Content-centric User Networks: WebRTC as a Path to Name-based Publishing

Christian Vogt, Max Jonas Werner, Thomas C. Schmidt
Department of Computer Science
Hamburg University of Applied Sciences, Hamburg, Germany
christian.vogt@haw-hamburg.de, maxjonas.werner@haw-hamburg.de, t.schmidt@ieee.org

***Abstract*—Users eager to publish content on the Web need to either set up a server or use third-party infrastructure. However, the increasing desire to use the Web as a sharing platform for content demands solutions that counter disadvantages of centralized or restricted platforms. Recent efforts are underway at the IETF and W3C to standardize WebRTC for direct browser-to-browser communication. This paper introduces BOPlish, an infrastructure-independent naming and content access architecture for sharing information in User Networks. We demonstrate how BOPlish leverages WebRTC for an easy to use, secure content publishing solution. A custom URI scheme serves as a location-independent addressing mechanism to separate publishing and content retrieval from the underlying infrastructure.**

I. Introduction

The Web Real-Time Communication (WebRTC) specification [1] jointly defined by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) enables Web applications to establish direct connections between two browsers. The protocol and API allow for transferring audio/video data via Secure Real-time Transport Protocol (SRTP) as well as generic binary and textual data via Stream Control Transmission Protocol (SCTP) over Datagram Transport Layer Security (DTLS). Major browser vendors such as Mozilla and Google already ship working implementations of the current specification status and a further deployment of WebRTC-enabled browsers from other vendors can be expected shortly. This enables use cases that were not possible in the past.

With the uprise of Web 2.0 technologies over the past ten years, Web platforms have shifted from pure content silos to services for publishing user-generated content. Today, users also see the Web as a platform to share media, documents and exchange individual information among each other. Currently, perceiving user-generated content on the Web follows a centralized, host-based approach. Examples for such central content sharing community platforms are Facebook, Flickr and Youtube.

Information-centric Networking (ICN) describes the idea of moving from a host-centric to a data-centric networking paradigm. It abstracts publishing and accessing content from the underlying infrastructure facilitated by a defined location-independent naming scheme. ICN potentially fosters the decoupling of user-generated publishing from a dedicated distribution system.

In this paper, we introduce a decentralized, name-based publishing architecture called *Browser-based Open Publishing* (BOPlish) that pursues a similar objective. A BOPlish application runs in the Web browser and connects all participating browsers directly via WebRTC Data Channels. These browsers form a virtual content-centric infrastructure where each user can publish and retrieve content. None of the browsers rely on a server or a central service. Similar to ICN, the accessed content is addressed employing a user-centric naming scheme decoupled from the delivering host. WebRTC acts as an enabler for BOPlish. Any device that has a WebRTC-enabled browser installed can join the overlay without requiring additional software. Additionally, BOPlish benefits from the inclusion of the native identity certification mechanism in WebRTC. Our approach uses this mechanism to secure content and authenticate peers. Encryption and a secure transport of arbitrary data is directly provided by WebRTC.

978-1-4799-1270-4/13/$31.00 ©2013 IEEE

In the remainder, we work out the problem along with related work (Section II) and present the BOPlish approach (Section III). We conclude with an outlook in Section IV.

II. Problem Statement and Related work

Currently, publishing content on the Web requires access to infrastructure such as Web servers and name resolution services like DNS. ICN approaches this problem and elevates the meaning of content by assigning explicit identifiers to data rather than referring to the location of content, e.g. by using an HTTP URL. Content can be stored directly in the network which inherently provides functionality to store, cache and search for it. The ICN approach, however, suffers from conceptual shortcomings as Wählisch et al. [2] have pointed out. Besides unresolved security issues, control over the infrastructure is shifted towards end users. This paradigm opens up the control plane and requires modification of routing states at every user-generated publishing act.

BOPlish focuses on distributing user-generated content within interest groups that we call *User Networks*. Contrary to ICN, our approach is not supposed to operate on Internet-scale. It provides an infrastructure-independent name and content access architecture. Web application providers could potentially benefit from such a system by reducing the server's bandwidth consumption and transfer delay. The clients, on the other hand, are not required to rely on a server for sharing content with other users. As the browser is the natural application platform for the Web, we want to leverage its broad deployment and OS independence. WebRTC provides the required transport mechanism by offering the Data Channel protocol that is used to transfer generic data directly between two browsers.

Various approaches to ICN have been proposed that provide efficient user-centric publishing mechanisms [3]. For addressing content, there exist two major techniques: Using either a flat identifier or a naming hierarchy. The Data-oriented Network Architecture (DONA) [3] is an example of an architecture that uses flat identifiers. The DONA approach assures self-certifying names. Hierarchical identifiers on the other hand, that are used e.g. in Content-Centric Networking (CCN) [3] allow for content aggregation on intermediate routers. Additionally, such namespaces allow for wildcard searches.

