Simple Web Server Project

Andrew Siegle, Sam Kim, Richard Thai

|  |  |  |
| --- | --- | --- |
| Tactic | Attribute Affected | Status |
| Persistent Connection | Availability, Performance | Completed |
| Concurrency | Performance | Completed |
| Faster File IO | Performance | Completed |
| Revoke access from DOS attack | Availability, Security | Completed |
| 403 | Security | Completed |
| 401 | Security | Completed |
| 304 | BONUS, Performance, Availability | Completed |
| 505 | BONUS | Completed |
| Change Provider | BONUS | Completed |

# Github

<http://github.com/siegleal/477SimpleWebServer.git>

# Tactics Implemented

## Persistent connection

Allow a client to make multiple requests to the server without requiring connection each time. This reduces the number of requests needed by nearly 50%.

## Concurrency

For every request received, a new thread is started to handle it. In addition, new threads are starting for every Runnable task. This is done by passing threads to a singleton Executor interface.

## Faster File IO

Utilizes Java’s NIO library to perform connection handling in a more efficient manner. The traditional library relies upon blocking implementation, and requires a separate thread to manage each connection. The overhead of creating and switching between running threads makes such an architecture un-scalable. Java NIO, however, uses channels rather than threads to manage connections. This allows for a non-blocking implementation where a single thread can manage a large number of connections. From there, a separate thread can be run only when there is a transaction.

## Revoke access

The old system didn’t do anything if a client is connecting too many times in a given time period. The revoking access features implemented in the system allows the server administrators to see what IP’s are blacklisted and to reset or whitelist some IPs if they are known to be safe. The system works by using a specified sample and time-between-connections threshold to log connections by different IP’s and determine if malicious behavior is probable. If this is the case, the IP is automatically blacklisted and will not be able to connect with the server until it is removed from the blacklist.

## 401 / File permissions

The system allows the administrator to provide the server with a “passwd.txt” and a “permissions.txt” file in the server root which identifies the allowed users and their passwords, as well as files which require permissions and the users that can access them. This tactic improves security by restricting access to certain files. Before, any file was accessible to the outside.

## 403 / Forbidden

The system implements a 403 Forbidden HTML code when the user attempts to access the passwd or permissions file. This is put in place because the server by default returns any file in the root directory. This improves security by hiding files from the public and only allowing them to be accessed from the local machine.

## 304

Utilizing the “304 Not Modified” packet improves the system’s availability by reducing the need to re-transmit large files. Instead, the system looks at the HTTP header and determines if the client has the last updated version of the file and if this is the case, it will send back a 304 message instead of a large message containing the file again. It also improves the system’s performance (namely throughput) because it can avoid long delays as it loads large files, seemingly making the system faster and able to process many more requests per time period.

## 505

This is a bonus. It serves no major purpose in the system other than announcing to the client that the requested transaction cannot be completed because the server does not support the transaction type.

## Change Provider

This is a bonus. It serves no major purpose in the system other than saying who is providing the responses.

# Improvements

## Security

Before, the system was very vulnerable. It would accept all connections from all IP’s and let those connected clients access any file in the directory tree from the server root. We implemented 403, 401, and blacklists and whitelists to solve these problems. Blacklist and whitelists allow the administrator to dynamically allow and forbid ip’s from connecting. The file permissions (401) restricts access for specified files and users. 403 forbids everyone from accessing the passwd and permissions files to prevent anyone from acquiring access to those files.

## Performance

To improve the performance of the system, we implemented persistent connections to keep a host connected and reduce the number of packets being sent when an already-connected client requests a file. Concurrency was also implemented to improve throughput and latency since each client gets its own thread to run in. We also implemented faster file IO which is new to Java 7 and allows the server to retrieve files faster, improving throughput. Last-modified-since was the last implemented feature and improves latency and throughput since the server doesn’t waste time retrieving files that the client already has.

## Availability

The system was vulnerable to DOS attacks. To fix this issue, we implemented revoking access from clients if their time between connections falls below a certain threshold. We also implemented persistent connections to reduce the number of packets that need to be sent and make the system seem more available. Implementing the 304 also improves availability by reducing the clock time needed to process large requests.