A Study of Discovery Mechanisms for Peer-to-Peer Applications[1]

M. Kelaskar, V. Matossian, P. Mehra, D. Paul, and M. Parashar
Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ.
kelaskar@paul.rutgers.edu {parashar, vincentm, mehra, dennisp}@ece.rutgers.edu

Abstract

Peer-to-peer applications allow peers to connect or disconnect from a network at any time and are based on a loosely coupled resource distribution model. As a result, robust and efficient discovery mechanisms are central to the efficient functioning of such applications. In this paper we evaluate four discovery mechanisms (flooding and the forward routing algorithms CHORD, Pastry and CAN) against the requirements of three prevalent classes of peer-to-peer applications, and investigate the suitability of these mechanisms for the applications.

Keywords: Peer-to-peer, discovery, content location, CHORD, CAN, Pastry, flooding, forward routing

1. Introduction

Peer-to-peer systems are typically composed of a large number of distributed, heterogeneous, autonomous, and highly dynamic peers. As a result, peer-to-peer applications need sophisticated discovery mechanisms to enable peers to find, identify and communicate with other peers. Centralized discovery mechanisms may not scale, while decentralized discovery mechanisms may not provide the guarantees required by the application. In this paper, we investigate the requirements of three popular classes of peer-to-peer applications. We then evaluate existing discovery mechanisms – flooding and forward routing algorithms as implemented in Chord [1], Pastry [2] and Content-Addressable Network (CAN [3]). Finally, we discuss the suitability of these mechanisms for each application category.

2. Application Categories

We study 3 classes of peer-to-peer applications – distributed file sharing, person-to-person messaging and distributed computing. Distributed file sharing applications (e.g. Freenet, Lime Wire, Morpheus) allow peers to share files with every other peer of a group in an application-based virtual network. Person-to-person messaging systems (e.g. Jabber, Groove, Yahoo! Messenger) allow peers to exchange text as well as whiteboard type of messages. Finally, distributed computing systems (e.g. SETI@home, Entropia), use peers to perform computations. **Table 1** summarizes the discovery requirements for these application categories.

Table 1 Application Categories Requirements

Applications / Requirements	File-Sharing	Messaging	Distributed Computing
(Normalized) Time to Discover	Unbounded	Bounded	Unbounded
Range of Discovery (One/Some/ All) Resources	All/Some	One	All
Accuracy - Correctness (Low / Medium / High)	Low	High	High
Number of Nodes Traversed (Hop Count)	Unbounded	Bounded	Unbounded
Fault Tolerance (Low / Medium / High)	Low	High	Low
Upper Limit on total number of Nodes	Unbounded	Bounded	Bounded

3. Discovery Mechanisms

In this paper, we study four popular decentralized discovery mechanisms based on flooding and request forwarding techniques – Flooding, Chord, Pastry, and CAN. **Table 2** summarizes the relevant parameters of these mechanisms.

Flooding Protocols: In flooding protocols (e.g. Gnutella [4]) peers flood an overlay network with queries to discover a resource. Robustness and extensive reach of discovery characterize flooding protocols. Although the flooding protocol might give optimal results in a network with a small to average number of peers, it does not scale well. Furthermore, accurate discovery of peers is not guaranteed in flooding mechanisms.

Chord: Chord [1] addresses key location and routing in an overlay network forming a ring topology. Keys are assigned to nodes using a consistent hashing algorithm, enabling a node to locate a key in O (logN) hops where N is the total number of nodes in the system.

Pastry: Pastry is a self-organizing, prefix-based routing protocol. Each node in the Pastry network has a unique identifier (nodeId) from a 128-bit circular index space. The Pastry node routes a message to the node with a nodeId that is numerically closest to the key contained in the message, from its routing table of O(logN), where N is the number of active Pastry nodes. The expected number of routing steps is O(logN). Pastry attempts to minimize the distance traveled by the message by taking into account network locality.

Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)
0-7695-1582-7/02 $17.00 © 2002 IEEE

Content Addressable Networks: A Content Addressable Network [3] is a mesh of *N* nodes in a virtual *d*-dimensional dynamically partitioned coordinate space. The CAN discovery mechanism consists of two core operations namely, a local hash-based look-up of a pointer to a resource, and routing the look-up request to the pointer. The CAN algorithm guarantees deterministic discovery of an existing resource in $O(N^{1/d})$ steps. According to its authors, the CAN design is fault-tolerant, robust, and self-organizing.

