Comp 4300 Project Report

Group (sole): Ryan Loeppky

Ideas?

Create a manual version of DNS to simulate recursive and non recursive solution to a problem in the system? This could be done with a Web Application that logs the time taken to move from page to page, and the time it takes the DNS to retrieve the address needed. How could web cache be implemented to best reduce the time.

Use a local cookie that holds the number of contacts made that gets recorded and reset on a completed request.

Have a single home page that contains multiple buttons that run certain simulations based on the button selected.

**Summary:**

**Problem**: Which is the faster method to retrieve the address for a website? Recursive or non-recursive or a unique mothed? Expanded: What is the most effective cache system to reduce the number of contacts between endpoints? Caches tested: FIFO, LRU, and a Unique method.

**Solution**: Create a simulation that simulates different types of cache methods for both recursive, non-recursive and a unique recursive methods to test and compare the results. Results will comprise of averages on time taken, number of connections required, and rate of cache hit rates to complete a DNS request.

**DNS pathing Options**:

**Recursive**: The Front DNS calls the Second DNS, which calls the Third DNS, which finally calls the Origin DNS.

**Non-Recursive**: The Front DNS calls the Second DNS to ‘get’ Third DNS connection. The Front DNS then calls the Third DNS to ‘get’ the Origin DNS connection, which the Front DNS finally calls the Origin DNS.

**Unique Recursive**: The Front DNS calls the Second DNS, which calls the Third DNS, which finally calls the Origin DNS. (Front and Second contain a cached immutable set of the x least recently used Domain Names and calls Origin directly if one was called)

**DNS caching Options**:

**None**: A set of tests exist for the DNS Pathing options that

**FIFO**: The Front DNS calls the Second DNS, which calls the Third DNS, which finally calls the Origin DNS.

**LRU**: The Front DNS calls the Second DNS to ‘get’ Third DNS connection. The Front DNS then calls the Third DNS to ‘get’ the Origin DNS connection, which the Front DNS finally calls the Origin DNS.

**Unique**: The Front DNS calls the Second DNS, which calls the Third DNS, which finally calls the Origin DNS. (Front and Second contain a cached immutable set of the x least recently used Domain Names and calls Origin directly if one was called)

Data results are stored in the execl file on GitHub, in the root folder, called Datagraphs.xlsx

Results:

**Technical components:**

**Instructions:**

* First, clone my repository onto your machine following GitHub’s instructions.
* Second,

I personally recommend using VS Code to open the project and run it as it is what I am using.

If I have any dependencies, then include the requirement to run “npm install” to setup the domain used.

Hopefully setup a website using typescript with npm start to run it locally. (Make multiple “Backends” to simulate DNS connections?)