

# Introduction to web server traffic modeling and control research

Andersson, Mikael

2005

# Introduction to Web Server Modeling and Control Research

TECHNICAL REPORT

Mikael Andersson
Department of Communication Systems
Lund Institute of Technology
Email: mike@telecom.lth.se

Publication code:
CODEN: LUTEDX(TETS-7211)/1-27/(2005)&local 26

October 28, 2005

## Abstract

A significant amount of papers has been published on web server server modeling and control. This survey presents the current results in these two fields. Background information is given that explains and examplifies the problems found in respective fields. A list of references for further reading is included.

# Contents

# 1 Introduction

During the last years, the use of Internet has increased tremendously. More and more users connect to the Internet. In Sweden, more than 70 percent of the population used the Internet last year (according to Statistics Sweden, [1]). Not only the number of users has increased, the number of services offered on the Internet has exploded the last few years. Companies take their business onto the Internet to a greater extent. 75 percent of the companies in Sweden use Internet to market themselves. The companies are e-commerce ventures that sell records, books, clothes and services, companies that want to present themselves on the Internet, banks, gambling sites, web hotels and so on. The growth in Internet popularity has lead to increasing demands in bandwidth and performance over the Internet and both bandwidth and computer speed *have* increased steadily. However, this is not always enough. Many people still experience the WWW as the ”World Wide Wait”. Instead of being fast and useful, the Internet is at many occasions time-consuming.

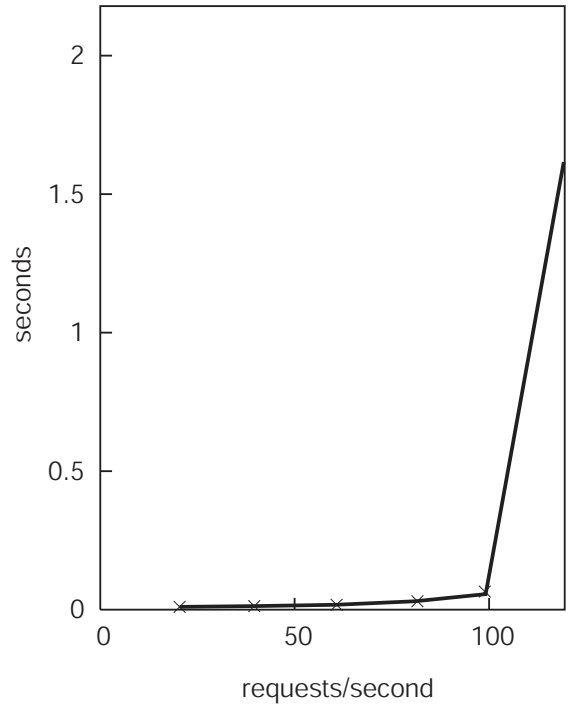


Figure 1: The response time goes to infinity when the server load increases.

The long response times do not necessarily have to depend on too little bandwidth or too slow clients. Instead, the bottleneck is often the server systems. Numerous examples can be found when web servers have become too overloaded, leaving all visitors ignored. Situations when this occur is for example when a news site reports events like sport tournaments, crises or political elections. Web shops can be hit with many visitors during sale events on the web, bank sites during pay days, regular companies when they release new products etc.

When a web server gets overloaded, the response time for a web page becomes long which affects the company, as shown in Figure 1. If visitors experience long response times, they tend to choose other alternatives on the web, they turn to another web shop or go to another news site.

Web servers also plays a central role in the event of a crisis. During a crisis, information has to be published by authorities, companies and governments that are involved in the crisis. A crisis put hard requirements on a server due to increasing traffic. Several occasions have shown that critical web sites were severely overloaded. The result is that no information is available when it is most needed.

The rest of this survey is organized as follows: Section 2 gives a background of the research fields, Section 3 covers the current results in web server modeling. Section 4 covers research on overload control on a web server. Finally, a short description of suggested research problems is given in Section 5.

# 2 Background

This section gives an overview of the areas of research covered in this survey. Web servers play a central part, so an explanation of web servers is given, together with a description of the architecture of Apache [2], which is the server used in many papers. Performance modeling of web servers is discussed and the general structure for an admission control mechanism[1] in a web server is given.

## 2.1 Web servers

The web server software offers access to documents stored on the server. Clients can browse the documents in a web browser. The documents can be for example static Hypertext Markup Language (HTML) files, image files or various script files, such as Common Gateway Interface (CGI), Javascript or Perl files. The communication between clients and server is based on the Hypertext Transfer Protocol (HTTP) [3].

When a document on the web server is requested, an HTTP request is constructed and sent to the server. The HTTP request is encapsulated in a TCP segment. The TCP contains the port address (the destination application on the server, i.e. the web server software) and is in its turn encapsulated into IP packets containing the IP address of the server. On local area networks, the IP packets are put into Medium Access Control (MAC) frames that addresses the next computer on the route towards the server. Figure 2 shows the decomposition of an HTTP request into TCP segments, IP packets and MAC frames.

An HTTP transaction consists of three steps: TCP connection setup, HTTP layer processing and network processing. The TCP connection setup is performed as a so called three-way handshake, where the client and the server exchange TCP SYN, TCP SYN/ACK and TCP ACK messages. Once the connection has been established, a document request can be issued with an HTTP GET message to the server. The server then replies with a HTTP GET REPLY message. Finally, the TCP connection is closed by sending TCP FIN and TCP ACK messages in both directions. Figure 3 illustrates the TCP communication.

