

A Generic Publish/Subscribe Framework for Peer-to-Peer Environment

Shih-Chiang Chien, Yung-Wei Kao, Shyan-Ming Yuan

Department of Computer Science and Engineering
National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu 300, Taiwan
polo.hellfire@gmail.com, ywkao@cs.nctu.edu.tw, smyuan@cis.nctu.edu.tw

Abstract — At present, the structured P2P algorithms have been proposed frequently. Consequently, the P2P application developers need to learn different API semantics. It causes additional efforts of switching to different P2P topologies for P2P applications. Moreover, it is difficult for the developers to evaluate the performance of an application based on a particular underneath P2P APIs. In this research, a novel P2P framework is proposed to assist in developing P2P applications by using various structured P2P protocols and P2P pub/sub algorithms. We construct the structured P2P functional blocks, including network communication components, P2P topology maintenance and routing, network bootstrapping, as well as pluggable pub/sub services in our system. By Comparing to other P2P libraries and platforms, our framework achieves a great success on providing a flexible and extensible development platform.

Keywords-component: P2P;Pub/Sub;

I. INTRODUCTION

Nowadays, with the computing power of PC and network bandwidth increasing, people are willing to dispense their computing power and share information with each others. In pure P2P network, each participant shares their resources in order to gain benefits from other peers. By the natural of sharing in P2P networks, the more users joining the network, the more capacity this P2P network obtained. The scalability is based on the performance of P2P protocols, not determined by the server capacity in traditional centralized architecture. The P2P networks are proved to be an alternative technique in distributed information processing [8]. In addition, the ownership of shared resources and the right of distribution are possessed by the user in the P2P network. On the other hand, the user grants the service provider the rights of using and distributing resources in the typical central server system.

In order to construct an efficient and scalable P2P network, many structured P2P networks have been proposed in recent years and have been verified as efficient and fault-tolerated in large distributed environment. Most of them, e.g. Chord [19], Pastry [17], Viceroy [9], etc., are able to route message between two peers in $O(\log N)$ hops where there are N peers within the network. With the feature of self-organize and failover, structured P2P networks have been widely used in file sharing [7,16], network data storage [4], and distributed indexing [18]. There are several researches work

on deploying distributed personal information portal [12] and online auction systems [6] onto the P2P networks.

Publish/Subscribe paradigm is effective in disseminating information to peers who are interested in such information. In order to apply this mechanism on the P2P network, P2P pub/sub algorithms are designed with the consideration of both time efficiency and transmission overhead. Efficient pub/sub algorithms are able to alleviate the communication burden when dealing with the burst of information on a large scale P2P network.

However, each P2P network was implemented under different approaches, providing various application interfaces. A standardized development and deployment framework is needed to reduce the overheads of implementing P2P protocols and applications. By this framework, developers can focus on the applications' unique functionalities, not the basic network communications.

In the following sections, the issues in P2P pub/sub application development are listed in section two. Section three shows the previous researches of defining common API for P2P programming and pub/sub application. In section four, a layered architecture and primary interfaces are described briefly. Section five shows the pros and cons by comparing with existing solutions. Finally, the conclusion and future work are given in section six and seven.

II. MOTIVATIONS

In the application domain of content management system, e.g., personal blog system, large amount of information is created and requested over the entire user community. With the search capability, users can retrieve information which has particular contents according to the given query. As the P2P community keeps advancing, the number of updating events will soon surpass the size of events that human can handle. By introducing pub/sub mechanism, applications can automatically disseminate information to the interested peers in P2P network. Similar to the RSS supported on many website, the pub/sub paradigm provides the functionality for users to focus on only the interested events. Therefore, pub/sub mechanism is a necessary feature while designing a platform for developing P2P applications.

There are three roles of developing a P2P Pub/Sub-related program: application developers, P2P protocol developers, and P2P pub/sub protocol developers. From the aspect of developing pub/sub applications, programmers usually need to learn new APIs when changing the underlying overlay network. The difference between different P2P networks will reside in peer initialization, network construction, and even communication mechanism. In other words, it is likely to cause code rewriting by implementing the same functionality on different P2P APIs. The same problem exists in changing pub/sub APIs. Traditionally, the applications are strongly coupled with P2P and pub/sub implementations. Application developers have no chance to compare the performance of their systems on different overlays.

