"Mosaic Will Kill My Network!"
Studying Network Traffic Patterns of Mosaic Use

Jeff Sedayao, Software Systems Engineer
Intel Corporation

Abstract

Mosaic frightens many network administrators. With a mere click of a mouse button, a user can cause megabytes of multimedia objects to enter a network. This is of great concern in many environments where networks are heavily utilized. But what are the typical network effects of a Mosaic users? How can we minimize any negative affects and still give users good performance? Through examination of packet traces, analysis of log files, and simulation of an HTTP proxy server. we discover the effects of running Mosaic. First, we attmept to construct a network profile of an "average" Mosaic user at Intel. We take a look at the characteristics of common HTTP operations and the sizes of URLs that users retrieve. With this information in hand, we recommend some ways to improve performance and minimize the utilization of network resources. Finally, we discuss areas for future investigation.

Introduction

Mosaic frightens many people. Network administrators fear that their networks will drown in an every increasing tide of multimedia objects. Managers worry that employees will spend all of their time browsing Web sites around the planet. While to some these views may seem alarmist, they are valid concerns in many environments. In any case, the following questions must be asked: what are the network effects of Mosaic use? How can Mosaic performance be improved? How can any negative effects of running Mosaic be minimized? This paper attempts to answer these questions.

We take the following approach. First, we try to construct a profile of an average Mosaic user. We then look at the characteristics of the URLs people access. Next, we examine what a typical Mosaic transaction demands of a network. With this data, we discuss ways performance can be improved and network impact minimized.

To answer our questions, we have available two resources. First, we have the log files of a CERN proxy server at Intel. This proxy server is used by about 800 people at Intel. The logs contain information on the size and origin of all the URL's accessed for the past four months. Very few if any users of the proxy also use the no caching pragma, so our record of the what people requested was very complete. Second, we could utilize a Network General Sniffer (tm) to get packet traces for analysis. This tool allows us to see exactly what happens on the network during Mosaic sessions.

Given these tools, we set out to look at what Mosaic use does to our network. In the first part of this paper is an attempt to profile the "average mosaic" user. The next section characterizes typical Mosaic transactions. With the data from the first two sections, we suggest ways that network managers, Web publishers, and Web browser implementors can improve performance and be less demanding on the networks and the Internet. The final section describes areas for future research.

Profiles of Mosaic use

Our first instinct was to go through the log files and determine the average number of requests and the average number bytes that the user put on the network. Since the vast majority of Mosaic users at Intel, are using PC's or Unix workstations, one user per client is a reasonable assumption. With this information, an analysis was run. The average number of HTTP requests made was up being 3.5 requests per user per day!

This number doesn't seem particular useful. Indeed, the whole notion of an "average Mosaic user" intuitively does not seem useful or correct. It is hard to imagine that anyone would make 3.5 requests per day (the half request would be particularly difficult). It seems more likely that people would use Mosaic a lot on a few days, not at all on some days, and a little on others. Simply starting up Mosaic with a home page containing in-lined images can result in three or four HTTP requests. The same analysis was then applied on a weekly basis. This new analysis would determine the number of accesses made per week by users using Mosaic that week. The results seemed odd. Although the number of total HTTP requests rises (figure 1), then number of requests per user per week declines (figure 2).

After some thought, the result does not seem odd. We installed Mosaic to be a tool to help people do their jobs. After exploring the Web and learning how to use Mosaic, Intel workers settle down to look at the URLs that are needed for their job or are of pressing personal interest. They are, after all, paid and rewarded for doing their jobs, not browsing around with Mosaic. This result should be reassuring to managers who fear that their employees will idle away all of their time with Mosaic. Accesses may increase in the future, however, as more and more internal information becomes available on Intel's internal Web, making Mosaic an integral part of the workplace.

We wanted to get an estimate of the local network traffic load that Mosaic users put on the network. The proxy server logs contain information on the size of URLs brought in. Unfortunately, the logs do not distinguish between a server refusing to provide information and a proxy server cache hit. Also, the logs do not record the amount of data brought in on gopher requests. These problems were dealt with as follows. Gopher was such a small part of the URLs accessed (less than 5%) that it could safely be ignored. To solve the other problem with the logs, we constructed a crude simulation of a proxy server. The simulation read in the log files, kept a cache of URLs and URL sizes, and then recorded the amount of data pulled in off the Internet and then resent on our Local Area Network (LAN) to Mosaic clients. The results were similar to those for the number of requests. Total traffic had a rising trend while the amount of bytes moved per user had a falling trend (figure 3).

