# DireWolf Goes Pack Hunting: A Peer-to-Peer Approach for Secure Low Latency Widget Distribution Using WebRTC

István Koren, Jens Bavendiek, and Ralf Klamma

Advanced Community Information Systems (ACIS) Group,
RWTH Aachen University,
Ahornstr. 55, 52056 Aachen, Germany
{lastname}@dbis.rwth-aachen.de
http://dbis.rwth-aachen.de

**Abstract.** Widget-based Web applications are outperforming monolithic Web applications in terms of distribution of the user interface on many devices and many standard browsers. However, latency of the remote inter-widget communication may be an obstacle for the uptake of Widget-based Web applications in near real-time domains like Web gaming and augmented reality. In this demo paper we show DireWolf 2.0 which is replacing the XMPP server of the DireWolf approach by a client-side relay realized by the means of WebRTC. This is not only decreasing the latency of the distributed interface for any application but also increasing the security by avoiding man-in-the-middle attacks on the XMPP server. This progress is enabling further uptake in Widget-based solutions in advanced Web engineering.

## 1 Introduction

The Web as a ubiquitous platform allows us to deal with complex tasks within the familiar environment of a Web browser. Mobile devices such as laptops, smartphones and tablets allow us to access Web resources from any possible location. This goes in hand with the shift away from fixed office settings to mobile and dynamic working environments. Yet using multiple devices in parallel is not sufficiently dealt with on the Web. Though the *Responsive Web Design (RWD)* [1] paradigm helps us to open the same Web site on devices with varying screen sizes, it only targets the UI level and does not care about data migration and synchronization issues.

To this end in earlier work we have presented the DireWolf framework [2]. DireWolf is a widget based Web application framework that allows distributing widgets over multiple devices while keeping the application state. It is based on the *Inter-Widget Communication (IWC)* capabilities of the underlying Open Source ROLE SDK[1]. However, the underlying architecture that involves messages being sent over an XMPP server turned out to be an obstacle for latency-sensitive near real-time Web applications.

[1] http://sourceforge.net/projects/role-project/

To solve this problem we hereby introduce the next iteration of the DireWolf framework that uses the recent *Web Real-Time Communication (WebRTC)* draft for sending peer-to-peer messages from browser instances across multiple devices [3]. In the next sections we present our architecture that involves a refurbished message passing and show a preliminary evaluation that confirms an average decrease of message round-trip times of around 80%.

A video of our demo is available at `http://goo.gl/ZV7RJ1`.

## 2 Peer-to-Peer Distributed User Interfaces

DireWolf 1.0 is using a server-side `Device Manager` that maintains different devices and their respective device profiles e.g. tablet or desktop of a user; the server also stores widget arrangements i.e. which widget is present on which device. Clients initiate the migration of widgets by requesting it at the `Device Manager` which then notifies online devices. The communication between clients and the `Device Manager` as well as between the devices themselves is organized by using publish/subscribe over XMPP. The central routing leads to two major problems: First, the indirection over the server is on the expense of high latency. Second, security is threatened as the XMPP server imposes a single point of failure for man-in-the-middle attacks. Since in most cases where devices are used jointly to master a task they are in immediate vicinity, it is obvious to establish direct peer-to-peer connections for message exchange.

The recent WebRTC draft introduces the DataChannels API that allows Web applications to connect and send arbitrary data to instances running on other devices; thereby firewalls as well as NATs and proxy servers are automatically taken care of. Additionally, WebRTC connections are encrypted by default. Therefore, WebRTC fulfills two major requirements to overcome the aforementioned drawbacks of the existing DireWolf framework: Security as well as low latency through avoiding intermediary servers[2].

To leverage WebRTC we have adapted the architecture of DireWolf. Instead of sending the IWC messages through the XMPP server we introduced a *relay* that acts as a proxy server for all participating devices. We deliberately opted for the relay approach in order to avoid a full-mesh topology of the peer-to-peer network as initial tests showed a performance loss caused by resource-constrained mobile devices. The relaying device now hosts the Message Router that is reponsible for setting up the proxied publish/subscribe node and the WebRTC connections to other clients. We still keep the XMPP connection in place for exchanging endpoint information and the initial connection negotiation process; all other messages are sent over the relay. We have successfully submitted an XMPP extension protocol that describes the DataChannel negotiation process over Jingle to the *XMPP Standards Foundation (XSF)* [4].

[2] In case all firewall-traversal techniques fail WebRTC still redirects the encrypted traffic through *Traversal Using Relays around NAT (TRUN)* servers as last resort.