- ***Issue 1a: Application developer need to learn different semantics from numerous P2P APIs.***
- ***Issue 1b: The cost of rewriting code is huge for testing performance of particular application on different P2P network.***

While developing a P2P algorithm, developers have to write programs to communicate with other peer. Each P2P API introduces redundant coding style for network programming. Developers have to take additional time on debugging network-related codes. Without network-related code reusing, the effort for adapting to different deployment environment is huge.

- ***Issue 2a: P2P network developers have to write redundant codes for network communication to accommodate different P2P networks.***
- ***Issue 2b: A common process is needed for overlay network initialization.***

Our goal is to solve these issues mentioned above. Thus, a standardized API and communication mechanism for P2P application development needs to be defined.

III. RELATED WORKS

A. P2P Common API

To facilitate independent innovations in P2P protocols, services, and applications, Debak et al. [5] propose a common API for structured overlay networks. There is also related research revises this API with the request-response communication pattern [3]. Moreover, a conceptual model for structured P2P network is proposed by Aberer et al. [1] to provide interoperability between decentralized overlay networks. These researches focus on providing a standardizing P2P network API to application developers.

JXTA [20] is a platform for peer-to-peer computing, proposed by open source community. The JXTA protocols include six protocols which standardize the behaviors between peers. In order to provide interoperability in different language and network environment, JXTA protocol uses XML messages and the super-peer architecture. The index information is also stored within the super-peers, providing reliability and supporting heterogeneous nodes with different set of services installed. JXTA achieves a great success as a P2P application platform, but offers no high level support for structured P2P topology.

B. Pub/Sub Common API

Java Message Service (JMS) [21] is a part of standard service which is included in Java EE platform. JMS defines the common set of interfaces and associated semantics. With the standard API implemented by JMS provider, developers can easily deploy programs with different messaging server. JMS provides two messaging domain:

- **Point-to-Point Domain:** This messaging domain is built on the concept of message queue. Each message has only one consumer. The point-to-point messaging is used when each message will be processed successfully by one consumer.
- **Publish/Subscribe Domain:** This domain is defined with topic-based model. In addition, JMS API defines an SQL-like selection language and provides a built-in facility for supporting application-defined property values.

However, the JMS API is a proprietary specification for Java to intercommunication with messaging server. In order to provide a lightweight API, Pietzuch et al. [15] defined a simplified abstraction for pub/sub system. This common API uses XML-RPC to describe the interaction, and preserve the interoperability with other languages and platforms. With little efforts, this API shows that many pub/sub systems can be brought to compliance. These pub/sub APIs assumed that both publisher and subscriber are clients to a messaging service. Therefore, an auxiliary server is required for delivering messages.

C. P2P Pub/Sub Library

Developing the P2P routing protocols and pub/sub systems is a cumbersome task requiring sophisticated experiments for scalability and reliability. P2P application developers tend to implement their system using a P2P library. FreePastry [14] is an open source P2P library which provides pub/sub functionality. The FreePastry implements the Pastry network routing protocol intended for the deployment on Internet. Based on the Pastry network, additional functionalities are built, such as pub/sub system

and distributed storage. The topic-based pub/sub system supported in FreePastry is Scribe system [2]. Moreover, with the design of peer factory, application can be simulated/tested on local peer without modifying program (other than the initiation codes).

IV. FRAMEWORK DESIGN

Previous researches of common P2P API show the common functionality of structured P2P networks. Inspired by FreePastry and PeerSim [11], we further extend the P2P API by abstracting the network communication from P2P protocols and introduce additional bootstrapping facility. A standard pub/sub API is designed to provide the functionalities of heterogeneous pub/sub model in pure P2P networks.

P2P Applications retrieve a live peer in the P2P network through Bootstrap Service. This live peer is used to initiate the join operation. Application can directly access the P2P Protocol Layer for message routing and performing lookup operation. By registering Pub/Sub Service to local peer, applications use Pub/Sub API to do event publication and subscription. P2P Protocol Layer delegates the physical network transmission to Transport Layer. Environment module loads external parameters from configuration file, and provides global variables to other modules.

