EchoChamber: Rule-Based Semantic Webhooks

Jackson Morgan

Georgia Institute of Technology, Atlanta GA 30332, USA
jmorgan45@gatech.edu

Abstract. The semantic web is well-suited for the storage and access of static information. However, the value in many datasets derives from being current and up-to-date. A service interested in such data could periodically query a SPARQL Protocol and RDF Query Language (SPARQL) enabled database to check for updates, but periodic querying is taxing to the receiving database and only guarantees new data is received within the intervals of querying. Time-sensitive datasets are often served by event-driven architectures that allow interested systems to be alerted every time information is updated. In this poster I outline EchoChamber, an event-driven, webhook-based implementation of a Web Ontology Language (OWL) triplestore using Semantic Web Rule Language (SWRL) rules as event triggers.

Keywords: Event-Driven, SWRL, Webhook.

1 Introduction

Event-driven architectures [1] are the method of choice when building systems to handle data that is quickly updated and requires external systems to be alerted. The need for a service to periodically query a data source to remain up-to-date is subverted by event-driven architectures, thus saving valuable computing resources.

One popular practice in implementing event-driven architectures is webhooks [2]. Under a webhook implementation, when a data source wants to emit an update to an interested service, it sends an HTTP request to a route at the service's domain or IP address that is provided upon configuration.

Webhooks have the potential to transform the way semantic web services interact with the world around them. Yet, webhooks and semantic resources have disparate paradigms that require creative solutions. Mainly, webhooks are often triggered by well-defined events. For example, when a repository on Github receives a commit it can trigger a webhook. Conversely, webhooks for a semantic web resource should be flexible, allowing for user-defined triggers for any kind of update that might be made to a dataset.

To address this problem, I propose EchoChamber, a triplestore implementation that takes inspiration from a solution for semantic web access control [3]. EchoChamber is designed to easily integrate with current standards and proposals for the semantic web including SWRL rules, [4] which allow for user-defined flexible triggers, and Linked Data Notifications, [5] within which an EchoChamber server would serve as a "sender."

2 Design

To understand the implementation of EchoChamber, consider the hypothetical graph in Figure 1. Three users exist in this scenario, two of which, User A and User B, are members of separate chat rooms, chat room A and chat room B respectively. A third user, User C, can be described as an "Admin" role. Each chat room has a variable number of messages that can be added or deleted. Finally, each user has a "chat webhook," a URL that should be called if ever an EchoChamber event is encountered. These routes presumably live on a server that will send the update to the user's device.

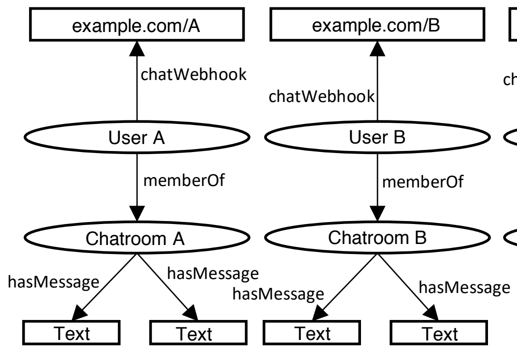

Fig. 1. A hypothetical graph describing users and their relationships to various chat rooms.

Our chat room application has two simple rules that should be provided to EchoChamber. 1) If a message is added or removed in a chat room within which a user is enrolled, alert the user via their corresponding webhook, and 2) admins should be alerted to all message changes regardless of chat room membership.

When a triple is added or removed from the triplestore, EchoChamber represents that change as a graph that is integrated with the main triplestore. Figure 2 displays a graph representing a change adding the string "Hello" to chat room B via the "hasMessage" property.

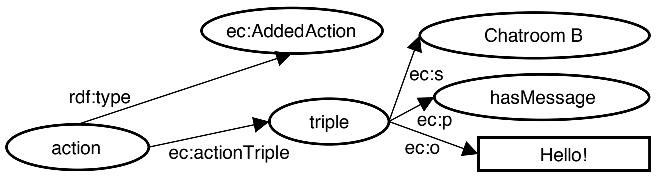

Fig. 2. A graph representation created by EchoChamber to represent the adding "Hello" to chat room B.

After creating the graph representation of the change, EchoChamber applies the SWRL rules to modify the object. Based on the two rules outlined above, Figure 3 represents the rules that would exist for our hypothetical scenario.

