Haggle: A Data-centric Network Architecture for Mobile Devices

[Extended Abstract] *

Erik Nordström
erik.nordstrom@it.uu.se

Per Gunningberg
per.gunningberg@it.uu.se

Christian Rohner
christian.rohner@it.uu.se

Department of Information Technology
Uppsala University
Sweden

ABSTRACT

Delay-tolerant and opportunistic networks relax the traditional assumption of end-to-end connectivity. Such networks are therefore suitable for content dissemination in sparsely connected regions of the world, and for complementing existing infrastructure by operating cost-free, high bandwidth, albeit high latency, content delivery services.

In this work we argue, however, that content dissemination in the above contexts requires us to revisit delay-tolerant communication at the architectural level, looking at multiple issues such as naming and addressing, congestion control and application interfaces. We propose a new architecture, called Haggle, that leverages the principles of search, as known from desktop operating systems and the Web, in order to achieve truly data-centric communication. Searching is naturally data-centric and embeds principles, such as ranking, that can be used to bind data to interested receivers and to prioritize the data to send during node encounters. We herein give an overview of the Haggle architecture and its basic design.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Store and forward networks

General Terms

Design, Experimentation

1. INTRODUCTION

In this work we describe Haggle – a data-centric network architecture for delay-tolerant and opportunistic content dissemination. Haggle takes a novel approach to dissemination by leveraging principles from Web and desktop search [1]. The design of Haggle is motivated by the lack of content dissemination schemes for delay-tolerant and opportunistic networks that go beyond basic epidemic ones – whether it be network wide epidemics [6], or community wide epidemics [5]. We argue that content driven dissemination requires a revisit of delay tolerant networking at the architectural level, including elements such as naming and addressing, data-centric design, network congestion control, and application programming interfaces.

Our idea of leveraging search principles comes from the observation that searching today is one of the most important ways for users to access digital content – no matter whether the content is stored locally on personal computers, or remotely on the Web. The strength of searching (at least as known from Web search and desktop search) is that it allows users to find content without knowing exactly what to look for. The results returned, usually called *hits*, are typically ranked and this helps with sorting out relevant content without necessarily discarding lower ranked hits. A search engine, like for example Google, may return millions of hits, but it is up to the user to decide which content is relevant through the aid of ranking. The accuracy of this type of ranked search, although certainly not always perfect, is surprisingly good in most cases.

Our goal is to leverage searching as a means to determine how to disseminate content among a set of intermittently connected nodes. A typical scenario that we envisage comprise users carrying mobile phones that hold various items of data. This data the users wish to share with other users having an interest in the data. When users move around and encounter each other, the data they carry is disseminated to interested receivers according to the data's rank against the receivers' expressed interests. The ranking allows us to limit the dissemination to only the top ranked data, and it also allows dissemination in order of relevance to the receiver. The interests of each user are spread in the network as any other data. Therefore nodes can actively *push* data towards interested receivers, potentially over multiple disconnected hops, by exploiting nodes that act as temporary forwarders.

The push-based model of dissemination that we employ is inherent in our design; it is necessary as pull-based dissemination does not work unless there is a fully connected path between the node that wish to pull the content and the node that holds the content. By our definition, a source

*Invited Talk at ACM MobiHoc S^3 Workshop 2009.

node using a pull-based scheme determines from where to pull the data, and is also responsible for managing the actual transmission of the data, for instance, by setting up a path or connection. In contrast, a push-based scheme leaves it up to the current content holders to decide how to best forward the data (i.e., determining the next hops).

With our push-based model, forwarding decisions happen anew at each hop as the data moves closer to the receiver. The decisions are based on the interests collected from other nodes as they are encountered in the network, and therefore the bindings between data and receivers typically become more accurate the further an item of data propagates in the network. The bindings are done by searching, and because this happens anew at each hop, it leads to *late binding* of receivers. Late binding is important in delay-tolerant networks as more information about the network is likely collected the further a binding is deferred in time. Therefore, more accurate bindings are made with late binding rather than early binding by the content publisher only.

A key advantage of push-based dissemination is that it is more suited for out-of-community forwarding of data. A community is, in this context, the set of nodes that are interested in receiving a specific item of data. In intermittently connected networks, pull-based dissemination only works during pair-wise encounters between nodes due to its receiver driven operation (i.e., a node pulls, from a co-located neighbor, the content that matches its interests). Therefore, a content publisher will not disseminate new data unless it actually encounters a node that pulls it. With a push-based scheme, on the other hand, the content publisher can disseminate immediately when the data is first created, assuming it has some information about other nodes that are interested in the data. The dissemination may in this case exploit also out-of-community nodes as relays, which is important if the community is segmented, or if the delay to reach the entire community is unacceptably large.

