Designing a Web Caching Infrastructure for Your Network

Joe Cooper

<joe@swelltech.com>

Copyright © 2000, 2001, 2002 Joe Cooper

2 May 2002

Revision History

Revision v0.01	7 Aug 2000	rjc
First working draft. Still very unfinished, but worth a read.		
Revision v0.02	2 May 2002	rjc
Second draft. Still unfinished, but worth a read.		
Revision v0.03	12 Nov 2002	rjc
XML Conversion. Minor additions and corrections.		

A complete guide to choosing a correctly sized web cache for your network bandwidth needs and the factors involved in integrating caching into your network. Answers common questions, such as how many dialup users a given cache can support, and how much bandwidth can be saved. Also, provides a general overview of the types of topologies commonly employed when using one or more web caches.

Table of Contents

Introduction 1
- Disclaimer 2

So You've Decided to Install a Web Cache 2
Internet Bandwidth Factors 2
- A Simple Formula 2
- Dedicated Land Lines 3
- Shared Land Lines 4
- Satellite Uplinks 4

Which Cache is Right for My Network? 4
Client Load Factors 5
- Maximum Open Connections 5
- Available User Ports 5
- Conclusion 5
- Explanation of the Benchmark Summary 6

Deciding Disk Capacity 7
- When Should Much Bigger Disks Be Used 7

Network Caching Topologies 8
Traditional Proxying 8
Interception or Transparent Proxying 8
- Port Redirection 9
- WCCP and Transparency 9
- Alteon L4 Switches and Transparency 12
- TransBridge Transparent Bridging Proxy 14

Introduction

Adding web caching to a network is an excellent way to boost performance and save bandwidth. But deciding how to implement web caching for your environment may cause great confusion. There are countless factors that can af-

fect what level of performance your web cache will need to exhibit to work effectively in your network.

The number of users, the types and bandwidth of incoming and outgoing web traffic, the type of routing and load balancing in place, and the use of pre-fetching satellite services, can all impact the demand placed on your web cache.

This document attempts to explain and quantify what one can expect from a web cache when these variables are known. This document does not replace discussing your needs with someone knowledgable about web caching issues. Feel free to contact the Swell support team with any questions you may have.

Disclaimer

The information contained in this document reflects and quantifies an ever changing reality. Networks get more complex every day, and more complexity leads to heavier loads on the equipment that runs them. If you have any doubts about the veracity of the information contained herein, please feel free to contact us. Questions, comments, and corrections are always welcome.

So You've Decided to Install a Web Cache

You know you need a web cache. You have read all the articles and heard about all of the benefits. The problem is, how big and fast does your cache need to be? And how do you fit it into your existing network with no disturbance of existing service? Luckily, the answer to those questions is contained in the document you are now reading. This document doesn't exist to tell you what Swell Technology web caches you need to buy. It tells you how fast and how big your cache--from any vendor--needs to be, and how that cache can be integrated into your network. If you happen to use that information to help you choose a Swell Technology web cache, then I think that is just grand. If, on the other hand you use it to build your own Squid-based web cache running the RPMS that we provide on our website, well...I think that is just grand also.

To start, this document will discuss what kind of load specific types of networks will put on a web cache. Calculating the expected load is the first step to deciding what to buy and how to install it into your existing network.

Internet Bandwidth Factors

Internet bandwidth is probably the single most important factor in deciding on the appropriate size and number of caches and the type of topology to use when implementing your cache(s). It is also one of the simplest variables to quantify. Different types of bandwidth will put different loads on your web cache. The three most popular forms of cached bandwidth are traditional dedicated land lines, broadband cable and DSL, and satellite links.

It should first be made clear that the bandwidth handling capabilities of a web cache are quite arbitrary. Real network workloads, and web traffic patterns are practically infinitely complex. However, there are ways to test a cache for bandwidth handling capabilities. Swell Technology is a strong supporter of the efforts underway by such companies as The Measurement Factory to use reliable, open source, industry-accepted, methods to measure the bandwidth handling capabilities of web caches. These numbers can be an excellent starting point in figuring out what cache will be right for your workloads.

In an attempt to make our caches as directly comparable as possible to other vendors offerings and to those of individual construction, Swell makes public the bandwidth capabilities through public participation in the Measurement Factory cache-off events every year. These benchmarks are not perfect real world examples. And the real world tends to be harder on a web cache than these benchmarks, for many varied reasons. Nonetheless, I use those numbers as a point of reference throughout this document. And when referring to the "size" of a cache in Mbps, it is referring to a particular cache performance in a Polygraph benchmark. For example, if there is a description of a network requiring 5Mbps throughput, then I mean a cache capable of 5Mbps sustained throughput on a Polygraph Polymix-4 benchmark. It would be impossible to accurately put these numbers onto caches without such a standard benchmarking suite.