There are many web servers on the market today. Four main types can be identified; process-driven, threaded, event-driven and in-kernel web servers. Threaded and process-driven web servers are the most

[1] An admission control mechanism is a mechanism used for preventing web servers from overload.

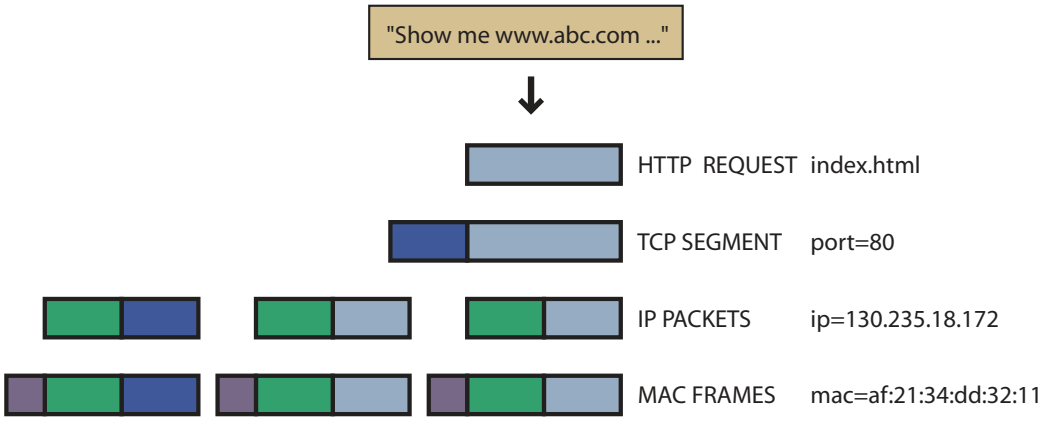


Figure 2: The structure of an HTTP request.

common, with Apache being the most popular currently. Another popular process-driven web server is Microsoft's IIS [4], covering about 21 percent of the market (Netcraft [5]). Examples of event-driven web servers are Zeus [6] and Flash [7]. A description of event-driven web servers and overload control strategies for such servers is found in [8]. In-kernel web servers are servers that are executed in the operating system kernel, for example Tux [9] and khttpd [10].

### 2.1.1 Apache

Introduced in 1995 and based on the popular NCSA httpd 1.3, Apache is now the most used web server in the world (Netcraft [5]). It is used in more than 67 percent of all web server systems (more than 52 millions in total, July 2004). One of the reasons to its popularity is that it is free to use. Also, since the source code is free, it is possible to modify the web server. Being threaded (threaded or process-driven depending on the operating system, on Unix, Apache uses processes, while threads are used in Win32 environments) means that Apache maintains a pool of software threads ready to serve incoming requests. Should the number of active threads run out, more can be created. When a request enters the web server, it is assigned one of the free threads, that serves it throughout the requests' lifetime. Apache puts a limit on the number of threads that are allowed to run simultaneously. If that number has been reached, requests are rejected.

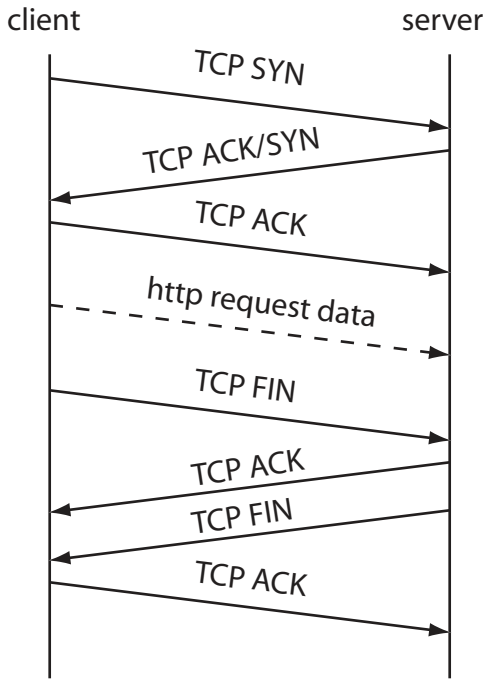


Figure 3: The TCP session.

### 2.1.2 Modules in Apache

What makes Apache so attractive is also its architecture. The software is arranged in a kernel part and additional packages called modules. The kernel is responsible for opening up sockets for incoming TCP connections, handling static files and sending back the result. Whenever something else than a static file is to be handled, one of the designated modules takes over. For example, if a page with a CGI script is requested, the mod_cgi module launches the CGI engine, executes the script and then returns the finished page to the kernel.

Modules are convenient when new functionality should be added to a web server, because nothing has to be changed in the kernel. A new module can be programmed to respond to a certain type of request. Modules communicate with the kernel with hooks, that are well-defined points in the execution of a request. In Apache, every request goes through a life-cycle, that consists of a number of phases, as shown in Figure 4.

