**A First Look at the Topological Patterns**

**in Web Browsing Behaviors**

Utkarsh Goel, Clint Cooper, Brittany T. Fasy, and Mike P. Wittie

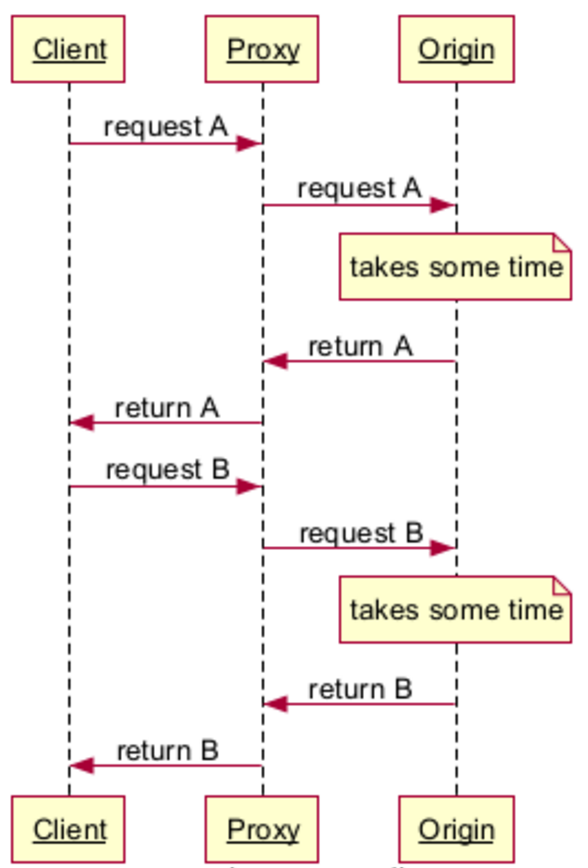
Department of Computer Science, Montana State University, Bozeman, MT

*{utkarsh.goel, clint.cooper, brittany, mwittie}@cs.montana.edu*

*(Project Proposal for CSCI 591)*

**1. Problem Statement**

It is challenging to accurately identify Internet-wide Web access patterns. This is because different people may have both similar and different interests in content at any given point in time. As a result, it remains unclear as to what degree people may share common interests on the Internet. Consequently, application developers lack strong insight of how their users request the data on the Web, and also whether common patterns exist among interests of different users. We argue that if application developers are made aware of such knowledge before their users begin to request content, end-user experience can be improved.



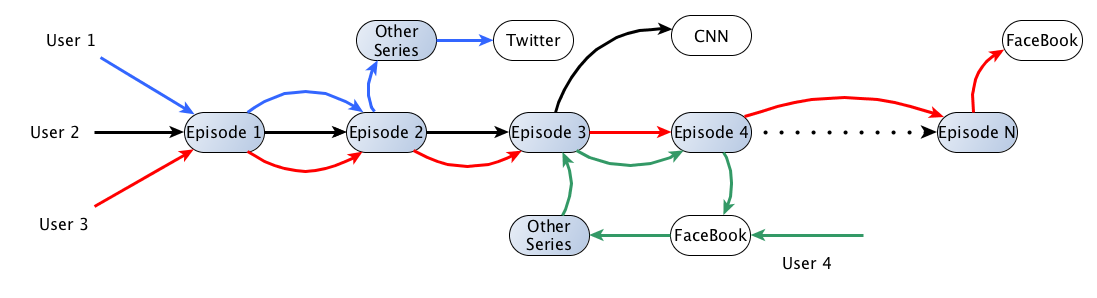
*Figure 1. A request-response model when the Proxy does not have requested objects in its local cache.*

For example in Figure 1, a client browser wishes to download Web objects A and B from a proxy to complete the loading of the webpage. When the request for object A reaches the Web proxy, the proxy searches for the object A in its local cache. If the object is available, the object is served to the client immediately. If the requested object is not available in proxy’s cache, the proxy then forwards the client request for object A to the origin server. The origin server then transfers the object A to the proxy. After receiving the object, the proxy sends the object to the client. The client then requests the object B from the proxy. Similarly to how object A was retrieved, object B is retrieved in the same manner.

Although the client received both objects A and B, client has to wait for both objects to be first downloaded by the proxy from the origin server. In a Content Delivery Network (CDN) environment, proxies are much closer to the clients than the origin servers. Therefore, if each object has to be downloaded from origin server, clients may have to wait a long time, which may eventually lead to user frustration with how the website is loaded. On the other hand, if the request data is already available in the proxy’s cache, clients can download the data immediately, resulting in faster load of the website. However, pro-actively predicting user’s next request remains challenging, because user browsing behavior may be unpredictable.