A Simple Formula

Barring all other factors, there is a simple formula for calculating the throughput needs of a web cache. Assuming an ideal network, with low latency, fast client connections, and fixed internet bandwidth, it is simply a matter of taking some known values and putting them into this formula.

This formula doesn't relate directly to real networks, but it can be used to estimate the needs of your network quickly. The other factors discussed later should be used to determine the exact needs of your environment.

This simple formula requires the following information:

- Internet Link Bandwidth in Mbps
- Number of Caches to be installed

Contrary to popular belief, the number of clients accessing the cache only has a peripheral effect on the load put on the web cache. The most important single factor is how much data will traverse the cache during peak traffic times, and an almost absolute limit is placed on that traffic by the amount of bandwidth available.

Now, given the internet bandwidth in Mbps, we can calculate the peak data flow possible when the cache has been added to the network, because it is known that a web cache can provide an additional "free" bandwidth of about 30% to 50%. For our model we will assume the maximum 50% as it is better to overbuild your cache infrastructure and have it perform extra quickly than to underbuild it and have it fall down under the strain.

Given that knowledge let's calculate a base cache size, using an imaginary network with a 4.5Mbps internet connection. 4.5Mbps is the equivelent of 3 T1 land lines, and a quite common bandwidth for small ISPs. Adding the 50% possible bandwidth saved gives us 6.75 as the normal throughput that will be available to clients and thus required of the cache. However, with a web cache there will be times when some site will prove extremely popular for a short period of time. This can account for a brief spike in bandwidth saved...but also a spike in the bandwidth demanded by clients. So to account for that we will want to add a small additional amount of throughput capability. Ten percent should be enough.

```
4.5 Mbps Internet Bandwidth x 50% = 2.25

4.5 + 2.25 = 6.75 Average throughput available

6.75 + .675 = ~7.5Mbps peak throughput needed
```

As stated above, this is *not* the whole picture. Your actual network environment will also play an important role in how powerful your cache will need to be.

Dedicated Land Lines

Dedicated leased land lines are currently the most common type of internet connection in use in ISPs, universities, and corporations. This type of line includes T1, T3, and the digital DS3, or OC3, etc. The medium used doesn't matter in the case of web caching. They all provide a similar bandwidth and latency environment from the perspectve of the cache.

Sizing a web cache for a dedicated land line is probably the most straight-forward, as it is very much like being on a local network, though slower and more expensive. The latency involved is very low and usually measured in fractions of seconds, and the bandwidth is fixed. Another factor when building a caching infrastructure is reliability. Dedicated leased land lines also provide the highest level of reliability of all currently available options. So reliability is easier to obtain, usually requiring less redundancy of hardware and network links.

Given the low latency and fixed bandwidth of dedicated leased lines, calculating required cache throughput is simple. Just use the formula provided in the first section of this chapter, and add a little bit on top. The extra added on top depends greatly on your specific network. If you know that the majority of your traffic is web traffic then the demands on the cache will be higher. If you have a very focused user base, such as a university or a business, and you

believe the cache will be more effective at providing the most popular data (i.e. you think your hit ratio will be higher than average), then your cache demands will be higher. Finally, if mixed mediums are in use, and you also have a satellite link or shared access land line, you'll need to read those sections below and account for them in your sizing and infrastructure decisions.

Shared Land Lines

The designation of shared land lines is used here to represent any network media where bandwidth is not technologically guaranteed, and may changed based on traffic being generated by other sites. This usually includes shared leased lines (like T1s, T3s, etc.), DSL, cable, and wireless. While it is true that many DSL and cable providers will "guarantee" a certain bandwidth, the actual bandwidth from these media will not be completely stable. Also, these media generally exhibit longer latency times than dedicated land lines.

Because of this higher latency, these types of lines are generally slightly more demanding on a web cache than a dedicated land line. You should plan an extra 10-20% possible throughput for your cache. Also, you should plan for more simultaneous connections in such a network, as the connections may take longer and are more likely to stall due to factors outside of your local network. If you are buying a pre-built web cache, your vendor will likely have recommendations for you. Using Squid under Linux (such as on a Swell Tsunami unit), these factors may require you to alter a few settings, including file descriptor limits and available user ports.

Satellite Uplinks

Another common type of internet connection, that is growing in popularity daily is the satellite link. Satellites generally provide quite high bandwidth at very low costs. However, satellite links are also subject to high latency, making them less than ideal for web traffic. Effective web caching, and intelligently balancing network traffic between satellite and land lines, can lead to a very fast and effective system, however.

When sizing a web cache for a satellite link, the cache will need to be significantly more capable than one in use on land lines. The reasons for this are many. The high latency of satellites means that your cache must support a large number of simultaneous connections. The possibility of extreme traffic spikes and very high throughput once transmission has begun, requires the web cache to handle extreme throughput peaks gracefully. Finally, to implement caching for a satellite system or a balanced satellite and land line system, one will need to maximize cache hit ratio in order to speed the end users perceived web reponse.

