
      /* how long the file really is */
      /* tmp buffer for reading the real file into */
      /* start the list */
      /* open the data dir */
      /* scan the whole dir */
            /* skip dot files */
            /* make room for the file and give it a name */
            /* open the file */
            /* read up to MAXFILELEN bytes from it */
            /* allocate the right amount of mem to this file and copy
               the contents in mem */
            /* record how long the file is */
            /* close the file */
            /* add it to our list of files */

/* create a new node in a file linked list */
      /* the new node */
      /* give enough mem and 0 out the node */
      /* tell the caller where to the find the new node */
/* place a new file at the end of the files list */
      /* the new node */
      /* the end of the old files list */
      /* the new node */
      /* find the end of the files list */
      /* add the new node */

/* find the file actually asked for */
/* name of the file asked for */
/* the file we are current testing */
      /* start at the beginning of the files list */
      /* scan the list */
      /* if it is a bad request then return DEFAULT_IMG */
/* read the reftab file */
      /* init reftab */
      /* read the whole thing in */
            /* make a new node and populate it */
            /* if this is longer the the longest ref we have then make
               it the longest */
            /* add the node */
            /* move on */
/* verify that we got the right image and have perms to view it */
/* the cur reftab */
```

```
      /* scan the whole thing */
            /* if it is the right name and referer return 1 */
      /* no matchs return 0 */
/* get field fn from s with delims of fs */
      /* the string to scan */
/* which field */
/* field delim */
/* loop counter, field counter, return string pos */
/* the return string */
      /* scan the whole string */
            /* if a field delim ++ our field count */
            /* if the right field and not the field delim then copy the
               whole field to the return string */
      /* terminate the return string and pass it back */
```

Afterburner conceptual model

Explain how the pseudocode works at a conceptual level using diagrams showing functions and how variables are passed around. This is very much like the "Detailed Description of the Invention" according to the patent attorneys. This is the area that counts to the attorneys in that "someone skilled in the art of programming should be able to rewrite the code according to what is described here"

Benchmark results

Attach benchmark results from Mindcraft, and from Acme.com

Afterburner comparison chart

At a detailed level compare Afterburner to a web server based on a similar concept and design (perhaps NCSA and/or Apache)

What I claim is

A section that precisely describes what portion of the invention, protection is being sought for.

Appendix 1 - Brief description of competitors products

Microsoft Proxy Server 2.0[1]

Microsoft Proxy Server 2.0 delivers the three things customers demand most for their intranet/Internet: High Performance, Extensible Firewall Security, and Easy Comprehensive Management.

High performance
With version 2.0, Microsoft Proxy Server introduces distributed and hierarchal caching to deliver unbeaten scalability and performance. This enables large enterprises and ISPs to make use of the product in their most demanding locations. Content caching is moving to branch offices and to the departmental level within enterprises and to ISP Points of Presence (POPs).

Microsoft Proxy Server 2.0 delivers distributed caching using a new standards-track technology called Cache Array Routing Protocol (CARP). With CARP, Microsoft Proxy Server 2.0 provides unbeaten distributed Web caching performance and deployment flexibility.

Firewall security
Microsoft Proxy Server acts as a gateway with firewall-class security between a Local Area Network (LAN) and the Internet. Several new features have been added to Microsoft Proxy Server 2.0 to enable its use as a firewall. These features include:

- Dynamic Packet Filtering
- Multilayered security
- Alerting and Logging
- Shielding of Internal Network Addresses and Internet Server Applications
- Virtual Hosting (reverse proxy)

Plus, Microsoft Proxy Server can be used with Windows NT® Server's Routing and Remote Access Service to provide cost-effective, secure Virtual Private Networks (VPN) VPNs.

Because Microsoft Proxy Server 2.0 is an extensible firewall, customers can choose from a variety of third-party products that are built on and complement the security of Microsoft Proxy Server. Such products include virus scanning, JavaScript, and ActiveX™ filters, site blocking enhancement products, and more. In addition, because the best security policy is one that includes multiple mechanisms to provide backup and depth, Microsoft Proxy Server 2.0 can be used in a very complementary way with high-end firewall solutions to meet the specialized security needs for a wide spectrum of customers.

Easy comprehensive management
Since Microsoft Proxy Server is integrated with Windows NT Server, administrators can use a single set of tools to manage their intranet and Internet access. This provides a lower total cost of ownership. Version 2.0 introduces more ways to manage Microsoft Proxy Server with graphical interface, command line support for scripting, and Web Administration (note: Web Administration Tool is currently available as a separate download). New features such as cache arrays have tools for easy configuration. There are also new tools to automate the deployment, configuration, and backup of Microsoft Proxy Server. Plus, network managers can enjoy the additional flexibility provided by SOCKS v4 support, HTTP 1.1, and FTP caching to enable expanded use of Internet and intranet services to their users.