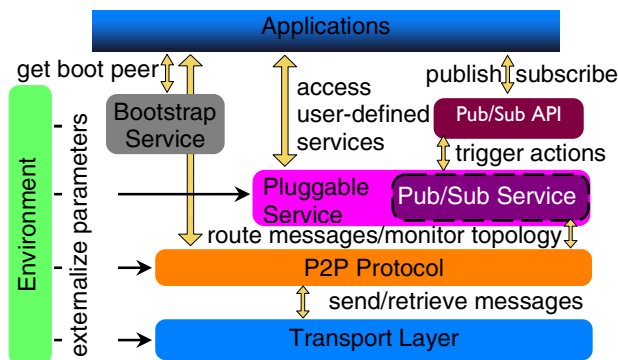


Figure 1. Framework Architecture Overview

A. P2P Protocol Layer

This is the core layer of performing structured P2P functionalities. In P2P Protocol Layer, we propose an object model to describe the relationships within structured P2P network components. This object model consists of the interfaces of common P2P functionalities, peer initialization, and constraints of generating topology. By implementing these interfaces, P2P network library developers are able to create a particular routing protocol.

Peer utilizes the common API for general purpose P2P network accessing. Each Peer associates with a PeerId mapping to identifier space and a CommunicationManager

for network accessing. The identifier space contains Id for the general key to any Resource and subclass for identifying peers. NodeHandle is a peer reference to be used for remote peer communication and topology maintenance.

We use Abstract Factory pattern to standardize the process of id creation and peer initialization. PeerFactory and PeerIdFactory define the interface for creating peer instance and assigning a unique peer identifier. IdFactory consists of the methods generating the key for resources.

The Service interface is defined for creating user-defined application which can monitor the activities of P2P network. In order to achieve the goal of defining pluggable pub/sub service, we introduce the Service interface which can receive several events while a message arrives and topology changes. With the service registration mechanism, developers are free to implement additional functionalities without polluting the code of P2P protocol.

B. Pub/Sub Service and API

This module provides a light-weight API for executing pub/sub related tasks. Publisher and Subscriber define the common pub/sub API which can connect with arbitrary pub/sub service. Each Publisher and Subscriber is associated with one topic.

The PubSubService is a subclass of Service which defines the SPI (Service Provider Interface) needed for implementing P2P pub/sub algorithms. PubSubService handles the actions of pub/sub applications via Publisher and Subscriber. In order to accommodate to both topic-based model and content-based mode, the pub/sub API is designed with topic-based model and additional selector language similar to the one used in JMS for attribute filtering. The content-based model is also supported by introducing a wildcard topic. Pub/sub client program receives interested events by registering EventHandler.

C. Transport Layer

The transport layer encapsulates the detail of resolving physical address and establishing connections. The *CommunicationManager* is the representative of physical network infrastructure. Through the abstraction of network communication, P2P protocol can easily be deployed on different network environments. In our design, peers can register to one instance of CommunicationManager, so that the overhead of activating multiple P2P networks can be reduced. CommunicationManager uses *Address* to establish network connections in order to perform message transmissions. Peers communicate with each other by sending messages. *Message* interface defines the essential attributes for indicating the source peer and the message handler.

D. Bootstrap Service

In order to join an existing overlay network, peers have to aware of at least one live peer which belongs to this network. The bootstrap service provides a general interface which can be adapted to different service implementations.

V. EVALUATIONS & COMPARISONS

The framework we proposed is based upon object-oriented architecture and event-driven methodology. According to the structured P2P specification defined in [3] and [5], we enhance the functionalities into object models. These models fully describe the relationships between the identifier space and the routing protocol. Moreover, based on the framework proposed by Aberer et al. [1], the additional service modules, e.g., P2P Storage Interface, and the P2P Basic Interface, i.e., P2P protocol, are objects directly inherited from the same parent class. However, in our architecture, we introduce pluggable modules, e.g., the Pub/Sub Service, which is independent to the P2P protocol implementation. The features are achieved by invoking services as events arrived. The events include communication messages as well as topology modifications. The event of state transition of handling peers (e.g., peer joins to a network and peer is ready to receive message) is not propagated to the services. Developers can only perform stabilization and replication in proactive style while generating persistence services. Moreover, this pluggable approach makes a lightweight peer implementation. Therefore, the P2P Protocol Layer only needs to handle routing protocols. The additional pluggable services are independent modules not included in the P2P Protocol Layer.

Our design has been focused on pure P2P networks. In other words, each peer in the architecture shares information and collaborates with other peers without a centralized server. In previous researches [15][21], publishers and subscribers are both clients of a message server. In our platform, each peer involves message dispersion and propagation via the pub/sub mechanism without an additional message server. Instead, each peer is involved in the information dispersal of the pub/sub mechanism in our framework, without establishing additional message server. The benefit of pure P2P is that applications do not depend on a pre-constructed server infrastructure. The index information is connoted in the P2P network topology and routing protocol, compared to the super-peer indexing mechanism used in JXTA. However, this statement assumes the computation power of each peer is about equal. According to the assumption, this framework does not grant developer the advantage of deploying P2P applications on the environment of heterogeneous devices. Moreover, this framework does not accommodate to an overlay network containing more than one role of peers, e.g., the super-peers architecture used in JXTA.

