Adaptive Proxy Networks

Authors: Ali Aburas, Rahul Gopinath

CS 533
Project Report

Introduction

HTTP Proxy servers are an integral part of the world wide web infrastructure. They serve in many roles including cache acceleration, spam blocking, access control for resources, controlled net access, as components of firewalls etc.Organizations typically employ a hierarchy of proxy servers that may be divided across offices, geographical regions, or may control access to certain resources. Since one of the main functions of proxy servers are to cache the responses and thus reduce the load on the network and origin servers, there are a variety of protocols available that allows a proxy to communicate with its peers and find out if another server has a cache of the current request available. However, most of these protocols like ICP and HTCP require additional network traffic to find where the cache might be. Others like CARP are static in that they cannot adapt to the changing network conditions like increase or decrease of load on a server. Our proposal is to develop a cache hierarchy that is able to adapt to changing conditions by making use of reinforced learning techniques.

Caching

A request starts its life at the browser, and if that browser is configured to use a proxy server, the request is forwarded that proxy server. This proxy server is the base proxy server in a caching hierarchy. From there, the proxy server tries to find the best possible route for the request so that it passes through the least loaded servers and reaches the required origin server. An origin server is the server the HTTP request is intended for. Once it reaches the server, the server generates the response which is sent down the same route. Depending on the settings in the request, response, the configuration of individual proxy servers, any or all of the proxy servers may decide to store the request.

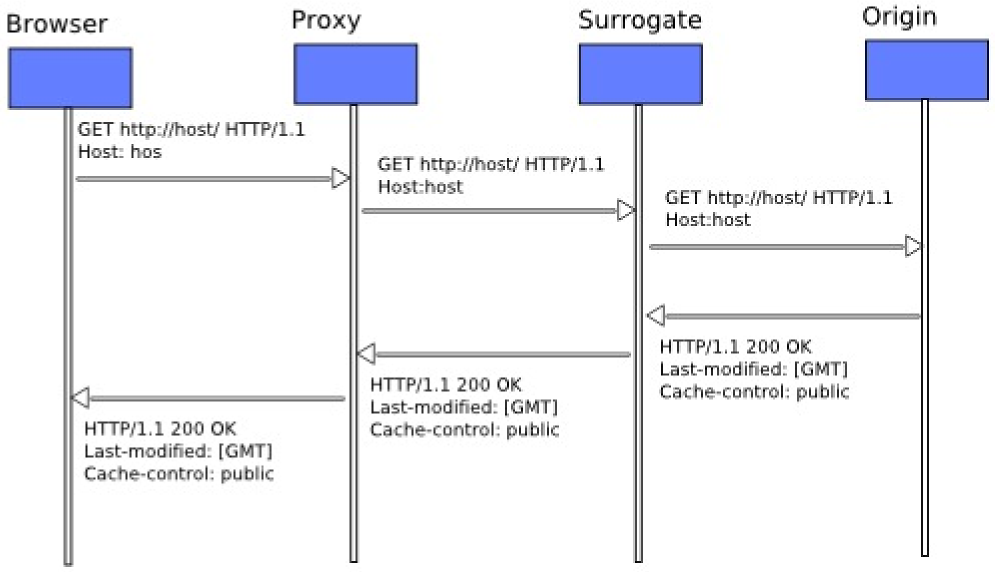

Figure-1

Next time a similar request comes to any of these servers, they may choose to serve the request from their own caches without going through the parent servers or the origin server.