Netscape Proxy Server[2]

The Server for Caching and Filtering Web Content

As traffic across Internet gateways and intranet wide area networks grows at an exponential rate, so does network congestion. This presents numerous challenges to network administrators who must manage network traffic, control user access to network content, and ensure content availability and quick reponse times.

Netscape Proxy Server is powerful software for caching and filtering web content. It distributes and manages information efficiently so that network traffic and user wait times are reduced. Proxy Server also helps organizations ensure that users are securely and productively accessing network resources. Tight integration with the rest of the network infrastructure, cross-platform support, and centralized management capabilities maintain Proxy Server's low cost of ownership.

Network Performance Boost

Proxy Server's efficient caching model distributes data where users need it, reducing network traffic and requests to remote content servers. Proxy routing makes it possible for organizations to deploy Proxy Servers at branch offices and network bottlenecks to benefit from caching on intranets. Caching on-demand intelligently caches documents based on user requests. Batch updates also enable caching on-command so administrators can download documents or sites on a scheduled basis.

Now Proxy Server enhances the scalability and reliability of caching by supporting proxy arrays. This distributed caching mechanism enables multiple proxies to operate as a single logical cache for load-balancing and failover. Support for dynamic proxy routing allows Proxy Server to query other caches for document availability.

Security and Productivity Enhancement

Networks are only as strong as their weakest link, which is often the gateway. Proxy Server enhances network security by providing a control point for Internet traffic and by logging all transactions. Fine-grained controls let administrators limit access to documents or sites based on individual users, groups, IP addresses, host names, or wildcard expressions. Proxy Server also provides filtering of objectionable URLs, content including viruses and HTML tags, and content types such as ActiveX.

Proxy Server facilitates user access through the firewall. In addition to being able to tunnel protocols supported by the web proxy, organizations can use SOCKS version 5 to traverse the firewall for any protocol or application. Reverse proxying makes it possible for the Proxy Server to act as a "web server stand-in," accepting encrypted traffic on behalf of a web server protected behind a firewall.

Simplified Management

Proxy Server makes it easy for administrators to manage intelligent networks of proxy servers. Native Lightweight Directory Access Protocol (LDAP) support is now available to centralize user name and password management via an integrated Netscape Directory Server. Clustered management capabilities enable administrators to configure and maintain multiple proxies. The Automatic Proxy Configuration (APC) feature of Netscape Communicator permits modifications to the proxy infrastructure without touching client software on every desktop. Proxy Server also supports Simple Network Management Protocol (SNMP) versions 1 and 2 for monitoring server status.

Scalable and Flexible Caching

- Provides efficient, transparent caching on-demand of web documents, automatically routing requests to Proxy Server and returning current documents from the cache.
- Batch updates enable caching on-command to download documents or entire sites on a scheduled basis. Proxy Server also refreshes data in the cache at specified intervals to ensure that content is current and available for periods of heavy use.
- Allows proxy chaining for building hierarchical caches that improve performance on internal networks.
- Supports Cache Array Routing Protocol (CARP), which uses a deterministic algorithm for dividing client requests among multiple proxies.
- Supports Internet Cache Protocol (ICP) for dynamically querying neighboring caches to determine document availability.
- Minimizes network traffic resulting from "push" technologies based on Hypertext Transfer Protocol (HTTP), such as Netscape Netcaster.

Fine-Grained Filtering

- Controls access to network resources by granting or denying access based on user name and password; named groups; or IP-, DNS-, and host-based wildcard expressions.
- Filters based on requested URLs. Supports third-party plug-ins with categorized lists of sites that may be blocked by Proxy Server.
- Enables outgoing headers to be blocked to ensure privacy.
- Scans incoming HTTP and File Transfer Protocol (FTP) files for viruses with Trend Micro's built-in InterScan Virus Wall engine, and alerts administrators when a virus has been detected. Includes access to regular pattern updates.
- Provides the ability to filter HTML tags and content types such as ActiveX.
- Enables and controls user access through firewalls via SOCKS version 5, a standard that supports streaming protocols such as RealAudio.
- Tunnels Hypertext Transfer Protocol Secure (HTTPS), SNEWS, and other protocols based on Secure Sockets Layer (SSL) to facilitate encrypted communication through the firewall.
- Logs all client transactions to enable auditing of user activity, and provides analysis tools for summarizing server statistics.
- Serves as a reverse proxy (an intermediary for all clients connecting to a protected web server). Secure reverse proxying provides an additional barrier for web servers and applications behind firewalls by accepting an SSL session from the client and creating a new SSL session with the server.

Enterprise Management

- Supports LDAP-based user, group, and password management for Proxy Server authentication.
