PASTRY

Please do not cite this document.
The information contained in this document is available in [5] and [2].

Version 1.0

1 Introduction

In this note we describe the current design of Pastry, a peer-to-peer location and routing substrate upon which Scribe was built. Pastry forms a robust, self-organizing overlay network in the Internet. Any Internet-connected host that runs the Pastry software and has proper credentials can participate in the overlay network.

There have been three primary papers written talking about different aspects of Pastry:

In [5] the original design of Pastry was described. Since this paper was published we have removed the neighborhood set, and simplified the joining algorithm [2]. We have also examined how to make Pastry resilient to failure and attacks [1].

2 Pastry

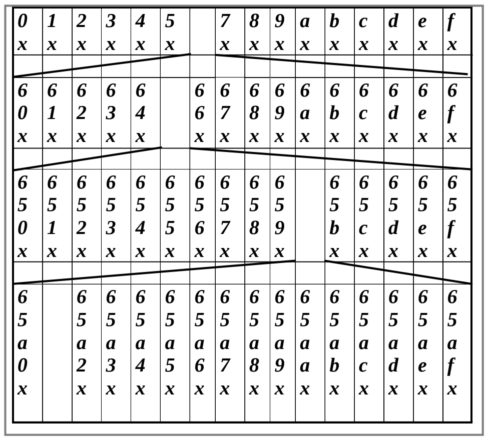
Each Pastry node has a unique, 128-bit nodeId. The set of existing nodeIds is uniformly distributed; this can be achieved, for instance, by basing the nodeId on a secure hash of the node's public key or IP address. Given a message and a key, Pastry reliably routes the message to the Pastry node with the nodeId that is numerically closest to the key, among all live Pastry nodes. Assuming a Pastry network consisting of N nodes, Pastry can route to any node in less than $\lceil log_{2^b} N \rceil$ steps on average (b is a configuration parameter with typical value 4). With concurrent node failures, eventual delivery is guaranteed unless $l/2$ or more nodes with *adjacent* nodeIds fail simultaneously (l is an even integer parameter with typical value 16).

The tables required in each Pastry node have only $(2^b - 1) * \lceil log_{2^b} N \rceil + l$ entries, where each entry maps a nodeId to the associated node's IP address. Moreover, after a node failure or the arrival of a new node, the invariants in all affected routing tables can be restored by exchanging $O(log_{2^b} N)$ messages.

For the purposes of routing, nodeIds and keys are thought of as a sequence of digits with base 2^b. A node's routing table is organized into $\lceil log_{2^b} N \rceil$ rows with $2^b - 1$ entries each. The $2^b - 1$ entries in row n of the routing table each refer to a node whose nodeId matches the present node's nodeId in the first n digits, but whose $n + 1$th digit has one of the $2^b - 1$ possible values other than the $n + 1$th digit in the present node's id. The uniform distribution of nodeIds ensures an even population of the nodeId space; thus, only $\lceil log_{2^b} N \rceil$ levels are populated in the routing table. Each entry in the routing table refers to one of potentially many nodes whose nodeId have the appropriate prefix. Among such nodes, the one closest to the present node (according to a scalar proximity metric, such as the round trip time) is chosen.

In addition to the routing table, each node maintains IP addresses for the nodes in its *leaf set*, i.e., the set of nodes with the $l/2$ numerically closest larger nodeIds, and the $l/2$ nodes with numerically closest smaller nodeIds, relative to the present node's nodeId.

Figure 2 shows the path of an example message. In each routing step, the current node normally forwards the message to a node whose nodeId shares with the key a prefix that is at least one digit (or b bits) longer than the prefix that the key shares with the current nodeId. If no such node is found in the routing table, the message is forwarded to a node whose nodeId shares a prefix with the key as long as the current node, but is numerically closer to the key than the current nodeId. Such a node must exist in the leaf set unless the nodeId of the current node or its immediate neighbour is numerically closest to the key, or



0x	1x	2x	3x	4x	5x		7x	8x	9x	ax	bx	cx	dx	ex	fx
60x	61x	62x	63x	64x		66x	67x	68x	69x	6ax	6bx	6cx	6dx	6ex	6fx
650x	651x	652x	653x	654x	655x	656x	657x	658x	659x		65bx	65cx	65dx	65ex	65fx
65a0x		65a2x	65a3x	65a4x	65a5x	65a6x	65a7x	65a8x	65a9x	65aax	65abx	65acx	65adx	65aex	65afx

Figure 1: Routing table of a Pastry node with nodeId $65a1x$, $b = 4$. Digits are in base 16, x represents an arbitrary suffix. The IP address associated with each entry is not shown.

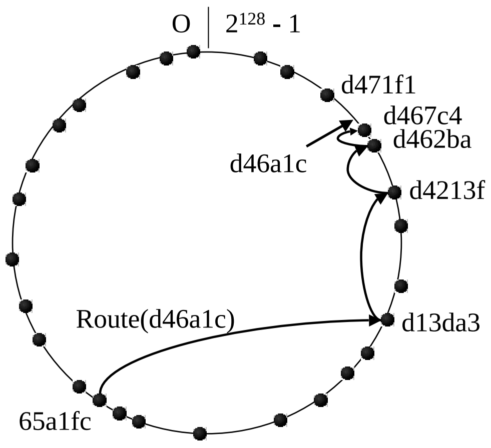


Figure 2: Routing a message from node $65a1fc$ with key $d46a1c$. The dots depict live nodes in Pastry's circular namespace.