If using a combination of satellite and land line connectivity, it may also be appropriate to implement QoS queueing and routing in order to maximize the odds that services requiring quick response times will be retrieved from the land line while those requiring high throughput will be retrieved from the satellite link. For example, because newsgroups are a non-realtime medium anyway, it may be appropriate to utilize satellite for the network's newsgroup feed. Web users on the other hand expect quick response for their requests, so preferring the land line when bandwidth permits is wise for web traffic. Finally, FTP generally does not need quick response time, and large files are often transmitted in either direction, so it is a prime candidate for satellite transmission.

Which Cache is Right for My Network?

Swell has attempted to provide an easy to understand method of rating the Tusnami line of web caching proxy appliances. Each web cache has been tested using an industry standard cache benchmark suite known as Polygraph. This benchmark makes it possible to accurately guage a web caches load handling abilities. Swell uses these numbers to assign a network capability rating to each Tsunami model. For example, the 20s is rated for a network of 10Mbps, while a 12i is rated for up to three T1 lines (4.5Mbps). Both can be upgraded to support higher bandwidths.

Generally, these numbers are enough to guide a cache deployment decision. However, it is likely that the designator may be innaccurate for your network. You may have a complex network topology including intermittent satellite bandwidth, flexible T1 bandwidth, cable or other shared media, etc. In such a case, Swell also provides a throughput rating for each cache. The throughput rating is a very conservative estimate of the amount of data throughput in Mbps a particular model can handle. It will usually be possible to get a pretty clear idea of your network caching needs from this information. One must also consider the disk size and estimated capacity in days for a cache.

Client Load Factors

Now that I've explained that the most important factor, by far, when sizing a cache is the available network bandwidth, I'm going to contradict myself and tell you that the type and number of clients can also have a large impact on the load on your cache. The type of load, however, is somewhat different and in the case of a Swell Tsunami an entirely configurable option. This means that any Tsunami, if it is capable of handling the bandwidth load, will be able to be tuned for the number and type of clients you have.

Maximum Open Connections

The most common concern is ensuring the system is tuned to maintain enough open disk and network file handles, called file descriptors, for your client load. For example, nine hundred simultaneous dialup users will usually keep about 1500 file descriptors open on a web cache. This is well beyond the limit of open connections on most systems that are not tuned for high load network service. To give a real world example, Linux defaults to a limit of 1024, of which only about 900 are usable for Squid or some other cache software. Swell Tsunami units, however, default to 8192 possible open connections. This is more than enough for any load that a Tsunami will encounter.

The type of client has an impact on the number of connections needed, obviously. Assuming equal bandwidth usage, slower connections require more connections, as each individual connection must stay open longer. So ten ideal 56K dialup users will require more file descriptorss than one 512K DSL user assuming an equal number of requests is being made and the same bandwidth is being consumed.

Further, networking standards dictate that connections enter a closing state that lasts at least 60 seconds beyond the last packet sent to that connection. Thus, for every open connection, a single client may have several already completed requests with connections waiting to be closed. This closing state tends to minimize, somewhat, the disparity between slow and fast clients. But the speed of the client does have an impact on the number of open connections.

Available User Ports

Much like the file descriptors discussed above, a high load web service system must have a large pool of ports available for usage. As with file descriptors, most systems do not have enough open ports by default. A Swell Tsunami is configured to allow the complete port range from 1024 through 32768. This range should be more than ample for any environment and network load that a Tsunami will be expected to handle.

Conclusion

In conclusion, a Swell Tsunami will handle any load that it is specced for. Other caches on other operating systems may require some tuning to provide service for high volumes of some types of clients. So you see I was mostly accurate in the beginning when I explained that internet bandwidth availability does dictate the required size and speed of your web cache.

All that's left with regard to sizing a web cache is to discuss how web caches can be compared from a purely performance standpoint. In the past, it was quite difficult to accurately compare web caches between manufacturers. Many vendors did not publish any benchmarks, and the ones that did chose to use their own proprietary and closed source benchmark tools. Not surprisingly, every vendor had the best price performance available according to their own benchmarking suite results. Thankfully for cache buyers of the world, the IRCache group (the folks who created Squid and the earlier Harvest project) created an open benchmarking suite for web caches, called polygraph. Polygraph was good enough to convince many caching vendors to start using it instead of, or in addition to, their own in-house benchmarks.

The impetus for creating Polygraph to start with was in order to have a 'race' for caching systems, now known as a Web Cacheoff. The creators of polygraph have now formed a company devoted to benchmarking web caches, and other content distribution systems, called the *Measurement Factory* [http://www.measurement-factory.com]. Swell Technology is a strong believer in open benchmarking of web caching systems, so that potential web cache buyers can compare systems directly and fairly to one another. Performance is not the only factor when deciding on a web cache, but it is definitely an important factor. So far we have participated in the last three of four web cacheoffs to be held, and we look forward to taking part in future events.