The proxy server logs provide data to characterize URLs brought into the server's cache. This data gives insight about Mosaic network traffic. Table 1 shows a list of the top 10 domains that Intel users accessed. The "local" domain is for traffic going to internal Web servers.

Table 1: Top 10 domains accessed

Domain	Occurences	% Total
edu	121126	30.68
com	120338	30.48
local	45145	11.48
nl	17283	4.38
net	14589	3.7
gov	10148	2.57
fr	10139	2.57
ch	8297	2.10
uk	5488	1.39
ca	5204	1.32

Despite the existence of several Web servers internal to Intel containing such things as on-line job listing and chip simulator interfaces and despite the fact that most people point their home page inside of Intel, queries to outside of the company forms the bulk of requests. Four of the top 10 domains are countries outside of North America. Table 1 shows that there is no locality in Mosaic accesses. Indeed, Mosaic and the Web are abstractions that hide network concerns from the user. This is important to know. Network designers often rely on locality of access to segment machines together. Mosaic is totally without locality. Caching proxy servers can help, but the servers will need to perform wide ranging accesses at some point.

The size and frequency of objects accessed is important. Table 2 shows the ten file types most frequently pulled from the Internet into the proxy server. Total traffic is the product of the frequency and the average size.

Table 2: Top 10 File Extensions by Average Size

Type	Frequency	Average Size	Standard Deviation	Total Traffic
gif	33378	19413.508	59989.442	647984086
html	18926	4302.440	15926.050	102246583
jpg	1951	75315.573	102576.154	146940682
xbm	1619	3326.491	13396.082	5385589
htm	805	6172.908	44303.098	4969191
mpg	659	622351.082	685805.078	410129363
txt	337	23220.125	58770.052	7825182
au	233	610052.112	1849714.549	142142142
ps	141	210336.475	286788.542	29657443
jpeg	57	63566.474	87362.260	3623289

GIF files are the most frequently accessed type of file, followed by html files. While the average size of a gif file is 19413.508 bytes, there is tremendous variation. Table 2 shows that multimedia nature of Mosaic traffic. Images (GIF, XBM, and JPEG), video (MPEG), and sound (AU) all are being moved through the network because of Mosaic. Note that the htm extension is also an HTML file. GIF files and XBM files are common because they are native in-line images formats in the Web.

Table 3 shows the 10 ten extensions of files sorted by total traffic.

Table 3: Top 10 File Extensions Sorted by Total Traffic

Extension	Frequency	Average	Standard Deviation	Total Traffic
gif	33378	19413.508	59989.442	647984086
mpg	659	622351.082	685805.078	410129363
jpg	1951	75315.573	102576.154	146940682
au	233	610052.112	1849714.549	42142142
html	18926	5402.440	15926.050	102246583
ps	141	210336.475	286788.542	29657443

Table 3: Top 10 File Extensions Sorted by Total Traffic

Extension	Frequency	Average	Standard Deviation	Total Traffic
mov	12	2425062.667	1429724.695	29100752
mpeg	15	1598018.933	3571310.232	23970284
zip	34	390159.647	717634.145	13265428
z	41	245654.293	565730.474	10071826

GIF files generated the most traffic. This is not surprising, considering that they are the most frequently accessed type of file. MPEG video clips produced the next most traffic. The average MPEG file is very large - 622351.082 bytes. Although there were only 659 distinct mpg files, they generated a tremendous amount of traffic. Note that the MOV file extension are video clips in the QuickTime (movie format). ZIP and Z extensions are for compressed files, which are often archives.

Another source of traffic comes from the Common Gateway Interface (CGI) scripts. These scripts are used to get forms input and to execute programs on a Web server. Analysis of our logs found 9392 unique scripts that were executed. These scripts generated 170 Megabytes of traffic, which would place it third in the list of URL types in total traffic, behind GIF files and MPEG clips. Scripts are notable in that their results cannot be cached. The top 3 most frequently accessed URLs going through the proxy server are scripts. They account for 11% of all requests made through our proxy server. This implies that the 11% of the requests cannot be cached and must go out to the Internet.

