WebGC: Browser-Based Gossiping [Live Demo/Poster]

Raziel Carvajal-Gómez, Davide Frey, Matthieu Simonin, Anne-Marie Kermarrec

WebGC: Browser-Based Gossiping

[Live Demo/Poster]

Raziel Carvajal-Gómez, Davide Frey, Matthieu Simonin, and Anne-Marie Kermarrec
name.surname@inria.fr

INRIA Rennes, France

ABSTRACT

The advent of browser-to-browser communication technologies like WebRTC has renewed interest in the peer-to-peer communication model. However, the available WebRTC code base still lacks important components at the basis of several peer-to-peer solutions. We tackle this problem by proposing WebGC, a library for gossip-based communication between web browsers. Due to their inherent scalability, gossip-based, or epidemic protocols constitute a key component of a large number of decentralized applications. WebGC thus represents an important step towards their wider spread.

Categories and Subject Descriptors

I.5.3 [**Clustering**]: Algorithms; D.4.7 [**Organization and Design**]: Distributed systems

General Terms

Design

Keywords

Real Time Communication, Peer-to-Peer, Gossip based algorithms

1. INTRODUCTION

Epidemic (gossip) protocols have received enormous attention by the research community due to their inherent scalability, resilience to failures, and wide applicability. Recent work has exploited epidemic protocols in a variety of decentralized applications ranging from video streaming to recommendation systems [4]. Such applications follow a peer-to-peer interaction model and require users to install a specific piece of software [7] to handle peer-to-peer communication. In a world where more and more applications run within web browsers, this represents a clear disadvantage for the success of epidemic protocols.

Our demo and poster address this concern by presenting WebGC (Web Gossip Communication), the first—to the best of our knowledge—browser-based library for epidemic protocols. WebGC allows developers define new gossip protocols in a few lines of Javascript, and to assemble them through a simple JSON (Javascript Object Notation) declaration. Both the protocols and the JSON declaration are included in a web page and fed to client browsers that can then start communicating in a peer-to-peer fashion. WebGC builds on the recent introduction of WebRTC [1], and on two related projects: PeerServer and PeerJS [2]. WebRTC extends browsers with real-time communication (RTC) capabilities. PeerServer bootstraps the connections between peers, while PeerJS provides high-level communication primitives that serialize messages through WebRTC streams.

Middleware'14: Demos and Poster Dec 8 - Dec 12, 2014, Bordeaux, France

DOI: http://dx.doi.org/10.1145/2678508.2678515

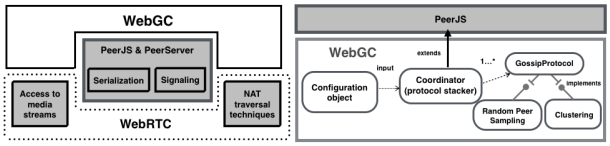

Figure 1: General architecture

Our demo will feature a protocol stack building an interest-based topology [4]. Participants will connect to a demo server with their own laptops and will enter keywords representing their interests on a web page. Their browsers will then initiate gossip communication and build a semantic overlay—grouping users with similar interests—while visualizing its evolution in real time.

2. WEBGC LIBRARY

The left of Figure 1 positions WebGC in the context of PeerServer, PeerJS, and WebRTC. WebGC uses a customized version of PeerServer to bootstrap gossip protocols. It then translates the peer identifiers, obtained from PeerServer or through the gossip protocols themselves using ICE (WebRTC's NAT traversal framework). Finally, it uses primitives provided by PeerJS to transfer data to and from other peers.

WebGC Internals.

The right of Figure 1 depicts WebGC's architecture. Its core consists of a COORDINATOR object inheriting from PeerJS. The COORDINATOR instantiates the gossip protocols and acts as a communication broker dispatching incoming messages to the various protocols. The library currently includes the implementation of two peer sampling protocols, CYCLON [10] and the generic protocol suite from [5], as well as a clustering protocol [11, 4]. All protocols implement a GOSSIPPROTOCOL "interface"—since Javascript does not natively support interfaces, we adopt the interface pattern described in [6]. The COORDINATOR makes it possible to stack these protocols on top of each other [4] to implement applications.

The GOSSIPPROTOCOL interface follows the scheme proposed in the literature [9, 8, 5] and defines the high-level operations that

```
<html>
...
<script type="text/javascript" src="http://cdn.peerjs.com/0.3/peer.js" />
<script type="text/javascript" src="./webgc.min.js" />
<script>
  var configurationObj = {
    peerServer: { ... },
    gossipAlgos: {
     cyclon1: {
       class: 'Cyclon',
       fanout: 5,
       ...
     vicinity1: {
       class: 'Vicinity',
       propagationPolicy: { push: true, pull: false},
       ...
</html>
```

Figure 2: Code snippet from our demo

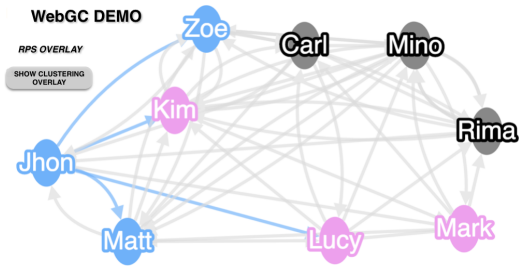

Figure 3: Screenshot of resulting RPS overlay

constitute a gossip-protocol. Developers can use the protocols provided by the library, but they can also implement the operations in GOSSIPPROTOCOL to define new protocols.

