PROST: A Programmable Structured Peer-to-peer Overlay Network

Marius Portmann[*], Sebastien Ardon[**], Patrick Senac[***], Aruna Seneviratne[**]

[*]*University of Queensland, Brisbane, Australia*
[**]*University of New South Wales, Sydney, Australia*
[***]*ENSICA, Toulouse, France*
marius@ieee.org, ardon@unsw.edu.au, senac@ensica.fr, a.seneviratne@unsw.edu.au

Abstract

In this paper, we present the idea of a programmable structured P2P architecture. Our proposed system allows the key-based routing infrastructure, which is common to all structured P2P overlays, to be shared by multiple applications. Furthermore, our architecture allows the dynamic and on-demand deployment of new applications and services on top of the shared routing layer.

1. Introduction

A large number of structured peer-to-peer overlay systems such as Chord [1] or Pastry [2] have been proposed recently. Due to their scalability, robustness and self-organizing nature, these systems provide a very promising platform for a range of large-scale, distributed applications, such as distributed file systems, event notification, content distribution, just to name a few.

The main underlying functionality of structured P2P systems is the mapping of data objects to nodes in an overlay network. For this, objects as well as nodes are assigned unique identifiers called *keys*. An object's key is dynamically mapped to a unique live node, called the key's *root*.

Data objects are located by routing lookup messages towards a key's root along the overlay links. Due to the fact that the overlay topology of structured P2P systems conforms to a specific graph structure, message routing can be implemented very efficiently.

Even though all structured P2P systems share the concept of key-based routing (KBR), all currently proposed applications built on structured P2P systems implement their own KBR layer, resulting in a significant duplication of effort. It is also more inefficient to maintain multiple parallel KBR infrastructures, rather than amortizing the cost of single infrastructure among several applications.

Finally, the deployment of new applications and services based on structured P2P systems is currently rather tedious, since it needs to be done manually.

To address these shortcomings, we propose PROST, a programmable structured P2P overlay network. PROST provides a single generic KBR infrastructure that can be shared among multiple applications. PROST is programmable in the sense that new applications and services can be deployed dynamically and on-demand on the PROST platform.

Our architecture builds on work presented in [3], where an API for a generic KBR layer is defined. We go a step further and use this standard KBR layer as a platform for the dynamic deployment of applications and services. Essentially, we are borrowing the idea of programmability from Active Networking and apply it in the context of structured P2P networks.

2. Architecture

PROST is a programmable infrastructure based on the key-based routing layer of a structured P2P network. Applications and services are deployed in PROST via dynamically loading code modules onto nodes of the P2P overlay. These code modules, which we call *Peerlets*, implement the application-specific functionality, while making use of the efficient lookup facilities of the shared KBR layer. Figure 1 illustrates the architecture of a programmable peer node in PROST.

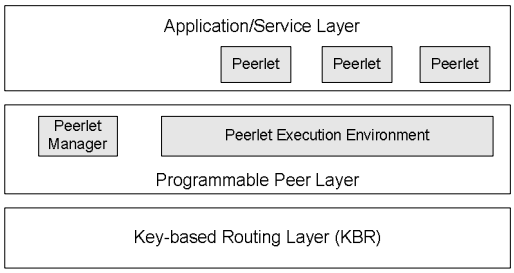

Figure 1. Node Architecture

The key-based routing layer forms the basis of our architecture, providing the main functionality of efficiently mapping object identifiers to live nodes and

locating them in the P2P overlay. The KBR layer also implements mechanisms that deal with the transient and unreliable nature of individual peer nodes by maintaining the routing topology of the P2P overlay in the case of nodes failing, or nodes joining and leaving the network. Reliability of applications based on structured P2P systems is typically achieved by means of replication. Even though replication is the responsibility of the individual applications, the KBR layer provides a number of supporting mechanisms, such as informing the applications of changes in the underlying topology. We refer to [3], for a detailed discussion of the KBR layer.

The Programmable Peer Layer sits on top of the key-based routing layer. It provides the platform for the dynamic deployment of application-specific functionality and services in the form of Peerlets. The Peerlet Execution Environment provides a safe environment in which Peerlets can run, isolating potentially faulty or malicious Peerlets from the host system and other Peerlets. The Peerlet Manager is responsible for the dynamic loading and installing of Peerlets. It also enforces the node's local security and access control policy by controlling and limiting the Peerlets' access to local resources such as CPU, storage and the network.

The Application/Service layer comprises all the application-specific functionality, implemented by Peerlets. The functionality of Peerlets can range from simple service abstractions such as a Distributed Hash Table (DHT) [4], to arbitrarily complex applications.

3. Dynamic Application Deployment

To enable multiple applications to share a common KBR layer in PROST, every message sent over the overlay network contains an *application identifier*, allowing the de-multiplexing of messages at the destination node, and delivering them to the appropriate application, i.e. Peerlet.

To deploy a new application in PROST, the corresponding Peerlet code needs to be made available on a code server, from which it can be downloaded by PROST nodes.

Then, the application, i.e. the corresponding Peerlet needs to be installed manually on at least one node in the overlay. As a simple example, we assume that this new application implements DHT functionality with a simple put(value,key)/get(key) interface. After the initial deployment of the application on one or more nodes, deployment on additional nodes is automatic and is performed on-demand. For example, if the application invokes the put(value,key) method, the KBR layer sends a message containing the application's identifier to the key's root node. Upon arrival, the root node's Programmable Peer Layer forwards the message to the Peerlet identified by the application identifier, which then performs the required application-specific tasks. In the case of a DHT, it simply stores the value for the given key.

If the Peerlet for the given application identifier is not available on the root node, the corresponding Peerlet code is automatically downloaded from the code server and the Peerlet is installed. Applications are therefore deployed incrementally, according to their level of utilization.

To implement certain services, such as multicast forwarding [5], application-specific functionality also needs to be deployed on intermediary nodes in the routing path, and not just the end nodes. PROST supports dynamic deployment of Peerlets for this case, but due to space limitations, we are unable to discuss the details here.

The case where nodes refuse the installation of new Peerlets due to resource constraints or other reasons, is handled similarly to a node failure, and replication is used to address the problem.

However, PROST does not provide absolute service guarantees and applications need to be able to cope with the best-effort service model of the underlying KBR layer.

4. Discussion

Our proposed architecture provides an ideal platform for the rapid implementation and deployment of new applications based on the structured P2P paradigm. We therefore hope that our work can facilitate innovation in this area of research. Our work is in its early stages, and a number of key challenges, such as for example security and resource management, still need to be addressed in detail.

We are currently in the process of implementing a prototype of PROST in Java, which we chose for its support of mobile code and cross-platform capabilities. Our implementation of the KBR layer is based on Chord. However, any structured P2P protocol with which we can implement the generic KBR interface defined in [3] could be used.

References

[1] I. Stoica, et al., "Chord: A Scalable P2P Lookup Protocol for Internet Applications", SIGCOMM 2001, San Diego, CA, August 2001.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," IFIP/ACM Middleware'01, November 2001.

[3] Dabek, F. et al, "Towards a common API for structured P2P overlays", IPTPS'03, Berkeley, CA, February 2003.

[4] F. Dabek, et al., "Wide-area cooperative storage with CFS", SOSP, October 2001

[5] A. Rowstron et al., "Scribe: The design of a large-scale event notification infrastructure", NGC'01, Nov. 2001.