In previous researches of P2P protocol, network bootstrapping is usually omitted. By considering the practicality of creating P2P applications, we define the bootstrap service interface and provide two boot server implementations. By externalizing the network communication, the framework allows different protocols transmitting messages through one single network port.

The FreePastry library is an open source implementation of Pastry. With the Scribe system implemented as an additional service, developers can create group communication system with efficient pub/sub capability. In the design of FreePastry, the factory methods are used for testing/simulating applications without modification to the source code. However, the FreePastry library is only can be used for Pastry network and Scribe system. It is functionally limited to develop applications by using FreePastry. Our framework provides a flexible architecture. In this architecture, applications can easily be deployed to any P2P network and any underlying network environment. With the lightweight pub/sub APIs, application developers can adopt any P2P pub/sub service to meet their system requirements.

JXTA is a general P2P platform which allows heterogeneous applications to be deployed on the top of a virtual JXTA network. JXTA can provide additional structured P2P network functionalities based on Peer Resolver Protocol. An open source project named Meteor [10] implements Chord and CAN on the top of the JXTA platform. This approach deploys the DHT overlays upon the virtual JXTA network, which causes the performance downgrades because of the communication overheads among peers introduced by JXTA. The JXTA platform provides a propagating pipe which can simulate pub/sub mechanism via the one-to-many message transmission. The message might be lost during the process of propagation. The performance degrading and reliability issues make this propagation mechanism not scalable to a large group communication system. In our framework, the message transmission among peers is directly delegated to underlying network transportation, which does not incur the overheads of additional node discovery. Without message propagating, our pub/sub service can maintain a distributed multicast structure and support many-to-many message transmission. Therefore, disseminating information among peers will not cause unnecessary bandwidth dissipation.

VI. CONCLUSION

In this paper, we identify several major issues of developing P2P pub/sub applications. These issues result from the lack of standardize P2P API, common P2P pub/sub API, and network abstraction. Therefore, we synthesize standardized P2P API and common pub/sub API into a generic P2P pub/sub framework. Our framework provides a standard P2P API for application developers to interact with various structured P2P networks. Furthermore, a P2P pub/sub API and SPI are introduced for using/creating P2P

pub/sub algorithms in pure P2P networks. In our design, a layered architecture is created with common P2P API, common pub/sub API/SPI, network transportation, and bootstrapping service. This framework allows P2P application developers to switch the underlying overlay with a little bit code to modify. We standardize the control flow between each module. The following benefits are brought out by this framework:

- Easy to develop/deploy applications on different P2P networks and different pub/sub systems.
- Supporting both topic-based and content-based pub/sub models.
- Deploying the P2P applications on various network environments.

This framework is designed for developing P2P pub/sub applications in pure P2P network. It provides an adaptive architecture for developing applications on any overlays without incurring performance degradation. By comparing to other P2P pub/sub library and P2P platform, this framework provides generality of adapting to most P2P routing protocols and P2P pub/sub algorithms and preserves the performance and reliability of P2P networks.

In conclusion, with the common API we proposed, this framework not only standardizes the semantic of using structured P2P network, but also creates a general control flow of develop a P2P pub/sub application. By adopting our framework, developers can generate multiple types of pub/sub applications on the top of every kind of structured P2P networks.

VII. FUTURE WORK

For further extension, we can provide additional common utilities which can help developing applications and P2P protocols. The transport layer can provide predefined retransmission policy for P2P network developer to implement routing protocol in a robust way. These retransmission policies are used to handle low-level transmission exceptions. In addition, providing response-waiting utilities helps P2P protocol developers implementing request-response operations, e.g. lookup operation, without establishing their own lock-notify mechanism. Moreover, this framework can provide connection security and data encryption for developing secured P2P applications. For instance, implementing a `TLPCommunicationManager` offers a secure connection or constructing a `MessageEncryptor` to provide data integrity.

Although this framework is designed in the way of adapting P2P application to different structured P2P network, to implementing these P2P network components completely requires excessive works. A protocol adaptor can

be introduced to reduce the overhead of implementing P2P components using legacy libraries.