Building Applications with WebGC .

Regardless of the protocols it uses, a WebGC application consists of an HTML file containing a user interface and the declaration of a configuration object. Figure 2 shows a snippet from the HTML file used in our demo. The interface allows users to enter their interests and displays the current state of the overlay as viewed by the user's browser. The HEAD section of the file includes the WebGC library, while the body includes the web-page content and the configuration object. This consists of a JSON declaration. It groups information about the PeerServer, as well as a list of protocols with their parameters and their interconnections.

3. DEMO TIMELINE AND POSTER

Our demo machine (a laptop) will run the PeerServer, ICE servers (STUN and TURN), and web server. Participants will connect to the web server with their own laptops and will access a web page where they will enter a name and keywords representing their interests. Their browsers will then start interacting in a peer-to-peer fashion, running an RPS [5] and a clustering [11] protocol. This will group participants according to their declared interests and will display the resulting overlay portion in graphical form. Each participant's browser will display a local view of the overlay centered on the associated user. However, participants will also be able to view the global evolution of the network on the bootstrap server. This latter visualization will run outside of the peer-to-peer protocol.

4. RELATED WORK

WebGC orchestrates the execution of gossip protocols within a web browser by exploiting WebRTC [1], and the PeerJS/PeerServer pair [2]. WebRTC was introduced by Google to enable Real Time Communication between web browsers and is currently being standardized by the W3C and the IETF. Essentially, it provides in-browser applications with access to media, and NAT traversal. PeerServer provides signaling, while PeerJS implements serialization and provides high-level communication primitives for peer-to-peer applications. WebGC exploits these tools to provide an easy-to-use programming framework for gossip-based applications.

Although WebGC constitutes, to the best of our knowledge, the first gossip-oriented library over WebRTC, a lot of existing work has concentrated on the modular implementation of gossip protocols [8, 5, 9, 3]. The authors of [8] propose a decomposition of gossip applications into standard building blocks. [5] present a family of random-peer sampling protocols by highlighting the common structure exhibited by gossip-based overlay-maintenance solutions. Finally [9] and [3] propose two libraries for the development and deployment of gossip-based protocols. However, both libraries only provide a Java implementation and do not support operation within a web browser.

5. CONCLUSIONS

Web browsers have became powerful enough to support applications as complex as desktop applications. The emergence of platforms such as HTML5/Javascript and WebRTC have even suggested the idea of a web-based operating system. In this context, this demo and poster fill an important gap by showcasing a library for the deployment of gossip-based applications within web browsers. The demo demonstrates the effectiveness of our library in a real context, while the poster provides details on the structure of the library, on how to build applications, and on how to implement new protocols. We believe that our framework may give a second wind to gossip-based peer-to-peer applications. Yet it also needs to evolve. WebGC currently relies on WebRTC's heavyweight connection-setup. We are therefore investigating how to replace this standard signaling mechanism with cheaper fully decentralized solutions.

6. REFERENCES

[1] `http://www.webrtc.org/`.

[2] `http://peerjs.com/`.

[3] `http://gossiplib.gforge.inria.fr/`.

[4] M. Bertier, D. Frey, R. Guerraoui, A. Kermarrec, and V. Leroy. The gossple anonymous social network. In *Middleware 2010, Bangalore, India, 29/11 - 3/12, 2010.*, pages 191–211, 2010.

[5] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM TOCS*, 25(3), 2007.

[6] A. Osmani. *Learning JavaScript Design Patterns*. JavaScript and jQuery developer's guide. O'Reilly Media, Inc., 2012.

[7] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. van Steen, and H. Sips. Tribler: A social-based peer-to-peer system. *Concurrency and Computation: Practice and Experience*, 20:127–138, February 2008.

[8] E. Rivière, R. Baldoni, H. Li, and J. Pereira. Compositional gossip: a conceptual architecture for designing gossip-based applications. *SIGOPS Operating Systems Review*, 2007.

[9] F. Taïani, S. Lin, and G. S. Blair. Gossipkit: A unified componentframework for gossip. *IEEE Trans. Software Eng.*, 40(2):123–136, 2014.

[10] S. Voulgaris, D. Gavidia, and M. van Steen. CYCLON: inexpensive membership management for unstructured P2P overlays. *J. Network Syst. Manage.*, 13(2):197–217, 2005.

[11] S. Voulgaris and M. van Steen. Epidemic-style management of semantic overlays for content-based searching. EuroPar 2005, Lisbon, Portugal, pages 1143–1152.