Using the information contained in Polygraph benchmark results from cacheoffs and independent tests is quite easy. The most commonly compared metric is the reqs/second, or the request throughput, of a cache. This field is simply a measure of how many requests were fulfilled per second by the web cache during a peak period in a test. This is often the metric touted by OEM's, and there has been a tendency to accept lower performance in other areas, like hit ratio and response time, in order to boost the request rate.

It is probably easiest to explain Polygraph report analysis through an example. Our example will be taken from the executive report of a benchmark of a Swell Tsunami 1000. This is one of Swells recently discontinued midrange systems, designed for a two T1 network (the 1000 series has been replaced by the 12i caching appliance, which has a faster CPU, larger disks, and more RAM than its predecessor).

Table 1. Executive summary

Throughput:	100.03 rep/sec
Response time:	1473.69 msec
- misses:	3044.69 msec
- hits:	194.60 msec
Hit Ratio:	55.12%
Errors:	0.00%
Duration:	4.00 hour
Phases:	top2

Explanation of the Benchmark Summary

The numbers above are mostly self-explanatory, but interpretting them and comparing them among vendors can be difficult, as mentioned earlier.

The first value above, *Throughput*, is the sustained request rate attained by the cache, in this case a base Tsunami 1000 with the standard RAM and hard disk capacity. Roughly 100 requests per second translates to 6000 web objects transferred (both from the cache and the origin servers) per minute. The web objects have an average size of about 11 kilobytes. In the case of a Tsunami 1000 this request rate translates to a requests per thousand dollars of approximately 41, meaning that a Tsunami provides 41 requests per second for every thousand dollars invested. Higher end Tsunami models provide better price performance, while lower end Tsunamis provide slightly lower price performance.

Next comes *Response time*, starting with the mean or average, then the response time for cache misses and finally cache hits. An average response time of less than 2 seconds, or 2000msec, is considered good for the Polymix-2 workload used in this test. Some vendors will accept slower response times in order to boost the request rate. Misses are less important,as they are generally dictated by the speed of the origin server rather than the cache. Hits, on the other hand, come only from the cache and thus are an indicator of how the cache holds up under its rated load. This is often an area in which some cache vendors will accept poor performance in order to artifically boost request rate.

We now come to *Hit Ratio*, which is likely the most important value after request rate. In the case of the polymix-2 workload used for this benchmark, 55% is the ideal hit ratio. No better hit rate is possible. As with *Response time* the *Hit Ratio* is often sacrificed by some vendors to achieve higher request rates. It is very easy to achieve much higher request rates by allowing the cache to drop hits when the system gets overloaded. In the case of a Tsunami it is usually possible to boost request rate by 10-15% for every 10% that is allowed to drop from the hit ratio. So the Tsunami in use here could provide approximately 110 reqs/sec at 45%, or 121 reqs/second at 35% *Hit Ratio*. One should likely be leery of a web cache that has been benchmarked but has proven not to cache very much! It generally means the web cache being tested was heavily overloaded and will not reliably and effectively perform at the request rate specified. When the hit ratio is significantly lower than the offered hit ratio, it is difficult, if not impossible to estimate the level of performance it can maintain while actually providing a good hit ratio.

The remaining three values are quite simple and do not need much explanation. *Errors* is the percentage of requests that resulted in errors. In this test, there were only three errors, so they do not even show up in the percentages. The percentage allowed by cacheoff rules was 3 percent. Clearly the Tsunami had no problem with errors on this benchmark. *Duration* is the length of the test phase that is represented by the summary. The complete test was 14 hours plus fill time, but the portion that was used for the test was only four hours long. The *Phase*, then, is the name of the period which the data represents. In this case, as in all of the second cacheoff reports, the phase is "top2", which is the second peak period during the test.

Deciding Disk Capacity

Choosing the correct disk capacity for your network load is necessary for your cache to return a good hit ratio. The ideal is usually to provide enough disk capacity for your disk to store about three to five days worth of average daily traffic. Anything beyond that is probably wasted space, as objects in the cache will eventually expire anyway, and the RAM required to index an oversized cache store is an unnecessary expense. Anything less, however, leads to less than ideal hit rates and reduced bandwidth savings.

Disk capacity is more a matter of balancing the two ends than choosing the biggest cache you can afford. The simple fact is that there are limits, very firm limits, to how much bandwidth can be saved through web caching. A web cache storing more than three to five days of average daily web traffic will generally not see notable improvement by even doubling the cache capacity. Thus, calculating the amount of disk storage needed to provide storage for that range of traffic is all that is needed to decide the required disk size. That being said, there are circumstances in which it is preferrable to have large quantities of possibly stale data, over not having a copy of the data at all. Those situations are discussed more thoroughly below.