The phases are for example **Child Initialization**, **Post Read Request**, **Handlers**, and **Logger**. When a module wishes to receive a request it has to register a hook in the kernel, that is valid for one or

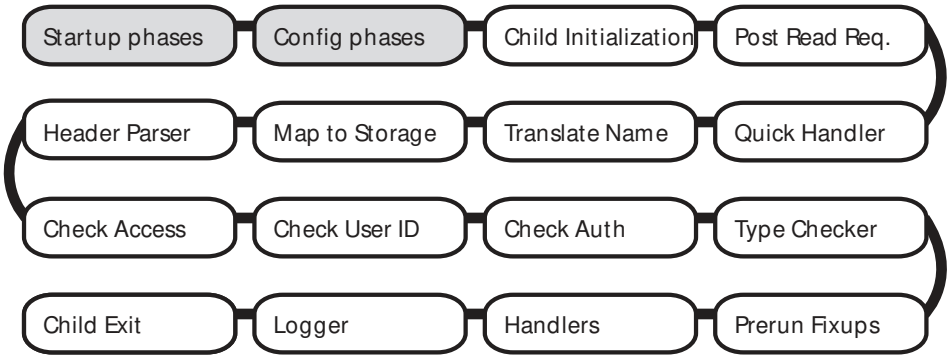


Figure 4: The phases in the request life-cycle in Apache as of version 2.0. The shaded phases represent the phases that occur during the startup of the web server. Child initialization and exit phases are only called once in Win32 environments. In a process-driven environment (Unix), they are called for all requests.

more of the phases. For example, mod_cgi is registered to be notified in the Handlers phase, which means that once the request has reached so far in the kernel, it is delivered to the mod_cgi module. mod_cgi then performs its duties and returns the request back to the kernel. The kernel checks whether other modules want to get a hold of the request in the Handlers phase before continuing to the next phase (Logger). A more detailed description of the Apache architecture is found in [11] and [12].

## 2.2 Performance modeling

To be able to design an efficient overload control it is important to have a good and reasonable performance model of the web server. It also has to be simple enough to be able to use in practise. Traditional modeling of telecommunication systems means modeling the systems as queuing systems from classical queuing theory. A performance model is meant to answer questions like *"What is the average response time at this request rate?"*, *"What is the throughput?"* and *"What is the rejection probability?"*. The M/G/1/K processor sharing model by Cao et al. in [13] (shown in Figure 5) works good for these questions.

Using the processor sharing queuing discipline models the concept of using simultaneously executed threads or processes served in the web server well. There are many other models that quite well captures

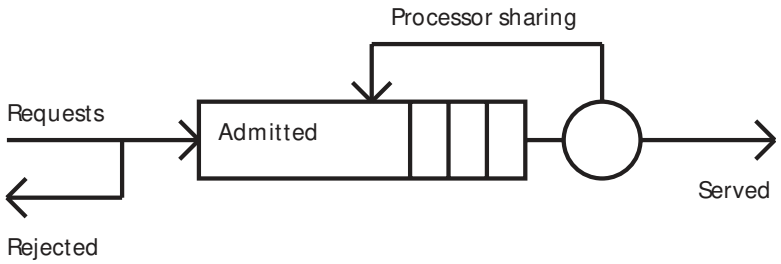


Figure 5: The web server model.

the inner-most details in web servers, described in Section 3.

An issue in performance modeling when it comes to overload control is that the model must capture the performance metrics well in the overload region of the web server.

## 2.3 Overload control

An overload control mechanism is designed for a certain system that is sensible to overload from being overloaded. The term overload is not clearly defined in the literature. Sometimes it refers to when response times from a system are too long or the system has crashed, but it can also mean that the CPU usage is close to 100 per cent or another part of the system is overloaded e. g. the hard disk or the Ethernet card. However, a general definition could be

> A load, placed on a device or facility, that is greater than the device or facility is capable of handling, i.e., capable of performing the functions for which it was designed. [14]

Four main types of overload control can be identifyed: **Admission control** (sometimes referred to as Access control), **Content adaptation**, **Load balancing** and **Scheduling**. It is not necessary to use only one of the four types in an overload control mechanism. Often two or more are combined in a more robust solution. A brief introduction to the four schemes follows below.

## 2.4 Admission control

Admission control mechanisms have since long been designed for telecommunication systems. The admission control mechanism is intended to

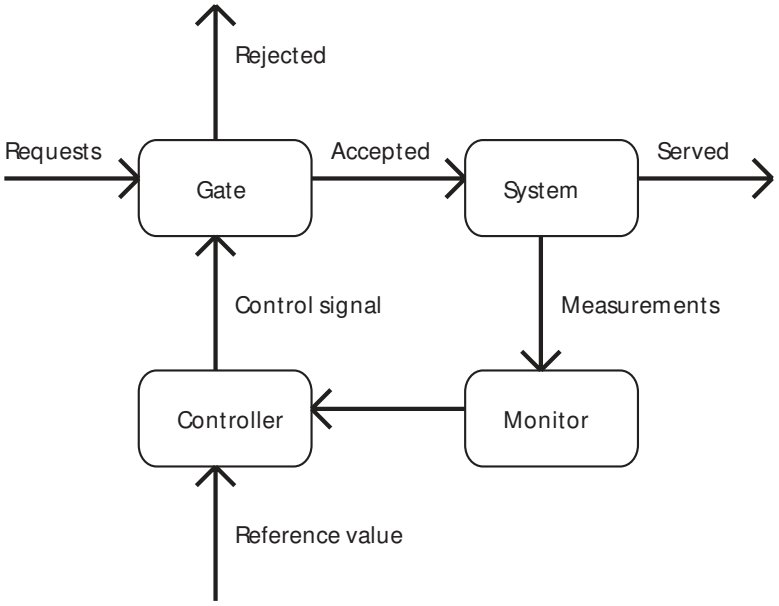


Figure 6: An admission control mechanism.

prevent the system from becoming overloaded by rejecting visitors. Figure 6 shows a general structure for admission control. The structure contains three modules; the **Gate**, the **Controller** and the **Monitor**. Since continuous control is not possible in computer systems, time is divided into control intervals. In the beginning of each control interval, the Controller calculates the desired admittance rate for the next interval based on the measurements from the Monitor. The Gate then either admits or rejects visitors depending on the control signal.