**2. Proposed Solution**

We understand and acknowledge the fact that it remains challenging for application developers to reliably predict the user behavior, however, we argue that efforts can be put in approximating user’s browsing behavior with some certainty, based on available historical data on how different people browse the Web.

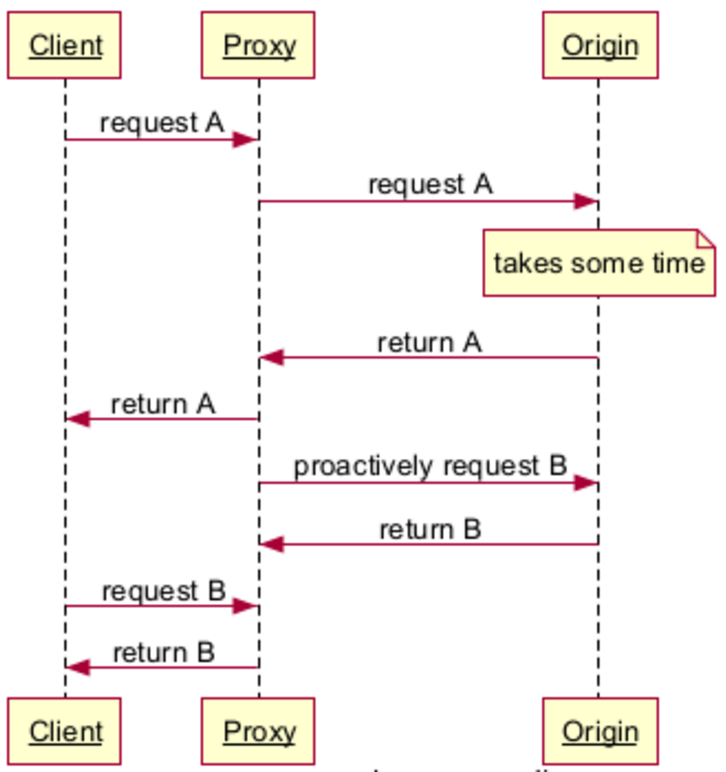
*Figure 2. A contrived topology of how different users may browse the Web.*

For example, in Figure 2, we show what some users’ browsing behavior may look like. Specifically, for some browsing behaviors, we show that User 1, User 2, and User 3 to all start their browsing session by requesting Episode 1 at Netflix.com. User 1 watches the first two episodes and then decides to watch a different series on Netflix.com, ultimately ending her browsing session at Netflix.com and requesting the homepage of Twitter.com. User 2 behaves similarly, but requests the homepage of CNN.com after watching Episode 3. User 3 watches all available episodes of the same series one after another and then requests the homepage of FaceBook.com. User4 currently on FaceBook.com finds an interesting article on Episode 3, making him to end her browsing session on Facebook.com and requesting Episode 3 on Netflix.com. User 4 also requests Episode 4 after watching Episode 3, after which she ends her browsing session on Netflix.com and returns to Facebook.com.

Based on the above example, we argue that similar Web browsing behaviors may exist on a large scale. Specifically, we list the contributions of our proposed work as follows:

* We propose to investigate on a large scale whether topological models that represent mutual interests exist in the real world.
* We also seek to identify overlapping patterns (if any) in how different users request Web content. Further, we hope to cluster these overlaps to identify generalized behavioral patterns. For example, a tendency to visit social media after reading the news.
* Finally, we seek to investigate how popular a given series of requests is across different users. For example, in Figure 2, we refer to “requesting Episode 2 after Episode 1” as a connection between Episode 1 and Episode 2 with a weight of three. Weight of a connection is calculated based on the number of different users that share a common interest. Therefore, we seek to identify how many sequential connections different users share, as well as, how stable each sequence is over time.

**3. Expected Impact**

We believe identification of mutual interests among different users can enable application developers to proactively predict user’s subsequent requests with some certainty, making the data available for the user before it is even requested. More specifically, we argue that using the knowledge of different topologies that may exist in Web browsing behaviors, application developers may perform data relocation from their origin servers to their proxy servers.

*Figure 3. A request-response model when the Proxy proactively caches the content for client’s subsequent request.*

For example, in Figure 3, if one can determine a topological connection between objects A and B (that clients generally request object A followed by object B), the proxy can proactively fetch object B from the origin server before the client requests it. Such logic will allow the proxy to serve the request for object B from its local cache, resulting in lesser wait time for the client while the webpage loads.

Performance overhead if we increased caching that no one will be requesting or that will be invalid by the time that it is requested. We can’t cache everything and we can’t determine what we should cache currently.