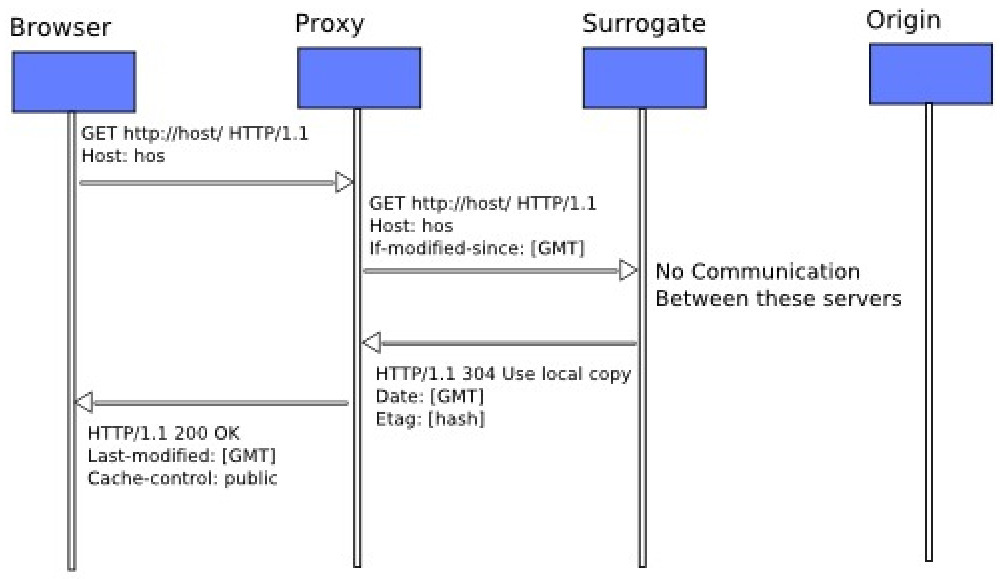

Figure-2

The usefulness of an HTTP Proxy Server is directly related to the amount of requests it is able to serve from its cache. This measure is called its cache hit ratio. If it is not able to serve the request from the cache, the next best option for the proxy server is to try and find a parent that has access to the resource, is lightly loaded and is probably caching the response.

Protocols for Caching

The usefulness of a proxy server is directly related to its ability to find the best possible route for a request that is directed towards it. There are many protocols that help the proxy servers to find the optimal path.

Internet Cache Protocol

ICP uses small messages to parent proxy caches to find if any of the parents have a cache of the request URL. If any of the parents respond, then the first parent to respond is used as the next hop address for the request. The protocol may also be configured to broadcast the ICP message instead of a limited number of servers. While this protocol is quite robust when the number of peers are low, and the number of levels in a proxy hierarchy is low, it begins to fail when the number of peers increases. This causes the number of ICP messages to increase exponentially. On the other hand ICP also relies on the first level parents caching the request. It is unable to take advantage of either a lower loaded parent without cached response or a route such that one of the higher level parents have the response in cache.

Hypertext Caching Protocol

The HTCP is similar to ICP. This protocol allows child proxy caches to monitor the caching store of their parent caches. It allows the full request and response headers to be used in cache management. While it addresses many of the known limitations of ICP, our criticisms on ICP especially that it looks only one level up still remain valid for this protocol.

Cache Digests

In this protocol, the proxy servers periodically exchange their cache indexes so that the other proxy servers know which proxy server to go for a particular request if it was already cached. The exchanged index uses a lossy compression technique called Bloom Filters which limits the size of the index while still providing some guarantee as to what URLs the cache contains. However it ignores the load of the proxy servers, more over it is unable to find an optimal path if none of the peers cache the request but one of the higher level proxy servers do.

Cache Array Routing Protocol

CARP uses a partitioned URL space to ensure that certain requests get served and cached by certain proxy servers. Even though CARP arrays factor in the load of a proxy server, it is not dynamic in that the magnitude of the load is captured in configuration files. Thus it is unable to utilize the optimal routing when the loading of proxy servers change dynamically.

Our Setup

We used a novel caching scheme in which the proxy hierarchy uses reinforced learning to dynamically learn the optimal network paths. We use Q learning because the individual proxy servers only need to estimate the Q values of their parents.

Simulator

We used a simulator to test our approach. Our simulator had the ability to dynamically vary the load. Each time a request came through a proxy server, the proxy server may increment or decrement the load by one point randomly. The servers also have the ability to decline a request if the load on it is high. This is very similar to the real-world scenario. In addition the server had a fixed limit on the number of cache entries.

The formulation of our approach is given below

Our MDP had these characteristics. For our experimentation, we found that these rewards gave the best learning curve after tweaking the rewards function for a little bit.

MDP<S,A,R,T>

1. State space - URL with a hash to domain names
2. Action - The server for the next hop

3. Reward
    a. -1 * load for each hop (load in percentage)
    b. 1000 points for reaching origin server
    c. 1000 points for a cache hit
    d. -1000 No Service - if a route cannot be found
4. Transition function - governed by the load on the server

Learning Parameters

We found that the reward points have to be in the same range for both negative and positive rewards or the algorithm did not learn effectively. For example increasing the weight of the load so that there was a higher negative reward for each hop resulting in the algorithm deciding to opt for No Service rather than to explore.