Characterizing Mosaic Transactions

What happens to the network when a Mosaic user clicks on some highlighted item? We took a packet trace to see how Mosaic clients, proxy servers, and Web servers behave when requests are issued. Since all our Mosaic clients go through a proxy server, we also record the proxy server's behavior. The following is the sequence of events for a successful HTTP GET request:

1. Client sets up a TCP connection to a proxy server.
2. The proxy server sends an acknowledgement.
3. The client sends a GET command.
4. The proxy server does a DNS lookup of the name of the machine associated with the IP address of the client.
5. The proxy server does a DNS lookup of the IP address of the name it just received. This is done to verify the information received in the previous step.
6. If the URL is not in the cache, then
    a. Proxy server does a DNS Look up of the server in the URL.
    b. Proxy server sets up connection to server.
    c. Server sends an acknowledgement.
    d. Proxy server sets up connection to server in the URL.
    e. Server begins sending requested data.
7. Data is sent from the proxy server to the client.
8. If the URL was not cached, the proxy server shuts down the connection to the server.
9. The client shuts down the connection to the proxy server.

There are a number of items of interest here. First, there are many DNS lookups associated with a proxy server. The proxy server looks up the client system's name from the IP address, and then looks up the IP address based on the answer it receives in order to verify the answer. It performs at least two DNS lookups for each URL, and three if the URL is not in the cache. Second, there are at least one (maybe two) TCP connection setups for each URL. Third, the whole process must take place for for EACH of the URLs that are used to display a page. Each in-lined GIF file in a page must go through this process. Apparently in-lined images were not known when HTTP was created. It would be more efficient and faster for Mosaic users if all of the items in a page were sent over the same TCP connection. It does, however, require more state to be maintained in a server. I am told that this issue is well known and may be addressed in the next version of http.

Mosaic waits for each GET request to complete before executing the next one. Any delay in the sequence of events will hurt the performance of a Mosaic user. We had an incident where our nameserver was returning answers to DNS queries extremely slowly (on a scale of seconds). As a result, the traversing links was extremely slow.

Implications of Mosaic Traffic Patterns

So far, we have profiled a large population of Mosaic users. We also described what happens at the network level what happens during a typical Mosaic transaction. The questions remain: what does Mosaic use mean for a network manager? How can system administrators, Mosaic implementors, and Web publishers maximize the performance of Mosaic while minimize the impact on network? This section tries to answers these questions.

We need to define Mosaic performance. Let us consider "good performance" to be the ability to bring up pages and images quickly. Mosaic ceases to be usable if it takes several minutes to bring up each page.

Implications for Network Managers

Mosaic intensively generates many short lived TCP connections. Each image in each page will need at least one TCP connection set up and tear down. These connections will not live long. The most common URL file type, GIF images, averages a little more than 19 Kilobytes in size. This is less than one second of transfer time on a T1 leased line, and much less on an ethernet. There is no locality to these connections. Networks will have to deal with this extremely fast connection set up and tear down to places all over the planet. Low delay on a network positively impacts Mosaic performance.

While dealing with the short-lived connections, Mosaic also brings in long-lived connections that move massive amounts of data. A multimedia object like an MPEG video clip can be several megabytes in length. High bandwidth would help Mosaic performance, although when getting data from the Internet, performance is limited by the server providing the information and the smallest pipe between the server and the client.

Mosaic demands low delay and high bandwidth, while at the same time providing no locality in access. What is a network manager to do? One of the best ideas would be to use a caching proxy server. This allows a network manager to provide some locality in access on a high bandwidth low delay LAN. There are some tradeoffs here. The proxy server adds some delay for URLs that are not cached or that cannot be cached. Also, because proxy servers (at least the one from CERN that we use) are very demanding on name servers, the systems running proxy servers should also be caching name servers.