The simplest way to decide on the idea cache storage size for your network is to estimate the amount of cacheable web content that passes through your network everyday. Because there are so many factors it is difficult, if not impossible to estimate what this value will be for your network. However, from our experience here at Swell with Tsunami units in varying network sizes, from the smallest sub-T1 to quite busy multi-T1 networks, I've come up with some pretty good estimations. I welcome comments from folks who have had significantly differing experiences than what I describe here, and I welcome descriptions of networks that do not follow this trend. It is usually quite safe to err slightly to either side of these figures by a small margin.

Table 2. Network Examples and Their Disk and RAM Capacity Needs

Network Bandwidth	Recommended Cache Storage Size	Recommended RAM Size
DSL 512Kbps	4 GB	128MB
T1 1.5Mbps	8 GB	192 MB
Dual T1 3Mbps	16 GB	256 MB
Three T1 4.5Mbps	24 GB	384 MB
Four T1 6Mbps	30 GB	512 MB
Five T1 7.5 Mbps	36 GB	1024 MB

Take particular note of the correlation between suggested RAM size and suggested disk size. While these numbers are pretty specific to Squid based caches like the Tsunami series, it is true across all platforms that web caching is a very RAM intensive application. Having too little memory in a web cache can cripple its performance, and in the case of Squid the memory demands on the system are directly proportional to the cache storage size. If you do wish to expand the cache storage size, in order to further increase hit ratio, be very careful to expand RAM in equal proportion. Also be prepared for the law of diminishing returns to rear its ugly head very severely soon after disk capacity surpasses that recommended here. Having larger disk capacity won't cause harm, as long as you have ample RAM to keep the index and maintain an effectively large in-memory hot object cache.

When *Should* Much Bigger Disks Be Used

As mentioned above, there are times when bigger disks and sometimes *much* bigger disks are a good idea. Those will be covered in more detail here. The most important factor is when bandwidth is so much more expensive than disks, or is simply not available for any price, it is a good idea to have more than "enough" cache storage.

In the continental United States, it is difficult to imagine bandwidth being so expensive that doubling ones disk and memory capacity to gain three to five percent in hit ratio is not a poor economic choice. But in many parts of the world, where bandwidth is considerably more expensive, those few percent soon add up to far more than the several hundred or several thousand dollars spent upgrading a cache to support twice the disk storage and twice the RAM.

Additionally, there are locations where bandwidth is only available at certain times. In remote locations, on ships, in the Arctic, etc. it is often impossible to maintain steady network connectivity. There may be hours when the network satellite link is offline. Thus, a very large web cache can mean the difference between possibly stale internet content, and no internet content at all. These also generally have far more expensive bandwidth than in the United States. Thus network administrators in these situations must choose the largest cache possible in order to provide a reasonable level of service from unreasonable conditions. A large Squid-based web cache would be ideal for such a circumstance, given its ability to operate in an offline Mode, allowing all cached data to be accessible while no outside accesses would even be attempted.

In these, and possibly other circumstances not covered here, it is wise to choose a much larger cache than would generally be recommended. Sixty or seventy gigabytes of cache storage and 2 GB of RAM in a low speed cache designed for an intermittent 512Kbps satellite link on a cruise ship would not be unrealistic or overspecced. These types of situation are an area of particular interest and development at Swell, and future features of the Tsunami line will provide improved service in these situations.

Network Caching Topologies

Web caches are integrated into networks in a wide variety of ways. All of these are derivatives upon the three common web cache modes: transparent, traditional proxy, and reverse proxy. Of the three, the first two are used for caching the web at large for local clients, while reverse proxying is used for caching one or more local websites in order to provide enhanced security or performance for those sites. All three types of web caching are supported by Swell Tsunami systems, and any other Squid based cache.

Your first decision when implementing a web cache is which type of web caching is needed. It is usually a simple and obvious choice. Many environments require transparency. Internet service providers, universities, and some large businesses may consider it necessary to provide web caching as inobtrusively as possible. Traditional proxying is preferred when transparency is not required, as additional features are available that are not possible in any transparent configuration. Finally, reverse proxying is used in very specific circumstances, wherein a web server is to be accelerated or made more secure through the use of a caching proxy.

Traditional Proxying

Traditional proxying is the simplest to implement, but can potentially be the most time consuming for the administrator in a large network. Each client browser must be individually configured to use the web cache for requests. In an environment of relatively literate users this problem can easily be overcome by simply redirecting all traffic on port 80 to a local webserver. The local webserver will then serve a single page or set of pages with detailed instructions for configuring the browsers in use on a given network. The users will then be forced to configure their browsers to use the cache if they wish to access the internet. Configuration only needs to be performed once per client system, after which the interaction with the proxy is invisible to the user.

In most business and school environments, a traditional proxy is the best choice. If user level authentication is to be used, traditional proxying must be used. Transparency in any web caching environment precludes use of proxy level authentication.