Research on leveraging native browser technologies for content distribution has already been addressed by Zhang et al. [4] introducing an implementation of a browser-based content delivery network (CDN). The authors have researched on the possibilities of building a CDN service that is based on a centralized P2P network using the Flash plugin provided by Adobe. Their implementation is centered around a coordinator node that holds mappings between peers and the data stored on these peers. Every distributed asset in the P2P network is fetched using JavaScript from one of the participating peers. Taking advantage of this solution requires the modification of the site's HTML code and the setup of the controller.

III. The BOPlish approach

A. *Overall Architecture*

The BOPlish architecture consists of three components that form the virtual infrastructure: A bootstrap server that handles joining peers and WebRTC signaling, a distributed hash table (DHT) to allow for an infrastructure-independent lookup mechanism and the actual peers that expose a Content API using the BOPlish application in their Web browsers. These components operate as follows:

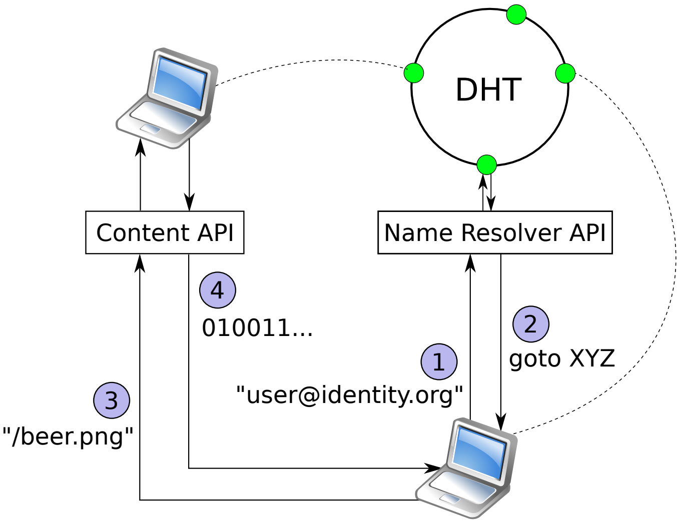

Fig. 1. Nodes in BOPlish retrieve content by issuing a lookup of the content's user ID to the underlying DHT (1) which returns a pointer to the actual node that holds the content (2). This pointer is then used to open a WebRTC Data Channel to the peer, query for the content (3) and transfer it (4).

1) Bootstrap Server: The bootstrap server allows peers to join the system using the WebRTC offer/answer mechanism. Joining a BOPlish group is achieved by simply pointing the users' browser to the URL of the application. The application then contacts the bootstrap server for signaling, requesting an assertion from the identity provider and sending the offer to the signaling server (see Section III-D). This offer is forwarded to an arbitrary node connected to the server so that finally the users' browser is connected to arbitrary other browsers via a Data Channel. This connection is then used to start the DHT joining algorithm.

2) Name Resolver API: The lookup mechanism of the BOPlish architecture is used to find a peer that holds the data the lookup request is directed to. Figure 1 illustrates the procedure. The initiating peer queries a DHT that is distributed among the participating browsers (1). The hash table maps user identities to host references that can be used to initiate a P2P connection (2). The DHT thus acts as an indirection mechanism from location-independent identifiers to actual hosts. By adding redundancy to the DHT this indirection becomes even more stable. We introduced the term *User Networks* that describes a community of peers such as the users of a specific Web site or service sharing an interest of each other's content. Limiting the scope to a User Network simplifies the task of creating a decentralized host lookup mechanism. Using a DHT that only holds references to hosts allows for a small, highly churn-resistant and therefore stable implementation.

3) Content API: The BOPlish architecture denotes a peer as a user of the system. In order for a user to publish content, we opted for a file system like identifier that, combined with the user identifier, is unique in the scope of a User Network. The retrieved host reference from the lookup mechanism is then used to connect to the host and query for the desired content as shown in Figure 1, step 3.

Both, the lookup mechanism and the content retrieval is tightly coupled to a distinct naming schema that is further elaborated on in Section III-B.

B. *Content Naming*

In the BOPlish architecture we have opted for hierarchical names that are syntactically compatible with the Common Multicast API [5] URI scheme:

```
bop://<user>@<identity-provider>/<path>?\
    <sec-credentials>
```

The significant difference to HTTP URLs used on the Web is that the authority part of the URI does not denote a host to connect to. Rather it stands for the asserted identity of the user that owns the content (see Section III-D). The URI thus is location independent.

The user identity part of the URI serves as the key for querying the underlying DHT as denoted in Section III-A2. When a user wants to retrieve content, the BOPlish implementation feeds the URI's authority part into the DHT lookup mechanism which then returns a pointer to the host actually taking hold of the content denoted by the user identity. *path* is the actual contents' path. In the simplest form it just points to a file name but implementations may choose to extend the semantics of the path to e.g. contain wildcards for performing searches. The optional *sec-credentials* query string may contain security credentials that are used to verify the content by using a cryptographic hash. Examples of possible BOPlish URIs look like this:

```
bop://alice@identityA.org/public/*.ogg
bop://bob@identityB.org/secret.doc?\
    hash=urn:sha1:<hash>
```

C. Content Retrieval

The result from the lookup is used to open a WebRTC Data Channel to a peer. Afterwards a query for the *path* is issued against this peer, eventually resulting in retrieving the desired content. If the URI contains a cryptographic hash in the security credentials part, the receiver is able to apply the specified algorithm to the received data and verify its integrity.