The contribution of our work is threefold. First, we identify search as a first class networking operation for data-centric communication. We device schemes and algorithms for using search to resolve data-to-node mappings and to order and prioritize the data to disseminate. Second, we have designed a push-based dissemination model that copes with intermittent connectivity, and enables out-of-community forwarding. Third, we have designed and implemented the Haggle architecture that represents an holistic revisit of architectures for delay-tolerant and opportunistic networks in order to better support data-centric communication.

2. HAGGLE OVERVIEW

Haggle essentially provides a push-based data dissemination service with a publish-subscribe API. In this section we do not detail all parts of the architecture framework that enables this service and API, and instead we limit our focus to the search-based dissemination of Haggle.

2.1 Data Addressing

The task of Haggle is to disseminate *data objects* that are tuples { *metadata, data* }. The metadata part exists in all data objects and is defined in XML format. Data objects need not necessarily have any actual data, but when they do the data is typically a file, such as an MP3-file, PDF-document, and so forth. Each locally stored data object on a device is indexed into a searchable data store based on

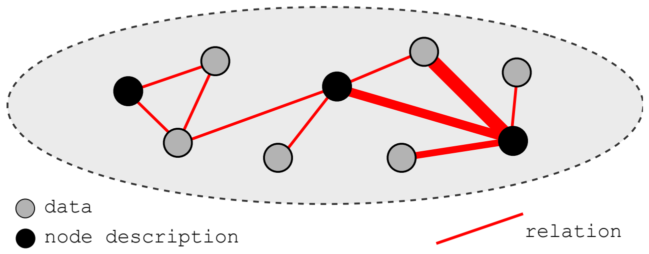

Figure 1: A relation graph holds data objects that represent either application data or other nodes in the network.

attributes (name-value pairs) that are part of the metadata. The attributes in the metadata describe the content of the data object and also form a means to address it. A pair of data objects that share one or more attributes have a relation that can vary in strength. The strength of a relation is determined by a weighting function which is not strictly defined by the architecture, but typically takes into account, e.g., the total number of shared attributes between the data objects in question.

2.2 The Relation Graph

The relations organize data objects in a *relation graph*, where the vertices are data objects and the relations are edges with associated weights. Some data objects are semantically interpreted as other nodes in the network rather than data stored on the local device, and these data objects are therefore called *node descriptions*. A node description has the same format as regular data objects, but the inclusion of special metadata tells Haggle to interpret these data objects as nodes. In this case, the attributes of a node description describe the declared interests of the node it represents instead of the associated content.

The logical structure of the relation graph that exists on all nodes is depicted in Figure 1. In the figure, nodes are black colored vertices while data objects are gray. The figure illustrates how each node can use their relation graph to find relations between the data they carry and the other nodes in the network. This, of course, is dependent on each node having an updated relation graph with recent node descriptions of other nodes. Therefore, a node pair first exchanges their own node descriptions every time they encounter each other, and second they optionally exchange the other node descriptions that they carry. Hence, node descriptions can spread epidemically in the network to ensure that nodes have updated relation graphs.

2.3 Searching

Node descriptions can be seen as persistent search queries that are continuously updated and propagated in the network. When a new node description is received, the receiving node invokes a search operation to resolve the new relations in the graph against the new node description. The new relations define a set of data objects that the node corresponding to the newly received node description – in this case called the *target node* – may be interested in. The weights of the relations determine the level of interest and they make it possible to rank the data objects against the target node. Imagine, for example, that the newly received node description is the right-most one in Figure 1. Any adjacent data objects in the graph represent the result of the search, and the differences in thickness of the relations against the adjacent data objects illustrate the weights and subsequent rankings. When many matching data objects are resolved, the resolver may define a cut-off in how many data objects to return in the search operation (i.e., compare to the top ten ranked hits in Google).

Search operations work analogously in the vice versa case, i.e., when a regular data object is inserted into a graph instead of a node description. In this case, a number of node descriptions are returned as a result of the search operation. The corresponding nodes are the *targets* of the data object in question, and the targets are similarly ranked in order of their interest in the data object.