### 2.4.1 The Monitor

The Monitor monitors the system through measurements on specific performance metrics. Measurements are taken each control interval. The performance metrics that are monitored differ from system to system:

**CPU utilization**
The Monitor measures the utilization in the server each control interval. The goal is to keep the utilization to be lower than a threshold value.

**Queue lengths**
Queue lengths can be measured, for example TCP buffers, HTTP server queues, network card buffers etc. Filled buffers indicate a high load on the server.

**Response times**
The average response time is also an important metric in overload control. If the response time is too high, the server is considered overloaded.

**Call count control**
Here, the arrivals are counted. Only a certain amount of visitors are allowed at one time. This is the case in the original Apache overload control, where a maximum number of threads are set.

### 2.4.2 The Controller

The Controller's task is to decide how many visitors can be admitted into the system, by trying to keep a certain reference value for the desired performance metrics. It compares the actual measurements to the reference value and then reacts according to the deviation. The Controller can be designed in a variety of ways, see Åström [15] for more details:

**Static controller**
The most simple is when the Controller has a static value that never changes, for example, "Admit 25 visitors every second."

**Step controller**
The step controller has a lower and an upper bound that it allows in the measurements. Whenever the monitored data goes above or beneath these bounds, the output signal to the Gate is increased or decreased with a fixed value per control interval.

**On-off controller**
The on-off controller works in a similar way to the step controller, but instead of increasing/decreasing the admission rate, it admits all or none of the requests in a control interval.

**PI controller**
There are several controllers that can be picked from control theory. A

classical example is the PI-controller that has two parts, one proportional to the error and one that is the integral of the error.

### 2.4.3 The Gate

The Gate's task is to admit or reject visitors based on the Controller output. Many different gates can be found in the literature, where the most common ones are:

**Token bucket**
The token bucket algorithm generates tokens (”admission tickets”) at a rate set by the Controller. If there are available tokens upon the arrival of a request, it is admitted. Each admitted request consumes one token. Should there be no tokens, the incoming requests are buffered in a queue.

**Leaky bucket**
The leaky bucket is similar to the token bucket. Both are designed to smoothen out bursty arrival traffic. Arriving requests enter a queue, that has a limited size. If the buffer is full, the requests are rejected. Admitted requests are allowed to leave the queue at a rate set by the Controller.

**Dynamic window**
The dynamic window version works like in TCP, where only a fixed amount of unacknowledged TCP are allowed to be sent. To be able to send more TCP segments, the user has to wait for the previously sent segment to be acknowledged. The basic admission control offered in an unmodified Apache works like this, a fixed number of threads or requests is set as an upper bound. In the Apache case, the Controller part can be seen as an on-off controller that reports to the Gate whenever the bound has been reached.

**Call gapping**
A call gapping gate admits a number of requests in the beginning of each control interval. Additional requests are rejected.

**Percent blocking**
When percent blocking is used, a percentage of the requests are admitted each control interval.

## 2.5 Content adaptation

Admission control is one way of preventing a web server from being overloaded. Another technique is content adaption. Content adaption means that content-heavy pages are reduced during heavy load. For example, CGI scripts are very time-consuming for a processor, but nevertheless, modern web sites are often written entirely in some script language. During overload, scripted pages can be dynamically changed to static versions instead. This lowers the load on the server at the cost of lower functionality. Content adaption is not covered in this survey. More can be read about it in the works of Abdelzaher et al. [16], Mohan et al. [17] and Lee et al [46].

## 2.6 Load balancing

Load balancing is closely related to overload control. It is used in server clusters where more than one server is used. A load balancer's task is to distribute incoming requests among the clustered servers so that the performance is optimized. The difference between load balancing and admission control is that admission control is used for only one server or a system seen as a unit compared to the load balancer that operates on several servers and also the fact that a load balancer does not reject any customers.

## 2.7 Scheduling

Another concept used in overload control is scheduling. The definition according to [18] is

> Scheduling (e.g., jobs). A schedule for a sequence of jobs, say $j_1$,..., $j_n$, is a specification of start times, say $t_1$,...,$t_n$, such that certain constraints are met. A schedule is sought that minimizes cost and/or some measure of time, like the overall project completion time (when the last job is finished) or the tardy time (amount by which the completion time exceeds a given deadline).

Scheduling is not surveyed in this work.

# 3 Web Server Modeling

The survey of web server modeling research is divided into **Traffic models**, **Queueing models** and **Other approaches**.