- Provides clustered configuration and management of multiple Proxy Servers.
- The Automatic Proxy Configuration feature of Netscape Communicator makes proxy configuration transparent to end users.
- Provides a consistent, cross-platform, easy-to-use administration environment through HTML forms. Encrypts communication using SSL for protected remote administration.

- Supports SNMP versions 1 and 2 for standards-based, remote monitoring and management.
- Enables administrators to tune configurations without major planning efforts or high-risk implementations. Rollback to the previous stable configuration is possible.

Novell Border Manager[3]

Novell BorderManager is the industry's first integrated family of directory-enabled network services that manages, secures, and accelerates user access to information at every network "border"--the point where any two networks meet. Through a single point of administration, it is possible to manage network security policies, protect confidential information, establish user access privileges to Internet content, and reduce WAN connectivity costs.

As organizations take advantage of the opportunities presented when Internet access is available to selected users, system managers are faced with higher bandwidth demand, increased security headaches, and a heavier system management load. Moreover, system managers must grapple with consistent security polices across the network and with concerns about loss of employee productivity due to the lures of sports sites, games, and other non-business information available through the Internet

Novell's BorderManager provides superior directory-based management, firewall security, and unmatched Web access performance for organizations that want to provide:

Secure access to Internet services from the intranet
Secure access to the intranet from remote sites
Centralized management to uniformly apply enterprise security rules both internally and externally
Linkage of geographically dispersed sites
Lower communications costs by using the Internet for voice, data, fax, and multimedia transmissions
Lower hardware costs for Web servers due to optimized performance.
BorderManager, like all Novell products, supports industry standards, which means it will accommodate clients and servers that are already installed. It is a total solution that includes:

- Centralized Network Administration
- Novell Directory Services™ (NDS™)
- User-Level Access Control (Not Just IP-Level) Services
- Firewall Services
- Packet Filtering Services
- Network Address Translation (NAT) Services
- Circuit Gateway Services
- Application Proxy Services
- Virtual Private Network Services
- Advanced Proxy Cache Services
- Full Routing and Remote Access Services
- IntranetWare (2-user) Runtime
- CyberPatrol URL Content Filtering
- Netscape Navigator™.

BorderManager enables organizations to combine best-of-breed technology in all areas, including server platforms, workstations, protocol stacks, Web browsers, and Web servers, while incorporating new technologies as they become available.

BorderManager dramatically reduces cost of ownership by:

- **Centralizing Network Administration:**
  Fewer people need to be involved in administering the network, and the "define once, apply globally" strategy reduces the chance for costly errors.
- **Protecting Hardware and Software Investments:**
  BorderManager improves overall Internet/intranet access performance, which leverages hardware investment in servers, desktop systems, and routers. BorderManager also easily integrates with existing Web servers and firewalls, making both more secure.
- **Reducing WAN Link Expenses:**
  BorderManager's proxy caching services cache Web pages locally, providing a cost-effective alternative to upgrading an

Internet connection. In addition, BorderManager's Virtual Private Network (VPN) services eliminate the expense of dedicated private lines by making the Internet the enterprise network backbone.

Novell FastCache[4]

BorderManager FastCache™ is software that accelerates users' access to information stored on Internet and intranet servers. You install FastCache on a server at your network's border (the point where your network meets other networks, such as the Internet). FastCache establishes a cache (a very high-speed block of RAM) on the server where it stores information that users request from Internet or intranet Web servers. When users request that information again, FastCache retrieves it quickly from this local cache. Because FastCache doesn't have to establish an Internet connection or go to an overburdened intranet Web server to retrieve cached information, users receive that information much more quickly—in fact, FastCache offers the industry's fastest retrieval rates for cached information.

FastCache offers proxy caching, hierarchical caching, and Web server acceleration services that not only increase the speed with which users access Web information, but also reduce your bandwidth requirements and significantly improve the performance of your Web servers.

Features

- Accelerates access to frequently used Web information
- Offers hierarchical caching
- Reduces bandwidth requirements
- Improves Web server performance
- Includes Novell's IP gateway to connect IPX™ users to the Internet
- Includes Netscape Navigator
- Is easy to install

Quick Access to Web Information

FastCache stores frequently requested Web pages in a cache, dramatically improving the rate at which users can access Web pages. When a user requests a particular document from a Web server, the request goes to FastCache, which checks its proxy cache. If the document is not in the cache, FastCache retrieves the document from the Web server hosting that document and caches the document before delivering a copy to the user. If the document is cached already, FastCache delivers the document to the user directly, rather than forwarding that request to the Web server. Consequently, users get the information they need quickly—up to ten times more quickly than they would without FastCache.