## 3 Evaluation

We performed both a comparative technical evaluation as well as a user study to prove our conceptions. For the technical part, we measured the round-trip-times of messages sent from a widget to an instance running on a remote device. Figure 1 shows the results of the test series which included 50 runs. First we measured the round-trip times on the previous DireWolf framework that uses the XMPP server for message routing; the average delay was around 143 ms.

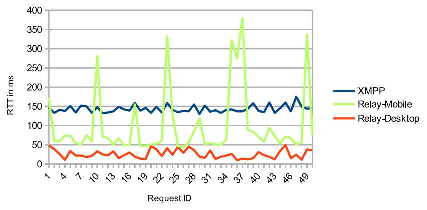

**Fig. 1.** Latency over various connections

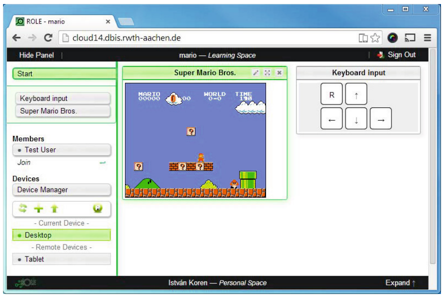

**Fig. 2.** Evaluation Space

In contrast, test runs performed within the newly developed DireWolf 2.0 framework show a **significant reduction** of round-trip times, however results vary depending on where the computationally intensive publish/subscribe relay was hosted. On a state of the art desktop PC we reached an average round-trip time of around 26 ms. Measurements on a recent smartphone revealed an average time of around 99 ms though there were significant outliers; we assume that these peaks were caused by periodical background tasks occupying the processor.

To the usability end we performed a preliminary evaluation with 16 participants recruited from our research lab. We tested the outcome with two highly latency sensitive applications: near real-time collaborative painting as well as gaming. Both widget spaces were shown on an Android tablet and a Windows laptop side by side and executed first on DireWolf 1.0 and then on DireWolf 2.0. First, users were requested to migrate widgets from one device to another. Then, participants were asked to draw a house on the tablet; the painting was synchronized to a laptop screen via the framework. Finally, users were presented a little platform game that can be seen on Figure 2; the control widgets were placed on the touchscreen device while the interactions were shown on the laptop screen.

For the migration 60% rated DireWolf 2.0 as being "very fast" while around 40% noted it was "fast". DireWolf 1.0 scored around 20% for "very fast" and around 55% for "fast"; around 20% of the interviewed persons even considered the migration time being "slow".

For DireWolf 1.0 the overall synchronization speed was rated as ”slow” by around 40% and even ”very slow” by around 25%, while the proportions for DireWolf 2.0 shifted to ”very fast” or ”fast” by almost 100% of the respondents.

## 4 Conclusion and Future Work

In this paper we have presented the demo of the DireWolf framework in its newest iteration which allows distributing widget based Web applications to various devices while leveraging the recent WebRTC draft for peer-to-peer style message exchange. For that we have successfully moved the publish/subscribe functionality from the server to a dedicated client in the form of a relay node; client devices connect to the relay for getting updates on the application state.

The technical evaluation has shown that we could decrease the average latency from around 150 ms to around 25 ms with WebRTC relayed by a desktop computer. Furthermore we are now able to prevent man-in-the-middle attacks on the XMPP server for synchronization messages. A usability study verified our findings.

What remains open for future work is a comparison with other DUI or IWC solutions like in the established *Omelette* project. Technical limitations of our systems include the comparatively high initial migration time, as currently widget resources like HTML, JavaScript and image files on a new device still have to be loaded from a Web server. Employing W3C widgets might solve this problem as they are packaged in a single zip file; the system would then send the whole widget bundle over the peer-to-peer connection. Besides, responsifying the widget spaces has a high priority in order to accommodate for today’s huge variation in display sizes and resolutions.

We are committed to tackle these challenges in future versions of the DireWolf framework and are dedicated to continue defining the underlying standards.

**Acknowledgements.** The work has received funding from the European Commission’s FP7 IP Learning Layers under grant agreement no 318209.

## References

1. Nebeling, M., Norrie, M.C.: Responsive Design and Development: Methods, Technologies and Current Issues. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 510–513. Springer, Heidelberg (2013)
2. Kovachev, D., Renzel, D., Nicolaescu, P., Klamma, R.: DireWolf - Distributing and Migrating User Interfaces for Widget-Based Web Applications. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 99–113. Springer, Heidelberg (2013)
3. Burnett, D., Bergkvist, A., Jennings, C., Narayanan, A.: WebRTC 1.0: Real-time Communication Between Browsers (2013)
4. Bavendiek, J.: XEP-0343: Use of DTLS/SCTP in Jingle ICE-UDP Version 0.1, Experimental (2014)