This framework can be further extended into a P2P service middleware, integrated with OSGi platform [13]. Our framework can be employed as the communication infrastructure for other OSGi services. With runtime deployment and activation, applications can easily be deployed on an existing P2P topology. Under this service-oriented architecture, P2P components are not only reused in development process, but also in runtime. Moreover, the monitoring services can be dynamically introduced based on the architecture of OSGi platform.

Currently, our framework supports only message-based communication. With application level socket, application usually needs to manage the interaction between peers with stream-based communication. The socket API should have ability for P2P application and service to establish long-live connections between peers. This long-live connection reduces the effort on waiting message acknowledgement in a frequent interaction scenario.

A P2P Blog system is planed to be built using this framework. With implementing real-world P2P pub/sub applications, we can further examine the usability of our framework.

REFERENCES

- [1] K. Aberer, L. O. Alima, A. Ghodsi, S. Girdzijauskas, S. Haridi and M. Hauswirth, "The Essence of P2P: A Reference Architecture for Overlay Networks", In Proceedings of the Fifth IEEE international Conference on Peer-To-Peer Computing, 2005, pp. 11-20.
- [2] M. Castro, P. Druschel, A. -M. Kermarrec and A. I. T. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure", Selected Areas in Communications, IEEE Journal on, vol. 20, 2002, pp. 1489-1499.
- [3] G. Ciaccio, "A Pretty Flexible API for Generic Peer-to-Peer Programming", Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, 26-30 March 2007, pp. 1-8.
- [4] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris and I. Stoica, "Wide-area cooperative storage with CFS", SIGOPS Oper. Syst. Rev., vol. 35, 2001, pp. 202-215.
- [5] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz and I. Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays", Peer-to-Peer Systems II, 2003, pp. 33-44.
- [6] D. Hausheer, "Decentralized auction-based pricing with PeerMart", Integrated Network Management, 2005. IM 2005. 2005 9th IFIP/IEEE International Symposium on, 2005, pp. 381-394.
- [7] Y. Kulbak and D. Bickson, "The emule protocol specification", 2005.
- [8] A. Loo, "The future of peer-to-peer computing", Communications of the ACM, vol. 46, issue 9, 2003, pp. 56-61.
- [9] D. Malkhi, M. Naor and D. Ratajczak, "Viceroy: a scalable and dynamic emulation of the butterfly", In Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing, 2002, pp. 183-192.
- [10] P. Manish, J. Nanyan, S. Cristina and M. Vincent. (2007, Feb. 21). Meteor. 2.4.1, Available: <https://jxta-meteor.dev.java.net/>
- [11] J. Márk, M. Alberto, P. Gian Jesi and V. Spyros, PeerSim: A peer-to-peer simulator, 1.0.3, 2007, Dec. 23. Available: <http://peersim.sourceforge.net/>
- [12] MONKIA Info., "NUWeb", 2007, Available: <http://tw.nuweb.cc/>

- [13] OSGi Alliance, "OSGi Service Platform Core Specification Release 4.1," October. 2007.
- [14] D. Peter, E. Eric, G. Romer, H. Andreas, H. Jeff, C. Y. Hu, I. Sitaram, L. Andrew, M. Alan, N. Animesh, P. Ansley, R. Charlie, S. Dan, S. Jim, S. Atul and Z. RongMei.. FreePastry. 2.0_03, 2007, Nov. 2. Available: <http://freepastry.rice.edu/FreePastry/>
- [15] P. Pietzuch, D. Eysers, S. Kounev and B. Shand, "Towards a common API for Publish/Subscribe", in DEBS '07: Proceedings of the 2007 Inaugural International Conference on Distributed Event-Based Systems, 2007, pp. 152-157.
- [16] J. Pouwelse, P. Garbacki, D. Epema and H. Sips, "The Bittorrent P2P File-Sharing System: Measurements and Analysis", Peer-to-Peer Systems IV, 2005, pp. 205-216.
- [17] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems", in Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, 2001, pp. 329-350.
- [18] I. Stoica, D. Adkins, S. Zhuang, S. Shenker and S. Surana, "Internet indirection infrastructure", in SIGCOMM '02: Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2002, pp. 73-86.
- [19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications", SIGCOMM Comput. Commun. Rev., vol. 31, 2001, pp. 149-160.
- [20] Sun Microsystems Inc., "JXTA v2.0 protocols specification", 2007, Oct 16th. Available: <https://jxta-spec.dev.java.net/>
- [21] Sun Microsystems Inc., "Java message service API", 2003, Dec 2nd. Available: <http://www.jcp.org/en/jsr/detail?id=914>