FastCache uses several methods to guarantee that users get the same up-to-date information from the cache that they would get from the Web server where that information was originally stored. For example, FastCache does not cache information that the hosting Web site manager flags as noncacheable. Instead, FastCache passes requests for noncacheable information directly to the host Web server. Further, FastCache recognizes expiration dates and times that Web site managers stamp on some information and retrieves fresh versions of expired information when users next request it. In addition, FastCache lets you set your own parameters for refreshing cached information that does not have an explicit expiration date.

Improved Access to Internet Information for Large LANs and WANs

Organizations with workgroups distributed across large local area networks (LANs) or wide area networks (WANs) will benefit from FastCache's hierarchical proxy caching. When you install FastCache on several Web servers, you can configure those servers to communicate with each other to determine whether documents missing from one cache might be present in one of the other caches within your organization. With this hierarchy of FastCache servers, you further reduce the number of requests to host intranet or Internet Web servers, thus reducing bandwidth on your Internet connection and reducing the burden on internal Web servers.

Reduce Bandwidth Requirements

You can dramatically reduce the demands on your network's Internet connections with FastCache. Organizations and workgroups tend to need the same information over and over, so users often make repetitive requests for documents. In fact, 60 to 80 percent of Web requests are for information that has been accessed recently. Because FastCache locally stores Web information the first time it is requested, FastCache can process the majority of users' requests for Web information without crossing the Internet connection. Thus, FastCache can reduce bandwidth needed on your Internet connection by up to 80 percent.

Web Server Acceleration

In addition to increasing the speed at which your internal users can access information from the Web, FastCache can also be configured to enable your Web servers to handle a much higher volume of requests from external users—without upgrading or installing additional expensive hardware. Web servers can be a bottleneck in your intranet or Internet infrastructures because they are often overloaded with requests, which produces slow response times. FastCache acts as a dedicated cache in front of the Web server and handles up to 95 percent of Web server requests directly from its own cache. It handles these requests transparently; no changes are necessary to the Web server, the browsers, or the browser clients. Therefore, FastCache can accelerate any Web server.

Inktomi TrafficServer[5]

Inktomi Traffic Server Coupled Cluster Configuration
- Supports single or multiple nodes with one or more CPUs per node
- Sun solution requires UltraSPARC systems with Solaris 2.6
- Digital solution requires Digital ALPHA UNIX 4.0D
- Silicon Graphics solution requires MIPS R10000 with IRIX 6.5
- Minimum requirement 128MB RAM, 2-50GB disks per CPU

HTTP
- Keep alive and persistent connections
- Cache control headers
- Expiration and revalidation

RTSP
- Special caching support for streaming Real Media types (audio and video)

FTP
- PASV & PORT mode

NNTP
- Cache news groups and lists
- NNTP content routing and peering

SNMP
- Configuration and monitoring

ICP
- Communication with legacy caches such as Netscape and Squid
- Peering and parenting support

Security
- Provides for multiple administrative privilege levels for different user profiles
- SSL 2.0 and SSL 3.0
- SOCKS 4.0
- HTTPS Proxying
- IP access restrictions

Cache Control
- Content fingerprinting-zero duplication of content regardless of URLs
- Multiple versions of cached objects for user and browser defined differences in content
- Never-cache, pin-in-cache, revalidate-after support
- Site or content blacklist filtering

- Negative caching
- Anonymization support
- HTTP parent proxy support
- HTTP parent fail over support
- Host name expansion
- Domain name expansion
- Content routing

Performance
- Improved performance
- DataFlow architecture supports simultaneous "store & stream"
- RAM caching
- System overload detection and throttling
- Web-object optimized raw disk object database
- Graceful degradation under high loads

Fault Tolerance
- Cluster self-monitoring
- Automatic virtual IP failover
- Automatic fast system restart

Logging Options
- Predefined logging formats including Netscape Extended and Extended 2, and Squid
- User-defined log formatting
- Logging port
- Fully configurable statistical sampling of logs

Transparency
- Multiple transparency solutions to fit specific customer needs: software router within Traffic Server or Layer 4 switching
- Ability to detect, learn, and adapt to problems encountered during transparent interception

Reverse Proxy Mapping
- Support for multiple host sites
- Designed to automatically serve fresh content or pin content in cache
- Efficient reproduction of most frequently accessed pages
- Support for URL rewriting to redirect queries to mirror sites
- Robust & flexible logging support for billing system integration

DNS/Host Information Database
- DNS BIND caching
- Host HTTP server version
- Host access frequency for keep-alive information

Network Appliance Netcache[6]

why is ours better:
describe other's faults:
detailed disclosure:
- we did this
- data flow diagrams, pictures, flow charts
- high level to low level
- pseudocode

[1] From Microsoft website

[2] From Netscape website

[3] From Novell website

[4] From Novell website

[5] From Inktomi website

[6] from Network Appliance website