Assumptions

We used the simplifying assumption that proxy servers only talked to their parents. That is, they did not try peer to peer connections when a request failed. We also assumed that only the highest proxy servers had access to the origin servers. Another assumption was that the number of levels each request had to go through was constant - we assumed that the proxy hierarchy was a rectangular grid rather than a graph.

Solution

We used Q Learning to solve the MDP. The Q Learning is given by

$$Q(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t,a_t)}_{\text{learning rate}} \times \left[\overbrace{\underbrace{R(s_{t+1})}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q(s_{t+1},a)}_{\text{max future value}}}^{\text{expected discounted reward}} - \overbrace{Q(s_t,a_t)}^{\text{old value}} \right]$$

(image from wikipedia)

We chose the learning rate as 0.1 and the discounting factor as 1.0. Each proxy servers saved their own Q values for each domain and communicated them back to the child proxy servers when ever a response was routed through them.

Results

We present below the results of running our simulation with the above rewards and Q values.

Terminology

Degree: The degree of a proxy server is the number of parent proxy servers it has. The degree of a network of proxy servers is simply the average number of parents each proxy servers have
Peer size: The peer size or the width of a network is the average number of proxy servers at each level
Levels Height : The level height is the average number of hops a request has to go through before it is served by an origin server. Figure-3 shows all these terminologies

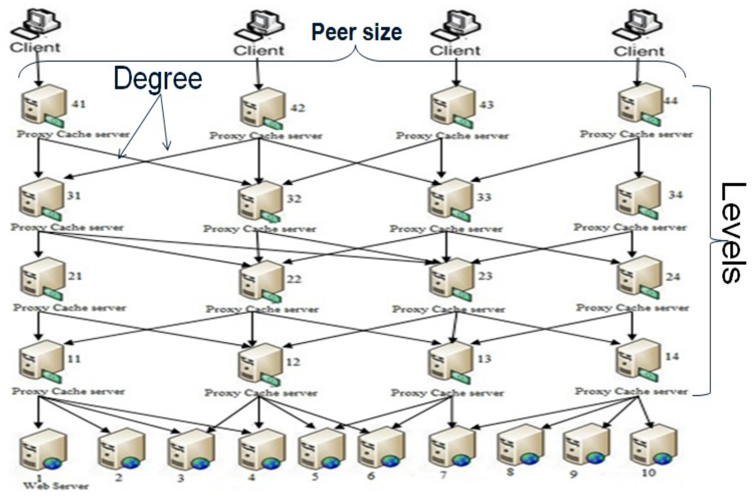

Figure-3

It should be noted that the main purpose of the simulation scenarios was to provide a framework to compare the performance of our Q-learning based reconfiguration of the topology of a network.

Varying levels

The performance of our network system has been evaluated with different levels and fix the number of degree and peer size. Figure-4 shows that even though the network complexity is increased for networks with multiple levels,this result can be explained that each new level introduces new paths to nodes.

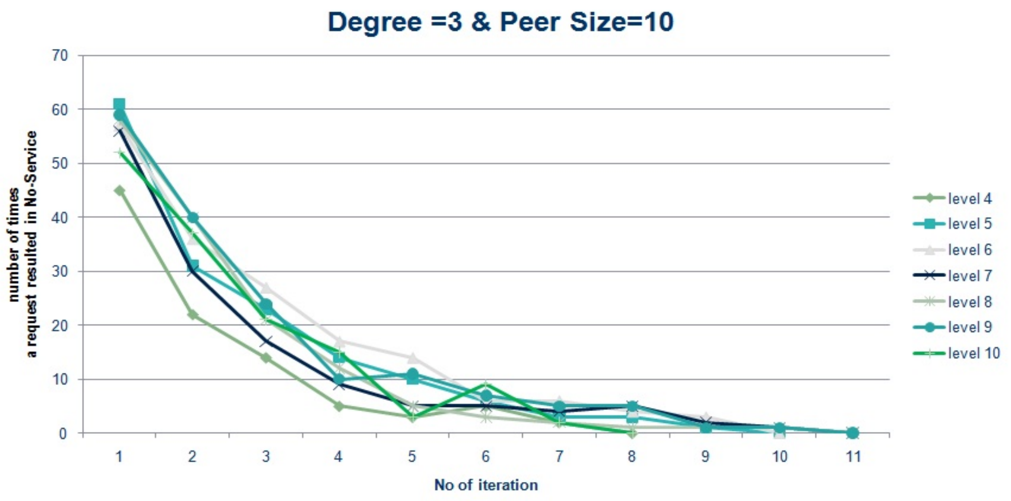

Figure-4

Varying peer size

In this analysis, the number of peer size is assumed to be random, the number of levels and degree are fixed. From the results represented in Figure-5 shows unlike adding new levels, it has an adverse impact on the .