Implication for Web Publishers

Publishers on the Web can do much to improve performance. One of problem that has been discussed in various forums is the gratuitous use of images. At least one TCP connection is set up for each image in page. One of the most annoyingly used graphic is the "ball" image used as bullets in lists of items. We looked through our logs for "ball.gif" files. We found 536 unique balls. The overhead for each different ball is high (2 connections and 3 DNS lookups with a proxy server) considering how each ball is typically less than 1 Kbyte in size. Moreover, Web publishers all seem to use their own balls, so new balls are often brought in even if the exact same image is in the cache. While there may be reasonable uses for balls of different colors on a page, there are many cases where they are totally unnecessary.

The point of many Web pages is the graphics. Still, there are ways to minimize the impact of images. Instead of many separate images that would have to be brought in individually, images can be combined into a single montage that only needs one TCP connection to view. Web publishers need to trade off presentation ideas with their impact on performance. When a user traverses a link, the user cannot know with certainty how many GET requests will results or how long it will take to transfer a multimedia file. It would be helpful if publishers give their audience a rough idea of how long it may take get a URL and let the reader decide whether to traverse a link.

Another thing that publishers should think about is the use of CGI scripts. There can be no caching of the results of running a script. While there are many reasonable uses for CGI scripts, there are some uses that harm performance. One particular irksome example is the use of a CGI script to deliver MPEG video clips. This may be understandable if the video clips change on a regular basis. But using CGI scripts means that every time someone wants to see a particular MPEG file, it has to be copied over the Internet. Even if a user has just seen the MPEG, it has to be downloaded all over again. This is not particularly nice to the user, and it certainly isn't nice to the Internet.

There are of course tradeoffs when using or not using CGI scripts. CGI scripts are useful when you don't want items to be cached. It is a good idea to present rapidly changing information with a CGI script. Another good place to use scripts is if you want tightly controlled access to information on a server.

One generally useful technique that Web publishers should consider is mirroring. Having mirrors of your information in different parts of the world can provide much better performance to users. It has the added benefit of making your information more available to your target audience even if a server goes down.

In general, Web publishers should not think of networks and the Internet as a mere abstractions. Performance is an inherent part of the user's experience. It is highly unlikely that a user will return to a page that takes 20 minutes to load, even if the page is loaded with pretty pictures.

Implication for Mosaic Implementors

Mosaic and other Web browser implementors can do a number of things to minimize network impact and improve performance. First, browsers should support proxy servers. Fortunately, most do. Second, implementations should support a large image cache. Some implementations do not. Third, implementations should make better use of documents that are cached in the browser. For example, if you use "back" and "forward", you can go to pages very quickly. If you do "open" (or even "home") to one of the same pages that you can go to with "back" and "forward", you reload the page. This is slow and generates unnecessary network traffic.
<H2>Directions for Future Work</H2>

While this paper has pointed out a number of problems and ways to alleviate them, there are many areas that still must be looked at. The upcoming URN technology (RFC currently in draft) may alleviate some of the problems described. Since objects will live forever, it will be easier for people to reuse and cache them and not have to worry about constant updates. Perhaps a set of multicolored balls could be defined and used by everyone. In another area, the HTTP protocol must be followed and pushed to generate fewer connections.

We built a crude simulation of a caching proxy server, provided as input a trace of actual requests, and then measured what happened. This is similar to the trace-driven simulations used to study microprocessors and memory caches. It would be worthwhile to do more detailed simulations of caching proxy servers. The effectiveness of caching strategies like pre-fetching popular URLs could be measured.

Finally, the tradeoffs of using different presentation techniques need to be investigated. For example, will an imagemap script be less of a network burden than a page of clickable images? Which is a more effective presentation? How frequently should my data change before I switch from using files to a CGI script? This research in this area should yield very useful results.

Acknowledgements

Sniffer is a trade mark of Network General.

Special thanks to Kevin Altis and Sally Hambridge for their comments, and suggestion. Thanks also to W. Wiley of cisco systems for bringing up important points.

References

Altis, Kevin; Ati Luotonen. ”World Wide Web Proxies.” **First International Conference on the World-wide Web**. Geneva, Switzerland. May 1994.

Altis, Kevin. Personal Correspondence.

Masinter, L; Sollins, K. “Functional Requirements Uniform Resource Names”. Draft Request for Comments, September 1994.