Interception or Transparent Proxying

Interception proxying, also known as transparent proxying, is usually the preferred mode of operation for Internet service providers. Clients do not need to perform any special configuration on their caches to gain the benefits of the

cache. There are a large number of methods for achieving transparency, many of which will be covered in some detail here. After reading this section you will likely have a good overview understanding of some of the options for providing transparent caching service.

The most common methods of achieving transparent caching include port redirection, WCCP redirection, and L4 switch redirection. Each will be briefly covered here. Port redirection is often available within the cache system itself. As in the case of Swell Tsunami units, the OS provides for redirection of port 80 traffic to the web cache port. Most free Unix-like systems do provide some means of port redirection. WCCP is a feature of many modern Cisco routers and is capable of providing balancing of multiple caches, in addition to providing transparency and reasonable failover features. L4 switches are the newest method of providing transparency and they also provide very effective load balancing and usually some form of failure protection. There exists new, enhanced switches called L7 switches that are able to operate on the highest level of request information (the application layer). Use of these switches is usually quite similar to L4 switches, and thus is discussed concurrently with L4 switching. As the technology advances and L7 switches begin to be used in more complex ways this document will be updated to reflect those changes, and L7 switching may receive an independent section.

Port Redirection

The fundamental building block of all of the other types of transparency, port redirection, is just what it sounds like. A given port, in this case port 80, is redirected to some other port, in this case the web cache port. When using Linux as the host OS for the cache system (as on Tsunami caching proxies), this redirection is achieved via use of ipchains, the kernel packet filter and forwarding tool. Other systems use different utilities but most can accomplish it in some way.

Port redirection can be used by itself to achieve transparency. One simply routes all traffic, or just all web traffic, to the cache system. The cache system then handles redirecting web traffic on port 80 to the web cache software. Using Linux, one would use a command something like this, to set up port redirection for use with a web cache on port 3128:

```
# /sbin/ipchains -A input -s 192.168.1.0/24 -d 0/0 80 -p tcp -j REDIRECT 3128
```

This rule simply takes all network traffic on port 80 from the 192.168.1.0/24 network destined for anywhere, 0/0, and redirects it to port 3128. In a small network, this is all that is really needed. Failure protection can be provided via a simple script that calls *echoping* or something similar and modifies the router configuration to bypass the cache if it fails.

Often, the other methods discussed below will still require some form of port redirection in order to perform corrcetly. WCCP and L4 switch transparency methods can be viewed as optional enhancements to simple transparent operation, allowing for better fault management, load balancing among multiple caches, or increased routing flexibility.

WCCP and Transparency

Routers that support the WCCP protocol may be used to provide transparent web cache operation, as well as fault tolerance and load balancing of multiple caches. While routers supporting WCCP are quite expensive, it is often the case that a WCCP enabled router is already in place on the network, and if it is not already under a heavy load, it is probably a good choice for use with one or more web caches.

WCCP uses an interesting method for directing traffic to the web cache. Each packet destined for the cache is encapsulated in a tunneling packet bound for the web cache. The web cache is then expected to unencapsulate the data and act correctly on what it finds. With a Squid-based web cache running on Linux, like a Tsunami, the encapsulated packet is handled by either a custom ip_wccp module, or a modified ip_gre module. The module will unencapsulate the tunneled packet as soon as it arrives from the router, and then port redirection must again take place to direct those packets to the cache port.

Implementing WCCP is somewhat more complex than using simple port redirection, but usually only takes a few more minutes assuming your system is capable of using WCCP. Familiarity with your version of IOS is recommended, as commands may differ somewhat between versions. Since the version of IOS on our router in 12.0(9) Enterprise+, that is the version I will cover here. It is quite likely that you will be using the T or S train of IOS, which work similarly, but may require additional WCCP configuration. Check with your router vendor for more information on using WCCP functionality if this guide does not work for you. The important things to note when dealing with Squid based web caches, such as the Swell Tsunami line, is that currently only v1 of WCCP is supported. Version 2 support will come at a later date and this document will then be updated to reflect that.

If using Squid and Linux, you must either have a patched ip_gre module in your kernel, or you must compile and use the ip_wccp module. Also, your Squid must be one of the versions that supports WCCP. This support began in 2.3 but a patch is available to provide this support in the 2.2 series. Other operatings systems are not covered in this document currently. Information regarding other OS' is welcome. Tsunami units have a patched ip_gre module included, and a WCCP enabled Squid. First I'll cover configuring the router and then Squid and OS configuration will be covered.

WCCP Router Configuration

Thanks to Harwant Pannu of Prominet, Inc. for this pointer.

In many environments, the following four steps will handle the router side of configuration:

```
RouterName(config)#ip wccp version 1
RouterName(config)#ip wccp web-cache

RouterName(config)#interface ethernet0
RouterName(config-if)#ip wccp web-cache redirect out
```

That's it! Easy enough. If that doesn't do it, check out the more complicated configuration notes below, or contact your support representative.