## 3.1 Traffic models

First some definitions that are used in this section. A **file request** in this context is an HTTP request for a single file on the web server. Normally, a request to an HTML file generates subsequent requests because a normal web page contains images or style sheet files that have to be requested as separate files. The way that subsequent requests are retreived from the server depends on web server settings (whether persistent connections are allowed or not). Requests to HTML files are called **page requests**.

A **session** can be defined as

> The sequence of file requests from a certain user from his first request until the last request before leaving the site.

Therefore, a session is made up of several file requests and also several page requests and the arrival of a session is at the same time the arrival of a file request and a page request. Figure 7 illustrates the session concept.

### 3.1.1 Arrivals of requests and sessions

**File request arrivals**
One of the standard methods to model file request arrivals is the Poisson model, because of its simplicity. In telecom networks, the Poisson process has since long been used for telephone call arrivals, because calls are generated independently from each other. However, web traffic does not fit into this description. To be able to model the burstiness and self-similarity [2] of file request arrivals a number of self-similar processes have been proposed. (An extensive bibliographical overview of self-similarity is given in [19] by Willinger et al.) In [20], Crovella and Bestavros show that file request arrivals are strongly self-similar due to document size distributions, caching and user preferences and that

[2] Self-similarity means that the traffic model exhibits fractal properties, that is, the process looks the same at different time scales.

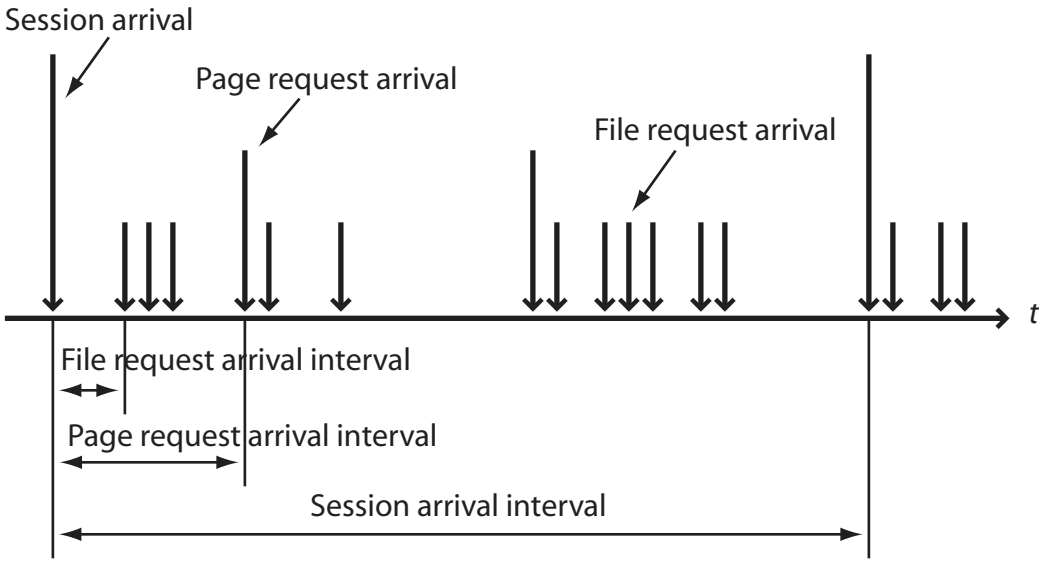


Figure 7: Arrivals of sessions, page requests and file requests.

the process can be modeled by an aggregate of multiple heavy-tailed ON/OFF sources.

**Sessions**
Since the reasons to why file request arrivals cannot be modelled as Poissonian processes do not apply to session arrivals it is not sure that sessions are self-similar. In the work of Yeung and Szeto [21], web server logs are analyzed. The interarrival time of sessions are investigated and the conclusion is that they can be modelled as beeing exponentially distributed (i. e. the arrivals follow a Poissonian process). This has also been confirmed by experiments in our research group.

**Site-dependent sessions**
Commercial web sites, e. g. web-shops, often define their own abstract sessions. A user who visits a web-shop has to log in, browse, pay and log out from the site. Session data is often stored as cookies, small text files that are stored in the customer's computer. These types of sessions are dependent of the specific site architecture and often use cookies to store session-dependent data, which can make it difficult to generalize them, Menasce et al gave a suggestion in [22].

### 3.1.2 Document distributions

Not only inter-arrival times can be studied in web server traffic. File popularity and file sizes are two other examples that can be seen in the literature.

#### File popularity

The popularity of a file (seen as the number of requests for the file compared to requests to other files at the same site) is often said to follow Zipf distributions [20, 23]. However, this is debated in [24] where a power law distribution is claimed to be the better choice.

#### File sizes

[23] uses a log-normal body for file size distribution that includes a Pareto heavy tail. Crovella and Bestavros also used Pareto modeling in [20]. In [25], Wells et al used a Weibull distributon after having compared a web server log to theoretical distributions.

## 3.2 Queuing models

Several attempts have been made to create performance models for web servers. Van der Mei et al. [26] modeled the web server as a tandem queuing network. The model was used to predict web server performance metrics.

Dilley et al. [27] use layered queuing models in their performance studies.

Kant and Sundaram give a very detailed model of a web server and its low-level components such as CPUs, caches and buses in [28] where queues or delays are used. Only static workloads are considered.

Cherkasova and Phaal [29] use a single server queueing model with deterministic service times. In their work they use a session-based workload with different classes of work.