The semantics of the *path* part of a BOPlish URI depend on the application. Since the retrieval of a path via the Content API opens a Data Channel between two users, applications can use this channel to establish a custom protocol.

For a file-sharing system, the *path* denotes a file system like structure. When a node is queried for content via the Content API, it performs a local search in its content repository. Implementations may differ in how the local retrieval of data is actually performed. Since Web applications don't have direct access to the file system, a possible solution for publishing a file may be to drag and drop it onto the browser. Then, the application can e.g. store it in the persistent local storage space that is exposed to every Web application. If the *path* contains wildcards, an implementation returns a list of all files matching the path. In a real-time chat application a path may lead to a chat room that users join by querying it via the Content API.

D. Security Aspects

Two important aspects have to be taken care of in BOPlish: Establishing a mechanism to authenticate user identities and being able to sign/verify content. Furthermore, the WebRTC DTLS layer ensures end-to-end encryption; communication to the bootstrap/signaling server is encrypted using HTTPS and encrypted WebSockets (WSS).

BOPlish leverages a WebRTC-native mechanism described in [6] that provides a solution for confidentiality, source authentication and integrity. In order to verify a user identity, a BOPlish peer triggers a verification process using the identity provider provided by the user identifier (see Section III-B). BOPlish leverages this mechanism to verify the identity of a particular node after resolving it through the lookup mechanism. This procedure prevents attacks that manipulate the host reference stored in the DHT. Additional countermeasures to DHT attacks depend on the chosen implementation.

A Web application implementing BOPlish needs to be able to encrypt and decrypt content stored on the peers as well as verify its integrity. There are efforts underway at the W3C to expose certain cryptography functions to Web applications via a JavaScript API [7]. Those functions allow a browser, amongst other things, for the generation of public/private key pairs, signing of data or verifying signatures. However, current browser versions only implement parts of the specification. Thus, an application will have to rely on cryptography functions implemented in JavaScript. Libraries providing such functionality are already available.

IV. Conclusion and Future Work

In this paper we have introduced an architecture for decentralized content-publishing between browsers. It enables users to share content without having to set up a Web server or relying on third-party providers. By decoupling content-centric addressing and an infrastructure-independent lookup mechanism, BOPlish has the potential to improve user experience over the current host-centric Web. Conceptually, applications built on top of BOPlish are able to cooperate across servers so that users of different applications can participate in one User Network, provided they implement compatible protocols. Leveraging the built-in mechanisms of WebRTC makes BOPlish inherently secure. Recent development of browser cryptography APIs add to its security.

Future work will be two-fold: We are currently working on a proof-of-concept implementation of BOPlish [8]. It will serve as the basis for conducting performance and user experience tests. In this way, we expect to evaluate the practical implications of a browser-to-browser sharing architecture. The usage of a DHT as distributed data structure is not crucial to BOPlish's design. Our future work may include evaluations of possible alternatives such as a simple mesh. The implementation approach includes developing a Chord DHT in JavaScript as well as defining and building the precise Content and Name Resolver APIs. The resulting implementation shall be usable as a JavaScript library that application developers harness to provide a BOPlish content sharing facility to their users.

Additionally, we focus on use cases that we anticipate for future enhancements of BOPlish. Such additions include the possibility to offload content from one machine to another which is eased by the content-centric nature of BOPlish addresses. Users could e.g., offload content from their stationary PC to their mobile phone when they leave home or use neighboring nodes to replicate content. Applications building on BOPlish range from simple file sharing to collaborative editing and real-time applications like audio/video or text chats.

References

[1] H. Alvestrand, "Overview: Real Time Protocols for Brower-based Applications," IETF, Internet-Draft – work in progress 06, February 2013.

[2] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp, "Backscatter from the Data Plane – Threats to Stability and Security in Information-Centric Network Infrastructure," *Computer Networks*, 2013, accepted for publication.

[3] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A Survey of Information-Centric Networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, July 2012.

[4] L. Zhang, F. Zhou, A. Mislove, and R. Sundaram, "Maygh: building a CDN from client web browsers," in *Proceedings of the 8th ACM European Conference on Computer Systems*, ser. EuroSys '13. New York, NY, USA: ACM, 2013, pp. 281–294.

[5] M. Waehlisch, T. Schmidt, and S. Venaas, "A Common API for Transparent Hybrid Multicast," IETF, Internet-Draft – work in progress 06, August 2012.

[6] E. Rescorla, "RTCWEB Security Architecture," IETF, Internet-Draft – work in progress 06, January 2013.

[7] D. Dahl and R. Sleevi, "Web Cryptography API," W3C, W3C Working Draft, Jun. 2013.

[8] C. Vogt, M. J. Werner, and T. C. Schmidt, "Leveraging WebRTC for P2P Content Distribution in Web Browsers," in *21st IEEE Intern. Conf. on Network Protocols (ICNP 2013), Demo Session*. Piscataway, NJ, USA: IEEEPress, Oct. 2013, accepted for publication.