MicroForwarder.js: an NDN Forwarder Extension for Web Browsers

Wentao Shang
UCLA
wentao@cs.ucla.edu

Jeff Thompson
UCLA
jefft@remap.ucla.edu

Jeff Burke
UCLA
jburke@remap.ucla.edu

ABSTRACT

MicroForwarder.js is a Web browser extension that brings NDN packet forwarding capability to the now-ubiquitous browser platform. It enables local communication and sharing of cached data among the NDN.JS applications running in the same browser. It also allows the NDN.JS applications to share the communication channels to remote NDN forwarders and manages those channels on behalf of the apps. MicroForwarder.js is implemented in pure JavaScript based on the WebExtensions API and can be installed on the latest versions of Firefox and Chrome without requiring custom-built browser platform.

CCS Concepts

•**Information systems** → **Browsers;**

1. INTRODUCTION

Modern Web browsers are powerful and ubiquitous platforms that can host a variety of applications such as email clients, document editors, online games, etc.. Due to its widespread availability and ease of use, the browser platform provides a convenient environment for the users to experience different applications without installing native binaries or compiling the source code.

Our previous work on NDN.JS [4] brings Named-Data Networking (NDN) [5] to the Web browser platform with the objective of empowering users and Web developers with convenient access to the NDN functionalities. Since its publication, NDN.JS has quickly become one of the most popular NDN libraries both inside and outside the NDN community. However, there is a fundamental limitation in the design of the library that affects the usability of the NDN.JS applications: the dependency on the remote NDN forwarders.

Like other NDN applications, the NDN.JS Web apps need to connect to an NDN forwarder (via the WebSocket protocol) in order to fetch and/or publish data over the network. Since the users may not have NDN forwarders installed on their local machines, the NDN.JS apps currently have to rely on remote forwarders (either on the testbed or deployed by the application provider) to provide NDN connectivity. This design has a few major drawbacks. First, when a user opens multiple NDN.JS apps in different browser tabs, each tab will create its own WebSocket connection to the NDN network and consume resources on both the local machine and the remote forwarders. Second, each NDN.JS app has to implement its own logic for automatic discovery and failover of remote forwarders, which creates extra burden for application developers. Third, when multiple NDN.JS apps request for the same data, that piece of data will be retrieved multiple times over the network since there is no sharing of local data cache among the browser tabs.

We have envisioned an NDN-enabled Web browser with built-in packet forwarding capability from the beginning of the NDN.JS project. There has also been an attempt to create an NDN browser by modifying the browser kernel [3]. However, that approach requires users to install a custom-built Web browser, which becomes a major obstacle in its adoption due to different user habits and/or security concerns. Unfortunately, a more lightweight and user-friendly solution was not available due to the limited browser extensibility at that time.

This extended abstract introduces MicroForwarder.js, a recently developed browser extension that provides NDN packet forwarder functionality for Web applications. The extension is implemented in pure JavaScript and leverages the cross-browser WebExtensions API [1] that is currently supported by the latest versions of Firefox and Chrome. Once installed, it allows NDN.JS apps to communicate with the in-browser MicroForwarder through the WebExtensions messaging API and share the connectivity to the external NDN network and the local data cache. The extension itself can connect to remote forwarders via WebSocket and manages those connections internally, thereby freeing the apps from maintaining network connectivity themselves.

In the rest of this extended abstract, we will describe the design and implementation of the MicroForwarder and address future work at the end.

2. DESIGN AND IMPLEMENTATION

MicroForwarder.js comprises three major components: a *background script* running the MicroForwarder logic, a *content script* implementing the communication interface to the MicroForwarder, and a *configuration interface* for managing the runtime state of the MicroForwarder. Figure 1 illustrates the communication model between these components inside a Web browser.

ICN'16 September 26-28, 2016, Kyoto, Japan
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4467-8/16/09.
DOI: http://dx.doi.org/10.1145/2984356.2985233

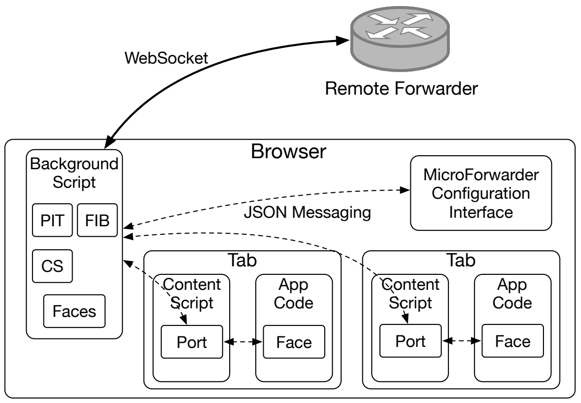
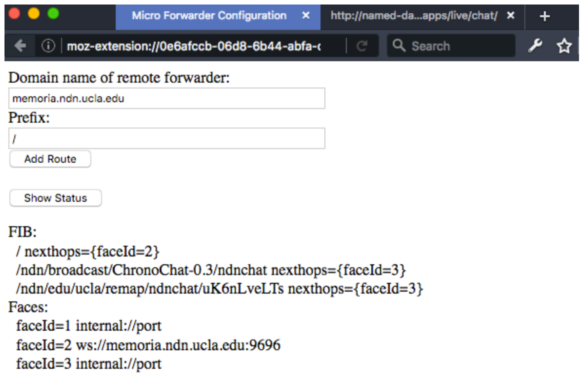