Author Biography

Jeff Sedayao received a B.S.E. in Computer Science from Princeton Unversity in 1986 and a M.S. in Computer Science form the University of California at Berkeley in 1989. He has worked at Intel Corporation since 1986, spending most of his time running Intel’s Internet gateways. Reach him at Intel Corp; SC9-37; 2250 Mission College Boulevard; Santa Clara, CA 95052-8119. Reach him electronically at sedayao@argus.intel.com.

Author E-mail Address

sedayao@argus.intel.com

Figure 1: Total Requests per Week

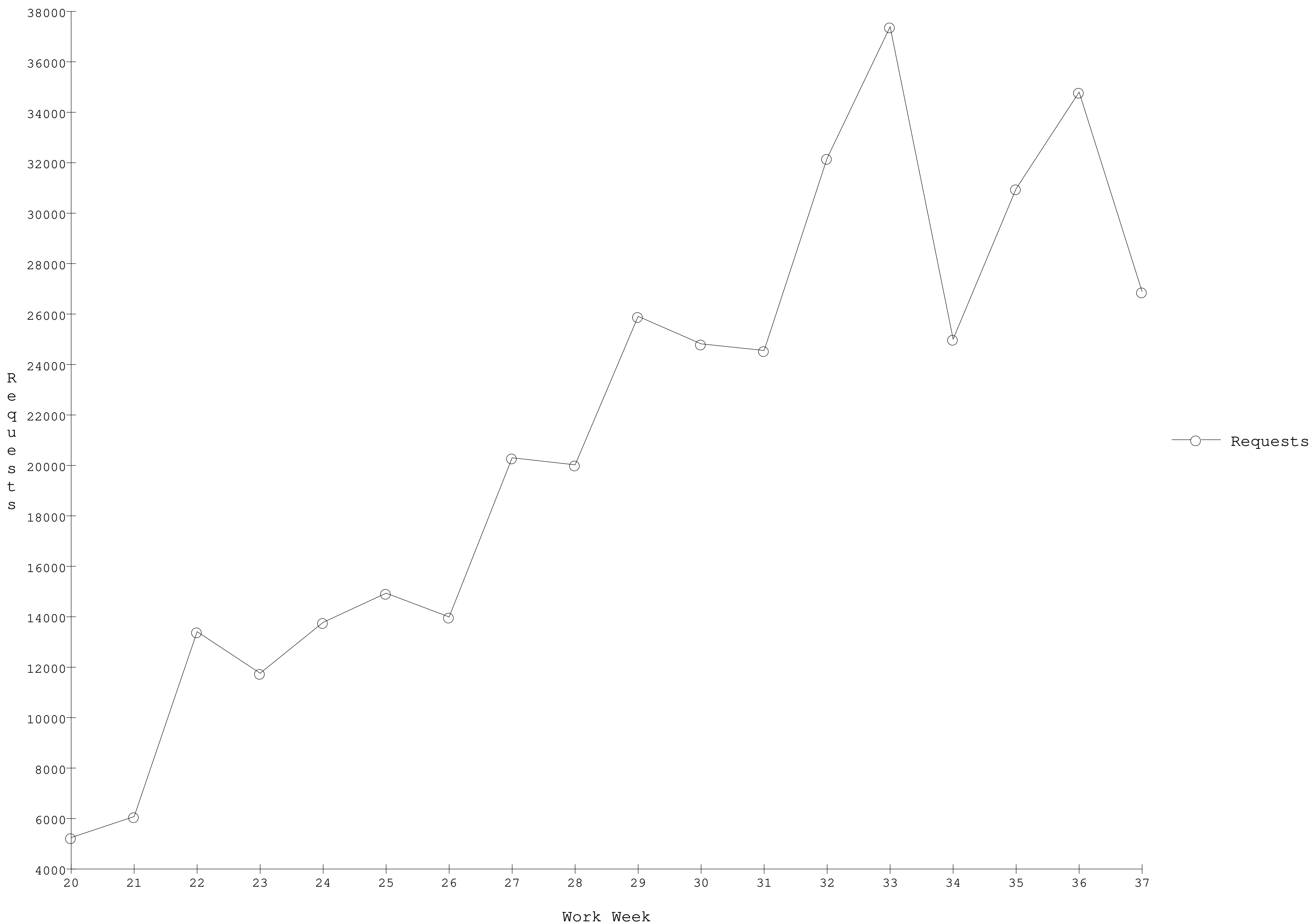

Figure 2: Requests Per User Per Week

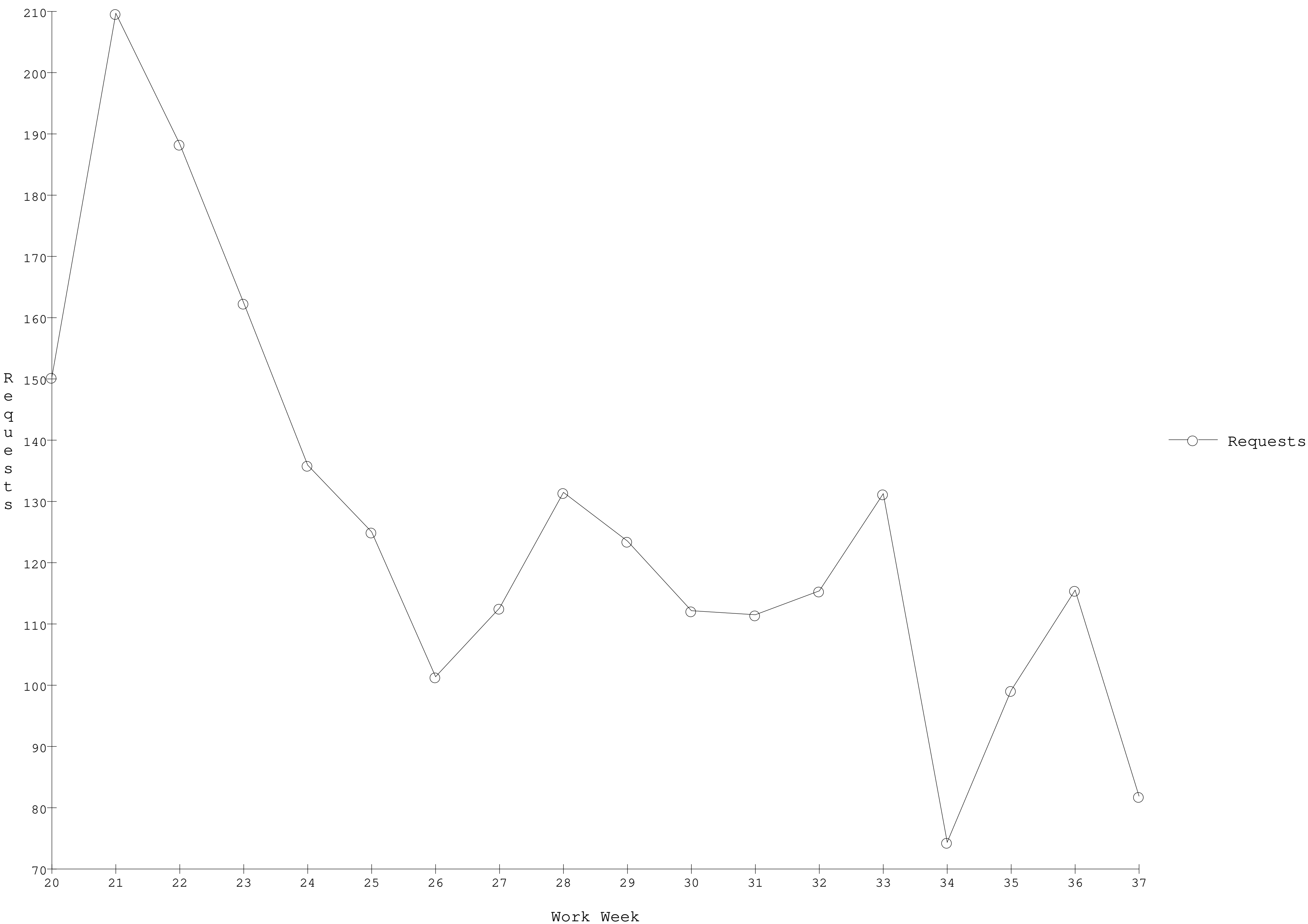

Figure 3: Average LAN Traffic per User per Week