Demo: A Versatile Architecture for DTN Services

André Goliath, Jó Ágila Bitsch Link, Klaus Wehrle
Communication and Distributed Systems (Informatik 4), RWTH Aachen University
Aachen, Germany
{goliath|bitsch|wehrle}@comsys.rwth-aachen.de

ABSTRACT

In this demo we present our architecture for delivering application services via Disruption or Delay Tolerant Networks (DTNs). The search for a killer application is still ongoing. Our architecture enables the community to quickly prototype new services and test them in the real world, outside of simulators. Common tasks such as data transportation through the network is handled by the framework but can still be influenced by the service, if required. A robust plugin architecture allows to deploy new applications and routing schemes without affecting stable-running services on the same network, even if the source of the new plugin is untrusted.

1. INTRODUCTION

In the last decade, the concept of Delay Tolerant Networking (DTN) lead to a number of implementation efforts. However, these efforts were mostly focused on routing or implementation of specific services [1, 2]. Besides the reference implementation of the DTN Research Group[1] only little effort was invested into implementations of a more versatile architecture for DTN application services. However, many real-world use cases do not require the complexity of a DTN-specific routing scheme or complex node addressing. Examples of such simple DTN networks include users of cellular-based networks on a train: Direct connectivity to the network will be available every time the train enters a radio cell. If the train is not in radio range neither direct nor multi-hop communication will be possible. Our framework allows to deliver services to such networks as well as to more complex networks requiring sophisticated routing protocols as proposed by the research community.

2. ARCHITECTURE OVERVIEW

Necessarily our architecture has similarities to existing DTN implementations already deployed in the wild. Our framework relies on three core entities, *Transport Agents* (TAs), *DTN Endpoints* (EPs), and *Service Plugins* (SPs), compare Figure 1. All entities are implemented in Java without the use of any platform-specific extensions. This allows to compile each component into a single self-contained `.jar` file that can be executed on a variety of platforms, including Android and other Linux/Unix derivatives, Mac OS, and Windows.

2.1 DTN Endpoints and Service Plugins

An endpoint represents the borderline of a DTN network. At each EP, service plugins are installed for every service provided to the user. Typically we expect at least two EPs in a network, one as part of the user client and one at the remote end. The remote endpoint usually acts as gateway node between the DTN and a second network such as the Internet. EPs are responsible for managing interaction between the user, the SPs installed, and TAs.

A service plugin implements all logic necessary for a single service. This allows to add new services to the network without modifying the EP core code. This also allows to incrementally deploy a specific service to only part of the network. Furthermore, testing new services can be done "live" with only a small number of involved EPs, without interrupting stable running services on the same network. SPs installed on the same EP may communicate with each other. This enables DTN-specific service behavior such as data prefetching and reduces the need for user roundtrips. As an example, consider an e-mail plugin that parses the mail body and if certain links such as youtube videos are found, it would ask the youtube plugin to fetch the video and forward it as an attachment along with the mail.

EPs enforce security policies based on Javas `SecurityManager` API. This allows the administrator to restrict SPs from performing certain tasks such as accessing the file system or opening ports on the local machine. Strict default limitations enable network administrators to allow automatic deployments of new service plugins even from untrusted third party plugin authors.

An EP provides the interface for users and SPs to communicate with each other, but does not offer any notable user interface by itself.

The EP invokes callbacks on the SP and provides the details of incoming user requests. The SP is then expected to answer with a response consisting of at least a status code and a response body, delivered via HTTP. This allows SP developers to design the user interface specifically to the requirements of the service. For data transportation, the host EP offers an API to the plugin. Among others, the API

[1] http://www.dtnrg.org

ExtremeCom '12, March 10-14, 2012, Zürich, Switzerland.

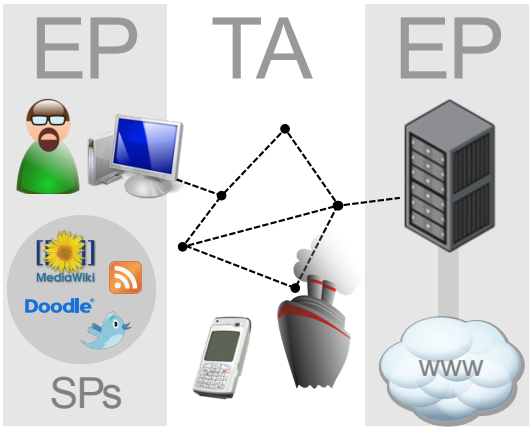

Figure 1: Transport Agents (TAs) forward data bundles between endpoints (EPs). EPs have service plugins (SPs) installed to process bundles and interact with the user.

includes methods to dispatch packets into the network and request status updates of packets. The payload provided by the SP will be wrapped into a DTN envelope. When dispatching packets the SP adds *tags* to the envelope. These tags can include both service-specific information as well as network-interpreted data such as the routing protocol to use.