WCCP Router Configuration, the complicated way

Thanks to Jan Haluza for his tutorial regarding this subject, from which much of this section is drawn.

The following information is a slightly more complex variant of configuring WCCP on a Cisco router. Note that these instructions have been tested on a Cisco 2501 with IOS 12.0(9) Enterprise (this is not T train, so it probably isn't what you have if you are an ISP). A rather old, and underpowered router that we have here for testing. You probably want the simpler instructions above if you have a more powerful and recent IOS.

Configuring the router requires a few simple steps. First, WCCP must be enabled within the router:

```
$enable

#conf t

#ip wccp enable
```

Then the list to redirect to WCCP must be set (adjust to suit your network):

```
#ip wccp redirect-list 1
```

Now set the interface up for redirection (again, adjust to suit your network):

```
#interface ATM1/0.3 multipoint
```

```
#ip address 192.168.1.1 255.255.255.0
#ip web-cache redirect
```

Finally, define the redirect list with the network(s) to redirect:

```
#access-list 1 permit ip 192.168.1.0 255.255.255.0 any
```

Configuring the WCCP ip_gre and ip_wccp Tunneling Interfaces

WCCP packets are *encapsulated* in order to achieve redirection to the web cache without interfering with the actual packet headers. In this way, the Squid cache actually receives the original packet that was sent to the router by the client. This allows for accurate IP logging of traffic. However, for this to work, something has to receive those packets and unencapsulate them so that Squid can understand them. Under Linux, this is done by the ip_gre module, or by the ip_wccp module. Configuring ip_gre is more complicated but provides more flexibility, and is the option availabe on Tsunami systems, so I will discuss it first.

If ip_gre is compiled as a module, then you will first need to insert the module. If you kernel has the ip_gre interface compiled in, you will not need this step:

```
# modprobe ip_gre
```

Then you will need to create a gre interface with ifconfig, or whatever your preferred method of creating an interface is. On Tsunami units, this can also be achieved in the Webmin interface. Consult the Users Guide for more information.

```
# ifconfig gre1 192.168.1.253 netmask 255.255.255.255 up
```

In the above interface, any address will do, and an internal address *(such as 192.168.1.1, 10.1.0.1, or 172.16.1.50)* will be fine. This will be the address set below in Squid for the outgoing WCCP address.

Configuring ip_wccp is even easier. Simply insert the module:

```
# modprobe ip_wccp
```

This module will then listen for any WCCP packets that arrive, and unencapsulate them, before any redirect rules see those packets. It should be noted that both of these method still require port redirection to take place. The packets will arrive on port 80 and will appear to be destined for the origin server. Without a redirect rule, they will either be directly routed, or the packets will simply stop, depending on whether routing from that source and to the given host is not disallowed by other ipchains rules. If you skipped the port redirection section please return to it now, and add the correct redirection rule for your network.

Configuring Squid for WCCP

Squid configuration is quite simple, and build on the steps discussed above for transparency via port redirection. There are only three additional WCCP options in the older Squids and four in the more recent Squids. They are:

```
wccp_router
wccp_incoming_address
wccp_outgoing_address
```

The first is self explanatory, simply set it to the address of your WCCP router. The second, **wccp_incoming_address** is used to define the address which accepts WCCP packets. This usually does not need to be set, as Squid will listen on all addresses by default. Finally, the **wccp_outgoing_address** is the address used to send "WCCP_I_AM_HERE" packets to the router. When using the ip_gre module, this should be set to the gre interface address as described in the section regarding the ip_gre interface. This is required so that the web cache that the router detects appears on the correct IP address that is capable of receiving WCCP encapsulated packets. When using the ip_wccp module this option does not need to be set as all traffic traverses the main ethernet interface.

The fourth option now available in some Squid versions is the **wccp_version** option. This is required for some old versions of IOS that need for Squid to report the version as 3 instead of the default of 4. It is rarely needed, and a source modification is required for older Squid's to make it work with these versions of IOS.

Alteon L4 Switches and Transparency

Another common method of achieving transparency and high availabity while also providing for balancing between multiple web caches is L4 switching. There are several vendors of L4 switches, and I do not have any particular recommendations to offer for which switch to buy. But a recent client invested in an Alteon CacheDirector and has been quite happy with it. He graciously answered several of my questions about its operation and also forwarded some relevant documentation, so I'm able to document this particular switch. I welcome similar information for other L4/L7 switches.

Layer 4 switching is in some ways similar to traditional port redirection, in that is doesn't require any special support from the web cache(s) the way WCCP does. Layer 4 switches will work on nearly any service, and can provide fault tolerence through a simple service ping method. In the Alteon product line these features are known as *Web Cache Redirection*, or simply *WCR*.