Beckers et al. [30] proposed a generalized processor sharing performance model for Internet access lines. They established simple relations between access line capacity and the utilization of the access line and download times of Internet objects. Andersson et al also demonstrated a processor sharing queueing model in [13, 31, 32] as did Fujita et al in [33]. The processor sharing models were all based on requests, not sessions and had maximum number of requests allowed in the server at the same time.

Zhu and Lü showed a performance model of a database-backed web site in [34] where they stress the importance of including databases into performance models . The model was an open multiple-class queueing network with separate queues for web server, database, disk and network.

## 3.3 Other approaches

Wells et al. have made a performance analysis of web servers using colored Petri nets [25]. Their model is divided into three layers, where each layer models a certain aspect of the system.

# 4 Overload control

The survey of web server overload control research is divided into **Control-theoretic approaches**, **Optimization approaches** and **Other approaches**.

## 4.1 Control-theoretic approaches

Some papers have investigated admission control mechanisms for server systems with control theoretic models. Abdelzaher [35,36] modeled the web server as a static gain to find optimal controller parameters for a PI-controller. A scheduling algorithm for an Apache web server was designed using system identification methods and linear control theory by Lu et al. [37]. Bhatti [38] developed a queue length control with priorities. By optimizing a reward function, a static control was found by Carlström [39]. An on-off load control mechanism regulating the admittance of client sessions was developed by Cherkasova and Phaal [29]. Voigt [40] proposed a control mechanism that combines load control for the CPU with a queue length control for the network interface. Bhoj [41] used a PI-controller in an admission control mechanism for a web server. However, no analysis is presented on how to design the controller parameters. Gandhi et al. use MIMO control theory in [42]. Huang and Cheng also use control theory in [43] where they approach the problem by using fuzzy control to predict the self-similar traffic arrival processes to the web server. Papers analyzing queueing systems with control theoretic methods usually describe the system with linear deterministic models. Stidham Jr [44] argues that deterministic models cannot be used when analyzing queueing systems.

## 4.2 Optimization approaches

Andersson shows in [31] how optimization can be used for achieving optimal profit during overload by categorizing requests into different classes. Each class has different properties such as guaranteed throughput, revenue and maximum response time. Bose and Altinkemer show in [45] a similar approach by providing an optimization-based formulation of the design of minimizing the cost of web site design. Lee et al. demonstrate in [46] how two different goals can be achieved; maximum admission and maximum profit, by using optimization. [47,48] also use optimization approaches to handle web server overload.

## 4.3 Other approaches

More on different types of overload control strategies for distributed communication networks can be found in the survey of Kihl, [49]. Chen et al. have investigated an admission control system in [50] where they have double internal queues in the web server where one of the queues stores requests that cannot be served immediately. If by any chance there are spare capacity, requests from this queue are served, which results in a smoother performance.

# 5 Open problems

## 5.1 Distributed systems

Not many papers deal with the notion of a distributed web site. Modern web sites often consist of more than one server, they have a gateway (firewall), a authentication server, a database server and sometimes multiple copies of the static files' server (Figure 8 shows a typical web site architecture). More research is needed on how to model distributed web sites and especially how overload can be handled in these cases.

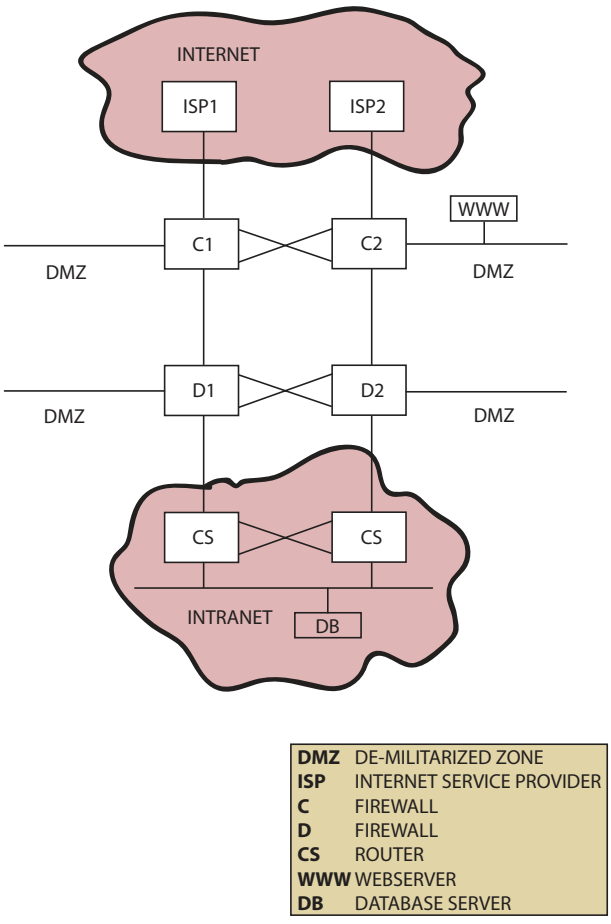


Figure 8: A typical web site architecture.