2.2 Transport Agents

A TA is responsible for forwarding data bundles through the network. A common DTN scenario includes installing TAs on mobile devices such as smartphones or public buses for data forwarding between EPs. The number of TAs a data bundle visits can vary from zero to hundreds, depending on the network structure. TAs take routing decisions based on the tags associated with DTN envelopes. Depending on the routing method used, these tags will also include information such as receiver and sender identification. By default, TAs use a tag-group based epidemic routing approach. Routing plugins allow to add new routing schemes similar to service plugins. A routing plugin for the bundle protocol, e.g., can be added to provide compability to the DTN2 implementation.

The file system representation of an envelope consists of two files, a meta data file in human-readable format and the actual payload file created by the SP. All information necessary for forwarding is contained solely in the meta data file. This allows to propagate the meta data very fast throughout the network while the actual payload can be transported in multiple chunks if necessary.

Incoming envelopes may be processed by SPs even if the payload was not fully received yet. In such cases SPs will receive callbacks whenever a new chunk of payload arrived. This allows the user to benefit from the payload if it is not required in full to be useful, e.g. for audio data that can be played even if the last minutes are still missing.

3. DEMO SETUP

In this demo we present our architecture and two implemented services, Doodle and Wikipedia.

The Doodle web service[2] allows users to setup polls, e.g., to agree on a meeting date and time. The Doodle SP enables a client to fetch and participate in such polls. Besides the actual poll details additional information such as the intended location of the meeting and comments by other users are delivered.

The Wikipedia SP enables users to retrieve and edit pages from the digital encyclopedia Wikipedia[3]. Once the user requests a page via his local EP, the remote EP will fetch the page and embed all necessary resources into a single bundle. Among other data, these bundles include the main article as rendered text. Additionally image thumbnails are Base64-encoded and embedded into the main HTML file. Any related CSS files are loaded and integrated as well. To allow editing the page, the same data bundle also includes the original article source code as wiki text. Once the local EP receives and displays the page it will preprocess links to other articles. Links will be displayed in different colors whether they were previously fetched, are currently being fetched or need to be requested. Clicking on a link pointing to a not-yet-fetched article will result in the dispatch of another article request without interrupting the user's reading experience.

We selected these services as first applications to be implemented for a variety of reasons. Doodle has low requirements on the actual communication, resulting in being a good candidate for a "Hello World" SP. Wikipedia on the other hand is one of the most used web applications today. Implementing a Wikipedia DTN Service allows users to access its information potentially everywhere, even where a conventional Internet connection is too limited to allow browsing and editing the encyclopedia via traditional HTTP transport.

4. CONCLUSIONS AND FUTURE WORK

In this demo we present a novel architecture for service delivery via Delay Tolerant Networks. Our framework allows developers to concentrate their efforts on implementing the service-specific details and eliminate the need for taking care of technical details such as routing decisions. Despite the abstraction introduced by the framework the service developer is still able to influence such core network decisions by the use of tags interpreted by the transport agents.

With the already implemented services, first user tests are conducted. As per the primary goal of this architecture, new services can now easily be implemented. This allows to prototype new services quickly and evaluate them with respect to usability and suitability for challenged networks. Further efforts also include creating service and transport plugins compatible to existing DTN implementations, especially the DTN2 framework and the bundle protocol.

5. REFERENCES

[1] Balasubramanian, A., Zhou, Y., Croft, W. B., Levine, B. N., and Venkataramani, A. Web search from a bus. In *Proceedings of the second ACM workshop on Challenged networks* (New York, NY, USA, 2007), CHANTS '07, ACM, pp. 59–66. http://doi.acm.org/10.1145/1287791.1287803.

[2] Lindgren, A. Social networking in a disconnected network: fbdtn: facebook over dtn. In *Proceedings of the 6th ACM workshop on Challenged networks* (New York, NY, USA, 2011), CHANTS '11, ACM, pp. 69–70. http://doi.acm.org/10.1145/2030652.2030674.

[2] http://www.doodle.com

[3] http://www.wikipedia.org