$l/2$ adjacent nodes in the leaf set have failed concurrently.

2.1 Locality

Next, we discuss Pastry's locality properties, i.e., the properties of Pastry's routes with respect to the proximity metric. The proximity metric is a scalar value that reflects the "distance" between any pair of nodes, such as the round trip time. It is assumed that a function exists that allows each Pastry node to determine the "distance" between itself and a node with a given IP address.

We limit our discussion to two of Pastry's locality properties that are relevant to Scribe. The *short routes* property concerns the total distance, in terms of the proximity metric, that messages travel along Pastry routes. Recall that each entry in the node routing tables is chosen to refer to the nearest node, according to the proximity metric, with the appropriate nodeId prefix. As a result, in each step a message is routed to the nearest node with a longer prefix match. Simulations performed on several network topology models show that the average distance traveled by a message is between 1.59 and 2.2 times the distance between the source and destination in the underlying Internet [2].

The *route convergence* property is concerned with the distance traveled by two messages sent to the same key before their routes converge. Simulations show that, given our network topology model, the average distance traveled by each of the two messages before their routes converge is approximately equal to the distance between their respective source nodes.

2.2 Node addition and failure

A key design issue in Pastry is how to efficiently and dynamically maintain the node state, i.e., the routing table, leaf set and neighborhood sets, in the presence of node failures, node recoveries, and new node arrivals.

Briefly, an arriving node with the newly chosen nodeId X can initialize its state by contacting a nearby node A (according to the proximity metric) and asking A to route a special message using X as the key. This message is routed to the existing node Z with nodeId numerically closest to X[1]. X then obtains the leaf set from Z, and the ith row of the routing table from the ith node encountered along the route from A to Z. One can show that using this information, X can correctly initialize its state and notify nodes that need to know of its arrival.

[1]In the exceedingly unlikely event that X and Z are equal, the new node must obtain a new nodeId.

To handle node failures, neighboring nodes in the nodeId space (which are aware of each other by virtue of being in each other's leaf set) periodically exchange keep-alive messages. If a node is unresponsive for a period T, it is presumed failed. All members of the failed node's leaf set are then notified and they update their leaf sets. Since the leaf sets of nodes with adjacent nodeIds overlap, this update is trivial. A recovering node contacts the nodes in its last known leaf set, obtains their current leaf sets, updates its own leaf set and then notifies the members of its new leaf set of its presence. Routing table entries that refer to failed nodes are repaired lazily; the details are described in [5, 2].

2.3 Pastry API

In this section, we briefly describe the application programming interface (API) exported by Pastry to applications such as Scribe. The presented API is slightly simplified for clarity. Pastry exports the following operations:

nodeId = pastryInit(Credentials) causes the local node to join an existing Pastry network (or start a new one) and initialize all relevant state; returns the local node's nodeId. The credentials are provided by the application and contain information needed to authenticate the local node and to securely join the Pastry network. A full discussion of Pastry's security model is beyond the scope of this paper.

route(msg,key) causes Pastry to route the given message to the node with nodeId numerically closest to key, among all live Pastry nodes.

send(msg,IP-addr) causes Pastry to send the given message to the node with the specified IP address, if that node is live. The message is received by that node through the deliver method.

Applications layered on top of Pastry must export the following operations:

deliver(msg,key) called by Pastry when a message is received and the local node's nodeId is numerically closest to key among all live nodes, or when a message is received that was transmitted via *send*, using the IP address of the local node.

forward(msg,key,nextId) called by Pastry just before a message is forwarded to the node with nodeId = nextId. The application may change the contents of the message or the value of nextId. Setting the nextId to NULL will terminate the message at the local node.

newLeafs(leafSet) called by Pastry whenever there is a change in the leaf set. This provides the application with an opportunity to adjust application-specific invariants based on the leaf set.

3 Applications

A number of applications have been built using Pastry, including PAST, a persistent, global storage utility [3, 6], Scribe, an application level-multicast system [7, 8], and a cooperative web cache [4].

For more information please visit the Pastry web site: `www.research.microsoft.com\~antr\Pastry`

References

[1] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Security for peer-to-peer routing overlays, 2002. Submitted for publication.

[2] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks, 2002. Submitted for publication.

[3] P. Druschel and A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *Proc. HotOS VIII*, Schloss Elmau, Germany, May 2001.

[4] S. Iyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized peer-to-peer web cache. In *12th ACM Symposium on Principles of Distributed Computing (PODC 2002)*, July 2002.

[5] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. IFIP/ACM Middleware 2001*, Heidelberg, Germany, Nov. 2001.

[6] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proc. ACM SOSP'01*, Banff, Canada, Oct. 2001.

[7] A. Rowstron, A.-M. Kermarrec, P. Druschel, and M. Castro. Scribe: The design of a large-scale event notification infrastructure. In *Proc. NGC'2001*, London, UK, Nov. 2001.

[8] A. Rowstron, A.-M. Kermarrec, P. Druschel, and M. Castro. SCRIBE: A large-scale and decentralized publish-subscribe infrastructure, 2002. Accepted for Journal Selected Areas in Communications. `http://www.research.microsoft.com/~antr/SCRIBE/`.