Assignment #9: Semantic Hypermedia

Advanced Software and Systems Engineering - HS 2023

Submitted by:
Michael Bruelisauer, Daniel Leal, Kaan Aydin, Stephan Nef

Submission Date:
December 23, 2023

Task 1: Semantic Hypermedia Overlays

To accomplish the tasks this week, we communicated with the Interoperability Task Force to coordinate decisions across the system. Below we document those decisions and our critique of them.

1.1 Discussion & Decisions

The group actively worked towards creating a semantic hypermedia overlay, with a particular focus on the discovery and linkage of auction feeds using typed links and semantic annotations. There was a commitment to a dynamic approach where auction houses declare their feeds and interests, facilitating crawling and discovery. We collectively decided on the use of link headers to communicate auction feed URIs and have considered the need for descriptive annotations to enhance clarity.

The group discussed the importance of a discovery mechanism that allows clients to find auction feeds dynamically. This led to the implementation of a `/discover/` endpoint that provides a list of links to other auction houses, with semantic information about the types of auctions they publish.

The conversation shifted from a static approach, where auction types are predefined, to a more dynamic one where auction houses can specify what types they will publish at runtime, enhancing the discovery aspect of the system. The group decided to make each auction house capable of defining its auction types and interested areas, ensuring interoperability and flexibility within the ecosystem. Each auction house starts with its registry entry, and as the crawling progresses, it discovers the offerings of other auction houses. This approach is more aligned with how crawling typically works on the web.

For the topology of the semantic hypermedia, a brute-force approach was adopted, where each group queries every other. Please note that this is our current preliminary understanding and will be discussed further with the interoperability task force. Upon reflection, we acknowledge that while functional, this method may not be in the true spirit of semantic hypermedia, which thrives on the efficient and intelligent networking of resources.

1.2 Critique

Our critique is twofold: it acknowledges the strides we have made while scrutinizing our approaches with a critical eye, particularly in terms of system complexity and topology efficiency.

Complexity of /discover/ Endpoint: The addition of the `/discover/` path has been critiqued for adding complexity without clear justification, given the existing GET requests that already facilitate the discovery of auction feeds.

Brute-force Topology: Our brute-force approach to the semantic hypermedia topology, though operational, is recognized as being inefficient. This strategy is counterintuitive to the fundamental principles of semantic hypermedia, which aim to streamline and intelligently manage resource connectivity.

Task 2: Hypermedia-based Discovery of Auction Feeds

Semantic hypermedia is a concept in the field of web development and software architecture. It extends the traditional idea of hypermedia, where resources are interconnected through hyperlinks, by adding semantic information to these links. This semantic enrichment enables clients (software applications) to understand the meaning or context of the links, allowing for more intelligent navigation and interaction with web resources.

Semantic hypermedia extends beyond the traditional concept of hypermedia, which primarily focuses on interconnecting resources through hyperlinks. While adding semantic information to these links is a crucial aspect, semantic hypermedia encompasses a more comprehensive approach. It involves creating semantic representations of the resources themselves, not just the links. This is typically achieved using technologies like RDF (Resource Description Framework) and Web ontologies. These semantic representations enable a deeper understanding and contextualization of the resources, allowing for more intelligent navigation and interaction within the web environment. By embedding semantic data within the resources and their interconnections, semantic hypermedia enhances the ability of clients (software applications) to interpret and interact with web resources in a meaningful way, based on their content and context. This approach leverages the principles of the semantic web, where information is interconnected not just through hyperlinks, but in a way that is meaningful and understandable to machines, thereby facilitating more intelligent and context-aware web applications.

In a semantic hypermedia system, clients don't just passively follow hyperlinks; they interpret the semantics of these links to make decisions about which resources to interact with, based on their current needs or goals. This approach leverages the principles of linked data and the semantic web, where information is interconnected in a meaningful and machine-understandable way.

Our implementation of semantic hypermedia in our Java application involves dynamically discovering and interacting with resources (in this case, auction feeds) based on the semantics of the hypermedia controls provided. This typically includes:

- Dynamically parsing hypermedia controls like HTTP link headers to understand the types of resources they reference.
- Making decisions based on the semantic information extracted from these controls, such as what type of content is available at a particular link and whether it aligns with the client's current interests or goals.
- Efficiently navigating through a network of resources (e.g., different

auction feeds) by following links that are semantically relevant to the client's objectives.

Our approach embodies these principles by crawling through auction feeds, interpreting the semantic content of link headers, and subscribing to topics of interest. In detail, the process works as followed:

2.1 Response Link Header:

First, every auction house provides a link header in the response of the `GET /discover/` (or `GET /auctions/`) request that contains an auction house URI of another auction house, together with the topics that they publish auctions for. The semantic annotation is currently defined as follows: `"<auction-house-uri>; types={type1,type2}"`. It was also decided by the interoperability task force to add your own auction house URI with your topics, so that once someone calls your auction house, the calling auction house can double-check if my topics that are advertised by the previous auction house are true. In fact, the previous auction house would not even necessarily need to advertise the next auction house's topics, as the next auction house's topics can be retrieved when calling it and then they are guaranteed to be true.