## 5.2 Crisis-related issues

So far I have found no papers that look at modeling and overload control from a crisis point of view. Modeling and control papers usually deal with normal or moderately overloaded traffic. When a crisis occur, my belief is that traffic patterns do not necessarily have to look the same and that the agreed upon distributions no longer have to be valid. Research needs to be done so that the nature of crisis-related web server traffic can be determined. Also, overload control has to be used when operating in times of crisis. During a crisis, normal assumptions of overload control goals might not valid. Web sites that normally try to optimize for profit during overload should instead return as much information as possible to as many visitors as possible. Content adaption is well suited for this goal.

The Swedish National Board of Psychological Defence (SPF) has described how cities and citizens can use Internet in a crisis situation [51] and recommend cities to have a prepared crisis page on their web sites.

In a publication ( [52]) by the Swedish Emergency Management Agency (SEMA) the murder of Anna Lindh, swedish foreign minister, in September 2003 is investigated from a crisis information point of view. One of the results from the study regards the role of the new communication technology;

> But despite the fact that the technological capacity to receive many visitors on the Net has increased over recent years, the loading is so heavy on the central authorities and national media that problems arise. Information Rosenbad's web site went down for a while due to the number of visits, e.g. when the web was used for press releases in other languages. The number of visitors to media websites increased just as strongly. At national level the newspapers' and TV companies' pages were overloaded, and the number of visitors to local papers' websites increased also at local level.

The Committee on the Internet Under Crisis Conditions: Learning from September 11 reports similar overload conditions from web sites during the reporting of the September 11 attack in New York [53].
In a report of a working session of the End-to-End Research Group in 2005 by Clark et al, [54], a set of research objective is presented that aims at making the future Internet better in some senses. One of the discussed research areas is the **Operation in times of crisis**. The authors describe a need for mechanisms for securing the Internet in

crisis situations with the following objective:

**”In 10 years, the network itself, and critical applications that run on it, should address the special needs that arise in times of crisis.”**

## Acknowledgements

This report builds on results from our research group, PANDA (Performance Analysis of Distributed Applications). The author would like to thank Dr. Martin Höst and Dr. Christian Nyberg for valuable comments to the manuscript.

## References

[1] "Statistics sweden," http://www.scb.se.

[2] "Apache web server," http://www.apache.org.

[3] W. Stallings, *Data & Computer Communications*. Prentice Hall, 2000, sixth Edition.

[4] "Microsoft internet information services," http://www.microsoft.com/WindowsServer2003/iis/default.mspx.

[5] "Netcraft," http://www.netcraft.com.

[6] "Zeus web server," http://www.zeus.com.

[7] "Flash web server," http://www.cs.princeton.edu/~vivek/flash/.

[8] T. Voigt, "Overload behaviour and protection of event-driven web servers," in *In proceedings of the International Workshop on Web Engineering*, May 2002, pisa, Italy.

[9] "Tux reference manual," http://www.redhat.com/docs/manuals/tux/TUX-2.1-Manual/.

[10] "khttpd web server," http://www.fenrus.demon.nl/.

[11] "Apache developer documentation," http://httpd.apache.org/docs-2.0/developer/.

[12] B. Laurie and P. Laurie, *Apache, The Definitive Guide*. O'Reilly, 2003.

[13] J. Cao, M. Andersson, C. Nyberg, and M. Kihl, "Web server performance modeling using an m/g/1/k*ps queue," in *In Proceedings of International Conference on Telecommunication (ICT)*, 2003.

[14] "Atis home page," http://www.atis.org/tg2k/_overload.html.

[15] K. Åström and B. Wittenmark, *Computer-controlled systems, theory and design*. Prentice Hall Internation Editions, 1997, third Edition.

[16] T. Abdelzaher and N. Bhatti, "Web content adaptation to improve server overload behavior," *Computer Networks*, vol. 31, no. 11-16, pp. 1563–1577, 1999.

[17] R. Mohan, J. Smith, and L. Chung-Sheng, "Adapting multimedia internet content for universal access," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 104–114, 1999.

[18] "Mathematical programming glossary," http://carbon.cudenver.edu/~hgreenbe/glossary/index.php.

[19] W. Willinger, M. Taqqu, and A. Erramilli, "A bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks," 1996. [Online]. Available: citeseer.ist.psu.edu/willinger96bibliographical.html

[20] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," in *Proceedings of SIGMETRICS'96: The ACM International Conference on Measurement and Modeling of Computer Systems.*, May 1996.

[21] K. H. Yeung and C. W. Szeto, "On the modeling of www request arrivals," in *Proceedings of 1999 International Workshops on Parallel Processing*, 1999, pp. 248–253.

[22] D. Menasce, V. Almeida, R. Fonseca, and M. Mendes, "Business-oriented resource management policies for e-commerce servers," *Performance Evaluation*, vol. 42, pp. 223–239, 2000.

[23] C. Williamson, R. Simmonds, and M. Arlitt, "A case study of web server benchmarking using parallel wan emulation," *Performance Evaluation*, vol. 111-127, no. 49, 2002.

[24] "File popularity characterisation," http://www.ee.ucl.ac.uk/~imarshal/sigreview.pdf.

[25] L. Wells, S. Christensen, L. M. Kristensen, and K. H. Mortensen, "Simulation based performance analysis of web servers," in *Proceedings of the 9th International Workshop on Petri Nets and Performance Models (PNPM 2001)*. IEEE Computer Society, 2001, pp. 59–68.