2.4 Dissemination

Whenever new targets are found, the resolver node tries to push the matching data objects to the targets. This operation may be deferred if the resolving node is currently out of contact with the target receivers, or, the node may optionally invoke a forwarding algorithm to determine whether any currently co-located neighbor is a suitable next hop toward one or more of the targets. Such a next hop node becomes *delegate* for the data object, because the delegate itself have no interest in the data object and only carries it on behalf of someone else.

Any host-centric forwarding algorithm can easily be adapted for use with dissemination in Haggle as the algorithm can separately compute a next hop for each target. Note that forwarding algorithms are only necessary for out-of-community forwarding, i.e., dissemination outside the interest community of a particular data object. If the community is well connected there is no need for a forwarding algorithm, the data can in this case spread through direct encounters between community members.

2.5 Aging of Data

All data objects, regular ones as well as node descriptions, are time stamped and may age when stored in a node's relation graph. An aging algorithm determines when the data object is old enough to be deleted from the graph, and hence from local storage. In general, only node descriptions and delegated data objects age as other data objects are by definition of interest to the node that stores them. When a node looses interest in a data object, the object becomes orphaned and is therefore essentially the same as a delegated data object.

Node descriptions age since some nodes may be encountered infrequently and therefore their interests – and hence node descriptions – quickly become outdated. If node descriptions are kept indefinitely, a node may try to push data objects to nodes that no longer exist in the network, or no longer have the necessary interests.

3. IMPLEMENTATION

We have implemented the Haggle architecture for multiple platforms including Windows Mobile, Linux, Android, Mac OS X and iPhone OS. There is also ongoing work to port Haggle to Symbian. The code is written in C/C++, consists of about 20'000 lines of code (excluding applications) and has matured through a series of releases. The source code will be available to the public in the near future.

4. RELATED WORK

The Haggle architecture can be seen as a publish-subscribe system for delay-tolerant and opportunistic networks. However, while publish-subscribe systems either disseminate based on boolean filters [2], or channels of topics [4], Haggle instead uses searching with ranking for matching content against nodes. The main difference between filtering and searching is that the former does consistent matching whilst the latter does relative matching that may change over time as new content becomes available (i.e., the top n hits may change).

PodNet [5] is an architecture with similar goals as Haggle. PodNet implements content sharing between mobile phones using boolean topic channels and pull-based dissemination. Haggle, however, uses content metadata and searching to determine disseminations instead of topic channels, and also uses push-based rather than pull-based dissemination. We argue that metadata searching is a more flexible approach to binding content to receivers than topic channels, because it also allows ranking that can be used to prioritize the data to disseminate. Further, push-based dissemination better facilitates out-of-community forwarding, as discussed above.

The delay tolerant network architecture (DTN) [3] provides a bundle delivery service with a host-centric addressing scheme based on end-point identifiers (EIDs). Our architecture neither does bundling nor has end-point addressing, as it focuses on data-centric rather than host-centric communication. The DTN architecture is therefore a complementary architecture to ours.

5. SUMMARY AND CONCLUSION

We have given a brief overview of the Haggle architecture which enables data-centric communication over delay-tolerant and opportunistic networks. A key novelty of Haggle is its use of search in the context of networking. This approach we have found to enable flexible and powerful abstractions, resulting in elegant solutions to many network problems; for example by enabling the resolution of target receivers with built in ranking. The ranking system can, for example, be used to prioritize the content to disseminate. Around the search principles we aim to build further functionality into Haggle that enables, among other things, network congestion control and adaptable resource management.

6. REFERENCES

[1] Google desktop. http://desktop.google.com/.

[2] Carzaniga, A., and Wolf, A. L. Forwarding in a content-based network. In *SIGCOMM* (August 2003).

[3] Fall, K. A delay-tolerant network architecture for challenged internets. In *ACM SIGCOMM'03* (August 2003).

[4] Lenders, V., Karlsson, G., and May, M. Wireless ad hoc podcasting. In *IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON)* (June 2007).

[5] May, M., Karlsson, G., Helgason, O., and Lenders, V. A system architecture for delay-tolerant content distribution. In *IEEE Conference on Wireless Rural and Emergency Communications (WreCom)* (October 2007).

[6] Vahdat, A., and Becker, D. Epidemic routing for partially-connected ad hoc networks. Tech. Rep. CS-2000-06, Department of Computer Science, Duke University, July 2000.