Figure 1: Communication inside a MicroForwarder-enabled Web browser


Micro Forwarder Configuration
http://named-da...apps/live/chat/
moz-extension://0e6afccb-06d8-6b44-abfa-
Search
Add Route
Show Status

Domain name of remote forwarder:
memoria.ndn.ucla.edu
Prefix:
/

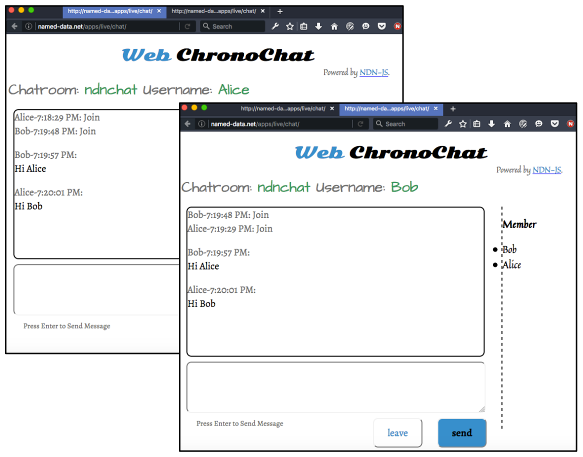
```
FIB:
 / nexthops={faceId=2}
 /ndn/broadcast/ChronoChat-0.3/ndnchat nexthops={faceId=3}
 /ndn/edu/ucla/remap/ndnchat/uK6nLveLTs nexthops={faceId=3}
Faces:
 faceId=1 internal://port
 faceId=2 ws://memoria.ndn.ucla.edu:9696
 faceId=3 internal://port
```

Figure 2: MicroForwarder configuration interface

The background script is started upon browser startup and executes in an isolated environment. It implements the NDN packet forwarding logic, manages the forwarding-related data structures such as PIT, FIB, and CS, and connects to remote forwarder via WebSocket. The content script is automatically loaded in every browser tab but is isolated from the Web app context. It serves as a bridge between the background script and the app, allowing them to communicate by passing JSON objects through the WebExtensions APIs.

The configuration interface is implemented as a built-in Web app that communicates with the MicroForwarder via the same message passing channel and controls its internal state by sending commands encoded as JSON messages. From the configuration page the user can add connections to the remote NDN forwarders and inspect the current forwarder status, as is shown in Figure 2. Note that in order to publish data, the browser needs to register its data prefix with the remote NDN forwarder, which requires a trusted client certificate. To simplify the on-boarding process and allow more users to experiment with the MicroForwarder, our current implementation includes a pre-configured certificate that is automatically trusted by the NDN testbed. One of our future work is to support auto-configuration of remote connectivity using custom certificates.

To facilitate the Web developers in utilizing the MicroForwarder functionality, we have updated the NDN.JS library to provide a new type of `Face` object that connects to


http://named-da...apps/live/chat/
named-data.net/apps/live/chat/
Search
Web ChronoChat
Powered by NDN-JS.
Member
Press Enter to Send Message
leave
send

Chatroom: ndnchat Username: Alice

Alice-7:18:29 PM: Join
Bob-7:19:48 PM: Join

Bob-7:19:57 PM:
Hi Alice

Alice-7:20:01 PM:
Hi Bob

Chatroom: ndnchat Username: Bob

Bob-7:19:48 PM: Join
Alice-7:19:29 PM: Join

Bob-7:19:57 PM:
Hi Alice

Alice-7:20:01 PM:
Hi Bob

- Bob
- Alice

Figure 3: Two ChronoChat apps communicating with each other locally via the MicroForwarder

the MicroForwarder instead of the external forwarders. To demonstrate the usage, we also updated the ChronoChat Web app [2] to use the new MicroForwarder face. We managed to run two ChronoChat apps from two tabs in the same browser and have them exchange chat messages with each other without any remote forwarder connectivity, which is shown in Figure 3. Once the MicroForwarder is connected to one of the NDN forwarders on the testbed, the ChronoChat app can also communicate with remote users over the NDN network.

3. FUTURE WORK

The current implementation of the MicroForwarder.js extension is still preliminary. In the future we plan to extend its functionality along the following directions. First, we will design and implement auto-discovery and auto-configuration of remote NDN forwarders so that the MicroForwarder can automatically connect to the testbed. Second, we will improve the NDN.JS library to auto-detect the MicroForwarder and fall back to remote forwarders if the extension is not available. Finally, we will carry out benchmarks to analyze and improve the performance of the current implementation.

4. REFERENCES

[1] Mozilla. WebExtesions online documentation. https://developer.mozilla.org/en-US/Add-ons/WebExtensions.

[2] NDN Project Team. ChronoChat-js Web App. http://named-data.net/apps/live/chat/.

[3] X. Qiao, G. Nan, Y. Peng, L. Guo, J. Chen, Y. Sun, and J. Chen. NDNBrowser: An extended web browser for named data networking. *Journal of Network and Computer Applications*, 50:134 – 147, 2015.

[4] W. Shang, J. Thompson, M. Cherkaoui, J. Burkey, and L. Zhang. NDN.JS: A JavaScript client library for Named Data Networking. In *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, pages 399–404, April 2013.

[5] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang. Named Data Networking. *SIGCOMM Comput. Commun. Rev.*, 44(3):66–73, July 2014.