[26] R. D. V. D. Mei, R. Hariharan, and P. K. Reeser, "Web server performance modeling," *Telecommunication Systems*, vol. 16, no. 3,4, pp. 361–378, 2001.

[27] J. Dilley, R. Friedrich, T. Jin, and J. Rolia, "Web server performance measurement and modeling techniques," *Performance Evaluation*, vol. 33, pp. 5–26, 1998.

[28] K. Kant and C. R. M. Sundaram, "A server performance model for static web workloads," in *2000 IEEE International Symposium on Performance Analysis of Systems and Software*, 2000, pp. 201–206.

[29] L. Cherkasova and P. Phaal, "Session-based admission control: A mechanism for peak load management of commercial web sites," *IEEE Transactions on computers*, vol. 51, no. 6, pp. 669–685, June 2002.

[30] J. Beckers, I.Hendrawan, R.E.Kooij, and R. van der Mei, "Generalized processor sharing performance model for internet access lines," in *9th IFIP Conference on Performance Modelling and Evaluation of ATM and IP Networks*, 2001, budapest.

[31] M. Andersson, "Performance modeling and control of web servers," Department of Communication Systems, Lund Institute of Technology, Tech. Rep. 160, 2004, lic. Thesis.

[32] M. Andersson, J. Cao, M. Kihl, and C. Nyberg, "Performance modeling of an apache web server with bursty arrival traffic," in *In Proceedings of International Conference on Internet Computing (IC)*, 2003.

[33] Y. Fujita, M. Murata, and H. Miyahara, "Performance modeling and evaluation of web systems with proxy caching," *Electronics and Communications in Japan*, vol. 86, no. 4, 2003.

[34] Y. Zhu and K. J. Lü, "Performance modelling and metrics of database-backed web sites," in *In Proceedings of 11th International Workshop on Database and Expert Systems Applications*, 2000, pp. 494–498.

[35] T. Abdelzaher and C. Lu, "Modeling and performance control of internet servers," in *Proceedings of the 39th IEEE Conference on Decision and Control*, 2000, pp. 2234–2239.

[36] T. Abdelzaher, K. Shin, and N. Bhatti, "Performance guarantees for web server end-systems: a control theoretic approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 80–96, January 2002.

[37] C. Lu, T. Abdelzaher, J. Stankovic, and S. So, "A feedback control approach for guaranteeing relative delays in web servers," in *Proceedings of the 7th IEEE Real-Time Technology and Applications Symposium*, 2001, pp. 51–62.

[38] N. Bhatti and R. Friedrich, "Web server support for tiered services," *IEEE Network*, pp. 64–71, Sept/Oct 1999.

[39] J. Carlström and R. Rom, "Application-aware admission control and scheduling in web servers," in *Proc. Infocom*, 2002.

[40] T. Voigt and P. Gunningberg, "Adaptive resource-based web server admission control," in *Proc. 7th International Symposium on Computers and Communications*, 2002.

[41] P. Bhoj, S. Ramanathan, and S. Singhal, "Web2k: Bringing qos to web servers," *HP Labs Technical report, HPL-2000-61*, 2000.

[42] N. Gandhi, D. M. Tilbury, Y. Diao, J. Hellerstein, and S. Parekh, "Mimo control of an apache web server: Modeling and controller design," in *Proceedings of the American Control Conference*, 2002.

[43] C.-J. Huang and C.-L. Cheng, "Application of support vector machines to admission control for proportional differentiated services enabled internet servers," in *Proceedings of the Fourth International Conference on Hybrid Intelligent Systems (HIS04)*, 2004.

[44] S. S. Jr, "Optimal control of admission to a queueing system," *IEEE Transactions on Automatic Control*, vol. 30, no. 8, pp. 705–713, August 1985.

[45] I. Bose and K. Altinkemer, "Design of a web site for guaranteed delay and blocking probability bounds," *Decision Support Systems*, pp. 131–140, 2004.

[46] S. C. Lee, J. C. Lui, and D. K. Yau, "A proportional-delay diffserv-enabled web server: Admission control and dynamic adaptation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 5, 2004.

[47] L. Liu and Y. Lu, "Dynamic traffic controls for web-server networks," *Computer Networks*, no. 45, pp. 523–536, 2004.

[48] S. Mahabhashyam and N. Gautam, "Dynamic resource allocation of shared data centers supporting multiclass requests."

[49] M. Kihl, "Overload control strategies for distributed communication networks," Department of Communication Systems, Lund Institute of Technology, Tech. Rep. 131, 2002, ph.D. Thesis.

[50] X. Chen, H. Chen, and P. Mohapatra, "Aces: An efficient admission control scheme for qos-aware web servers," *Computer Communications*, no. 26, pp. 1581–1593, 2003.

[51] S. Heine and M. Liljeström, "Kommunens hemsida vid en kris."

[52] L. Larsson, "Ministermordet," kBM:s temaserie 2004:4, Swedish Emergency Management Agency.

[53] N. R. Council, "Internet under crisis conditions: Learning from september 11," 2003.

[54] D. D. Clark, C. Partridge, R. T. Braden, B. Davie, S. Floyd, V. Jacobson, D. Katabi, G. Minshall, K. K. Ramakrishnan, T. Roscoe, I. Stoica, J. Wroclawski, and L. Zhang, "Making the world (of communications) a different place," End-to-End Research Group, Tech. Rep., 2005.