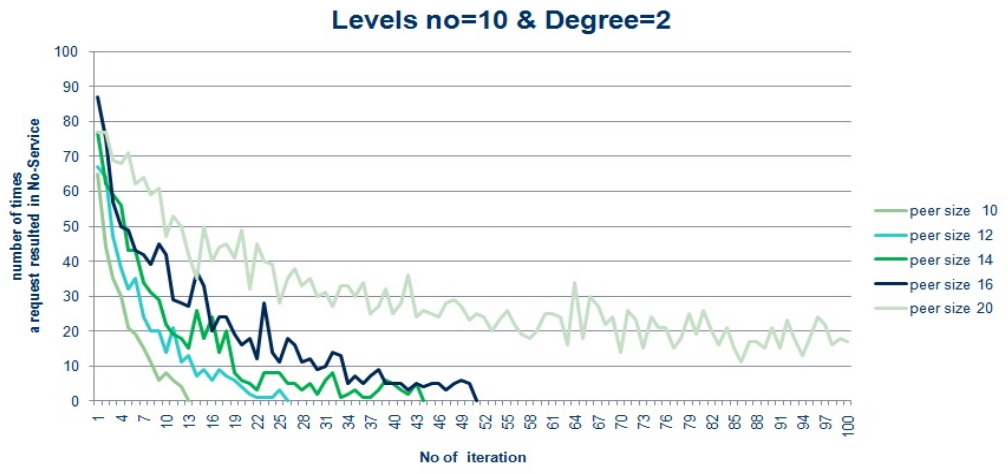

Figure-5

Varying degree

In this analysis, the number of degree is assumed to be random, the number of levels and degree are fixed. From the results represented in Figure-6 its clear that there is an impact on the Q-learning by adding the number of parent proxy servers it has.

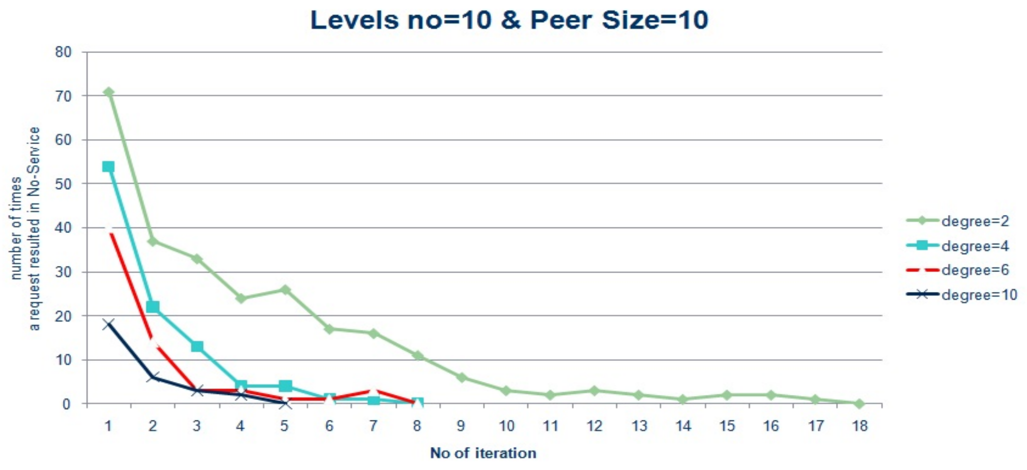

Figure-6

Conclusion

We found that Reinforced Learning with Q-Learning could indeed lead to better optimized networks. With our simulation we were able to consistently learn and adapt to any kind of network with dynamically changing loads. Thus RL learning was found to be a good fit for the HTTP Proxy server domain.

References

HTTP RFC http://www.w3.org/Protocols/rfc2616/rfc2616.html
ICP RFC http://icp.ircache.net/rfc2186.txt

CARP RFC http://icp.ircache.net/carp.txt
HTCP RFC http://icp.ircache.net/htcp.txt
Cache Digests http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.4810
Q-Learning http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=7548