The trade-off involved in not having the topics of the current auction house advertised by the previous one is a balance between ensuring data accuracy and managing network efficiency. When each auction house directly provides its topics, it guarantees the information is current and accurate, which is vital in a dynamic environment where auction topics may frequently change. This direct approach reduces the risk of propagating outdated or incorrect information that might have been advertised by a previous auction house. However, this method requires making additional HTTP requests to each auction house to discover their topics. This increase in network requests can lead to higher network traffic and potential latency issues. Each auction house needs to be queried individually, which could introduce delays in the crawling and discovery process, as the system waits for responses from each auction house. Therefore, while this approach offers the advantage of up-to-date and accurate information about auction house topics, it may come at the expense of increased network usage and potential latency in the system's response times.

2.2 Crawling:

For the crawling, we first call another auction house that is predefined as an entry point. In the response, we get two link headers, one with auction house URI + topics from the currently called auction house, as well auction house URI + topics from a next auction house. The semantic annotation from these link headers is as described above. We first retrieve the topics from the first link header (that relates to the current auction house), check the current topics and if they match any of the topics we are interested in, we subscribe to this auction house. Once we subscribed, we remove this topic from our topics interest list. We then add the auction house to the list of crawled auction houses.

After that, we repeat this process with the next auction house that we got the URI and topics from in the second link header. This continues until we either found an auction house for all topics in our interest list or if we crawled all available auction houses or if an auction house does not advertise another auction house in their response headers when calling it.

2.3 Summary

Our approach parses and processes link headers, identifying and subscribing to auction feeds based on specified topics. It embodies a practical application of semantic hypermedia, dynamically navigating through a network of auction feeds in a context-aware manner. During the crawling process, the class maintains a queue of URIs and a list of checked auctions, ensuring efficient resource discovery without redundant revisits. The implementation returns a list of subscribed auction feeds after completing the crawl, triggering subsequent subscription processes.

2.4 Critique

While the crawler effectively navigates and subscribes to relevant topics, it encounters certain challenges. First, distinguishing between the link header advertising the current hub's topic and the link to the next auction feed requires additional processing. This is due to the lack of distinctiveness in the link headers. A potential improvement could involve using a custom header field to advertise the current auction house's topic, reducing client-side processing complexity.

Additionally, the crawler's mechanism to decide the next URI to crawl can be further optimized. It currently compares the current request URI with advertised URIs to find the next link, which might not be the most efficient approach in complex networks. Enhancing this logic could streamline the crawling process.

The process, however, acknowledges certain limitations in its current form. The inability to handle multiple outgoing links is noted; the crawler prioritizes the first link differing from the current request URI. Future enhancements may include a more robust mechanism to manage multiple links, potentially employing a queue system to track and crawl through a broader range of links.

Task 3: On Decentralization

Our implementation exhibits several notable architectural characteristics and implications.

Scalability and Flexibility The design of the crawler, which dynamically discovers and interacts with auction feeds, promotes scalability. This decentralized approach allows for efficient scaling as new auction houses are integrated, and offers greater flexibility in adapting to changes in the network.

Increased Complexity and Maintenance Challenges The reliance on semantic hypermedia controls and dynamic discovery significantly increases the system's complexity. Understanding and maintaining such a system requires in-depth knowledge of semantic hypermedia principles and the specific implementation details.

Latency and Responsiveness Dependent on network communication for accessing and interpreting link headers, the crawler's operation might introduce latency issues, affecting the system's responsiveness. Although asynchronous communication can mitigate this, it remains a key consideration in a distributed environment.

Fault Tolerance and Single Points of Failure The decentralized nature of the crawler reduces the risk of a single point of failure, enhancing the overall fault tolerance of the system. If one auction feed becomes unavailable, the crawler can continue operating by accessing other feeds.

Data Consistency and Reliability Ensuring data consistency across different auction feeds in a decentralized system can be challenging. The crawler might encounter discrepancies, impacting the system's reliability. Implementing mechanisms to verify and ensure data consistency is crucial.

Decoupling and System Evolution The crawler's design follows the principles of loose coupling, facilitating easier modifications and evolution of the system. This decoupling means changes in one part of the system are less likely to impact others, allowing for smoother system evolution.

While our approach enhances scalability, flexibility, and fault tolerance in a semantic hypermedia-driven system, it introduces challenges in terms of complexity, error handling, data consistency, and reliability. Addressing these challenges in future enhancements could optimize the system's performance and robustness.