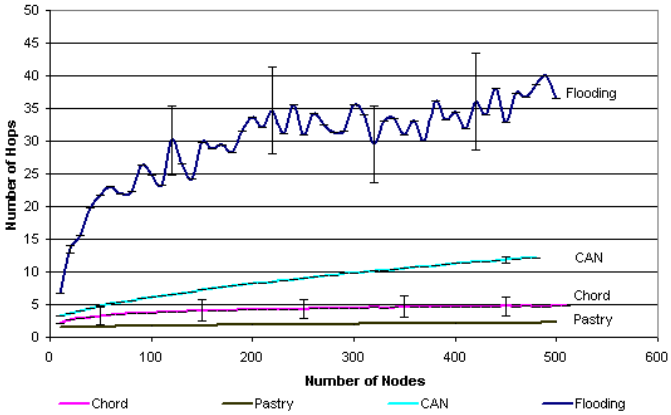
Table 2 Discovery Mechanisms parameters

Forwarding Mechanism / Parameters	FLOODING	CHORD	PASTRY	CAN
Characteristics	Spontaneous Neighbor forwarding	Consistent Hash Lookup Routing	Consistent Hash Lookup Route	Consistent Hash Lookup Route
R/T Table Size	Number of Connected Neighbors	Log N	$(2^b-1)Log_{2^b}(N)$	2^d
Type of Overlay Network	Random/ Closest node	Ring Topology	Self-adjusting proximity network	d-dimensional Cartesian space
Range of Discovery (One/Some/All)	All in [TTL,Fanout] range	One	One	One
Accuracy (Correctness)	Variable	Exact	Exact	Exact
Estimated Hop Count	Variable	$O(Log_2N)$	$O(Log_{2^b}(N))$	$O(n^{1/d})$
Upper Limit on total number of Nodes	Unbound	Dimension of Ring	Size of key	Dimension of space
Fault Tolerance	Yes	Update	Update	Update
Adaptability	No	Yes	Yes	Yes
Load Balancing	No	Natural	Natural	Natural

4. Experimental Evaluation

In order to evaluate the discovery mechanisms, we have developed a general-purpose peer-to-peer simulation framework. In our experiments nodes dynamically joined an overlay-network. The experiment comprised evaluating the number of hops required for discovery for 1000 queries in each network configuration. Each query constituted the lookup of a randomly selected destination from a randomly selected source. In the case of flooding, messages were assigned a Time-To-Live of 7 and a fan-out value of 3. For Chord, each node maintained a finger table pointing to the successor of LogN identifier nodes along the ring. For Pastry each node maintained key identifiers ranging from 0 to 2^{128} –1. The 128-bit key was computed as a cryptographic hash of the IP address of the node. Every node had a routing table of (2^b-1) columns and $[\log_{2^b} N]$ rows, where b is the number of bits required for prefix matching (b = 4). The Leaf set (L) and the Neighborhood set (M) in the node had $2*2^b$ entries each. In the CAN experiment, a two-dimensional overlay network with a single reality was constructed over the generated network topology. The CAN construction algorithm was based on random zone halving in which a new node sends a join request to a random CAN node and acquires half its zone. A query consisted of a randomly selected CAN node looking up a key from the uniformly distributed CAN key space. The results are presented in **Figure 1**. In this plot each point represents the average number of hops taken for a lookup, and the error bars represent the standard deviation of the measurements. As illustrated in the plot, the results are in concurrence with the published asymptotic upper bounds of the routing algorithms underlying Chord, CAN and Pastry. The hash-based request-forwarding algorithms outperform the flooding algorithm by orders of magnitude with an additional cost of routing table maintenance.

Figure 1 Evaluation of Discovery Algorithms



At present, we are actively working towards extending our experiments to evaluate fault-tolerance, accuracy and availability of the Chord, CAN, Pastry and Flooding discovery mechanisms. We will then use these to investigate the applicability of these algorithms to the applications categories.

5. Bibliography

[1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. *Chord: A scalable peer-to-peer lookup service for Internet applications*. Technical Report TR-819, MIT, March 2001

[2]A. Rowstron and P. Druschel, *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), November, 2001.

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker; *A Scalable Content-Addressable Network.* SIGCOMM01, August 27-31, 2001, San Diego, California, USA.

[4] The Gnutella Protocol Specification v0.4 Revision 1.2.

[5] *Peer-to-peer harnessing the power of disruptive technologies*, Andy Oram, O'Reilly, March 2001

[1] The research presented in this paper is supported in part by the National Science Foundation under grant nun [illegible] 9984357 (CAREERS) awarded to Manish Parashar.