Setting up Squid for use with L4 redirection is the same as the steps described earlier for port redirection trasnparency, and thus need not be covered again. This section will focus on the configuration of the Alteon L4 switch. Please consult with the documentation provided by Alteon as well before embarking on this configuration, as the information here is "hearsay" since we don't have an Alteon switch for me to test this configuration on.

Configuring WCR on your Alteon

This section assumes you have release 4 of the Alteon software. Other versions can also be used, consult with your documentation if using a different version. The first step in configuring your Alteon for redirecting to one or more web caches is to define the Real Servers in the switch:

The following portion of this document has been derived from documentation for Alteon products.

```
Main# /cfg/slb/real 1 (First cache Server will be #1)
Real server 1 # rip 192.168.1.25 (IP of first cache)
Real server 1 # ena (enable 1st cache)
Real server 1 # ../real 2 (Now, we'll configure the second)
Real server 2 # rip 192.168.1.26 (IP of second cache)
Real server 2 # ena (enable 2nd cache)
```

Next we'll create a server group and include the two web caches:

```
Real server 2 # /cfg/slb/group 1 (switch to editing group 1)
Real server group 1 # add 1 (add cache 1 to group 1)
Real server group 1 # add 2 (add cache 2 to group 1)
```

Now Web Cache Redirection *port states* must be set. This defines how each port of your switch will be used. Clients may be connections to dialup servers or other network switches.

```
Real server group 1 # /cfg/slb/port 1 (edit state of port 1)
SLB port 1 # state layer2 (cache 1 is on the first port)
SLB port 1 # ../port 2 (edit state of port 2)
SLB port 2 # state layer2 (cache 2 is on the second port)
SLB port 2 # ../port 3 (edit state of port 3)
SLB port 3 # state cached (client a is on port 3)
SLB port 3 # ../port 4 (edit state of port 4)

SLB port 4 # state cached (client b is on port 4)
SLB port 4 # ../port 5 (configure state of switch physical port 5)
SLB port 5 # state cached (client c is on port 5)
SLB port 5 # ../port 6 (configure state of switch physical port 6)
SLB port 6 # state layer2 (make sure connection to router is layer2)
```

Next, create a WCR filter on the switch:

```
SLB port 6 # /cfg/slb/filt 2 (create filter 2)
Filter 2 # sip any (from any source address)
Filter 2 # dip any (to any destination address)
Filter 2 # proto tcp (for TCP protocol traffic)
Filter 2 # sport any (from any source port)
Filter 2 # dport http (to the HTTP port--80)
Filter 2 # actio redir (set the action to redirection)
Filter 2 # rport 3128 (send traffic to port 3128 on caches)
Filter 2 # group 1 (select real server group 1)
Filter 2 # ena (enable the filter)
```

Then create a default filter on the switch:

```
Filter 2 # ../filt 224 (select default filter)
Filter 224 # sip any (from any source address)
Filter 224 # dip any (to any destination address)
Filter 224 # proto any (for any protocol traffic)
Filter 224 # actio allow (set the action to allow)
Flter 224 # ena (enable the filter)
```

Assign the newly defined filters to client ports:

```
Filter 224 # ../port 3 (client a is on switch physical port 3)
SLB Port 3 # add 2 (add the WCR filter to port 3)
SLB Port 3 # add 224 (add default filter to port 3)
SLB Port 3 # filt enable (enable the filter on port 3)
SLB Port 3 # ../port 4 (client b is on switch physical port 4)
SLB Port 4 # add 2 (add the WCR filter to port 4)
SLB Port 4 # add 224 (add default filter to port 4)
SLB Port 4 # filt enable (enable the filter on port 4)
SLB Port 4 # ../port 5 (client c is on switch physical port 5)
SLB Port 5 # add 2 (add the WCR filter to port 5)
SLB Port 5 # add 224 (add default filter to port 5)
SLB Port 5 # filt enable (enable the filter on port 5)
```

Finally, enable the WCR setup on the switch:

```
SLB port 5 # .. (select Server Load Balancing menu)
Server Load Balancing # on (enable L4 processing)
Server Load Balancing # apply (make changes active)
Server Load Balancing # cur (view current settings)
Server Load Balancing # save (save settings for restore after reboot)
```

TransBridge Transparent Bridging Proxy

The Swell Tsunami offers yet another form of redirection that is often appropriate in smaller networks where routing infrastructure may be minimal, or the control over DHCP service is limited. It allows a Tsunami to be inserted into nearly any network, where it will transparently bridge all non-web traffic and perform interception proxying of web traffic.

Enabling bridging with redirection is rather complex, as it requires customization of the kernel to enable bridging firewall functionality, setting up of the bridge across two or more interfaces, and then redirection of port 80 traffic on the bridge device. Tsunami systems automate this through the use of another system service called `bridge`. When combined with the `redir` service, very little configuration is required beyond setting an IP for the bridge `brg0` device.