If a message is added or removed in a chat room within which a user is enrolled, alert the user via their corresponding webhook:

```
ec:AddedAction(?action)
∧ ec:actionTriple(?action, ?actionTriple)
∧ ec:p(?actionTriple, <hasMessage>)
∧ ec:s(?actionTriple, ?chatRoom)
∧ memberOf(?user, ?chatRoom)
∧ chatWebhook(?user, ?webhook)
⇒ ec:sholdUpdate(?action, ?webhook)
```

Admins should be alerted to all message changes regardless of chat room membership:

```
ec:addedAction(?action)
∧ ec:actionTriple(?action, ?actionTriple)
∧ ec:p(?actionTriple, hasMessage)
∧ Admin(?user)
∧ chatWebhook(?user, ?webhook)
⇒ ec:sholdUpdate(?action, ?webhook)
```

Fig. 3. The SWRL rules that define the desired webhook functionality of the hypothetical scenario.

If a rule's antecedent is satisfied, the action will now have a variable number "ec:shouldUpdate" properties that correspond with the URL that should be requested as outlined in Figure 4.

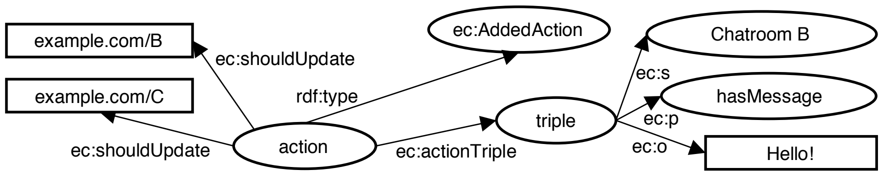

Fig. 4. The action graph after being augmented with the SWRL rules.

Finally, EchoChamber sends an HTTP request to each of the webhooks containing the action graph. The shouldUpdate properties are excluded for privacy reasons.

3 Implementation

EchoChamber is implemented in Java Spring using owlapi [6] and swrlapi [7]. The main datastore has a simple interface to include swrl rules and the ability to add triples as seen in Figure 5. After adding a tiple to the datastore, EchoChamber loops through all given rules and sends a request to the desired url.

One deviation of this implementation from the proposal above is the use of SQWRL queries in the stead of SWRL rules. When defining the query, a user is expected to include a `?webhook` variable that will include the url of the desired target. While it would be more advantageous to use SWRL rules for the sake of simplicity, it would require modifications to the core systems of swrlapi and owlapi as the object property `ec:p` which defines the predicate of an inserted triple is an object property that references another object property. Such a property does not currently exist in owlapi.

```
EchochamberDatastore datastore = new EchochamberDatastore("file.owl");
datastore.addRule(
    "ec:AddedAction(?action) ^ ec:actionTriple(?action, ?actionTriple) ^ " +
    "ec:p(?actionTriple, hasMessage) ^ ec:s(?actionTriple, ?chatRoom) ^ " +
    "memberOf(?user, ?chatRoom) ^ chatWebhook(?user, ?webhook)”
    “-> sqwrl:select(?webhook)");
datastore.addDataPropertyTriple("ChatroomB", "hasMessage", "Hello!");
```

Fig 5. An example use case for the EchoChamber datastore implemented in Java.

An implementation of EchoChamber's core features can be found at `https://github.com/jaxoncreed/echochamber.`

4 Further Considerations

Further work for EchoChamber includes efficiency improvements and the construction of a more complete interface for the datastore. Such an interface would include methods for removing triples and compatibility with SPARQL update queries while still maintaining the ability to send webhook requests based on updates.

A major inefficiency in EchoChamber is the need to loop through all rules given to the datastore to collect webhook urls. Given a datastore with millions of rules, this would be quite taxing to do on every update. Therefore, a scheme to retrieve SWRL consequents through indexing antecedents as boolean expressions [8] would greatly improve the implementation's efficiency.

References

1. Event-Drive Architecture Overview, http://elementallinks.com/el-reports/EventDrivenArchitectureOverview_ElementalLinks_Feb2011.pdf, last accessed 2018/6/1.
2. Web hooks to revolutionize the web, http://progrium.com/blog/2007/05/03/web-hooks-to-revolutionize-the-web, last accessed 2018/6/1.
3. Kagal, L., Finin, T., Paolucci, M., Srinivasan, N., Sycara, K., Denker, G.: Authorization and privacy for semantic web services. IEEE Intelligent Systems 19(4), 50-56 (2004).
4. Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabel, S., Grosof, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission, 21 (2004).
5. Capadisli, S., Guy, Amy.: Linked Data Notifications. (2017)
6. The OWL API, http://owlcs.github.io/owlapi, last accessed 2018/7/17
7. SWRLAPI, https://github.com/protegeproject/swrlapi, last accessed 2018/7/17
8. Whang, S. E., Garcia-Molina, H., Brower, C., Shanmugasundaram, J., Vassilvitskii, S., Vee, E., & Yerneni, R.: Indexing boolean expressions. Proceedings of the VLDB Endowment, 2(1), 37-48 (2009).