Routing overlay case of study: PASTRY

Andrea Marin

Università Ca' Foscari di Venezia
Dipartimento di Informatica
Corso di Sistemi Distribuiti

2009

Presentation outline

Presentation based on the original paper: A. Rowstorn and P. Druschel. *PASTRY: Scalable, decentralized object location and routing for large-scale peer-to-peer systems.*

What is PASTRY?

- PASTRY is an implementation of a Distributed Hash Table (DHT) algorithm for P2P routing overlay
- Defined by Rowstron (Microsoft Research) and Druschel (Rice University) in 2001
- Salient features:
  - Fully decentralized
  - Scalable
  - High fault tolerance
- Used as middleware by several applications:
  - *PAST* storage utility
  - *SCRIBE* publish/subscribe system
  - ...

Design of PASTRY: summary

- Any computer connected to the Internet and running PASTRY node software can be a PASTRY node
- Application specific security polices may be applied
- Each node is identified by a unique 128 bit node identifier (NodeId)
  - The node identifier is assumed to be generated randomly
  - Each NodeId in is assumed to have the same probability of being chosen
  - Node with similar NodeId may be geographically far
- Given a *key*, PASTRY can deliver a message to the node with the closest NodeId to *key* within $\lceil \log_{2^b} N \rceil$ steps, where b is a configuration parameter (usually $b = 4$) and N is the number of nodes

Sketch of the routing algorithm

- Assume we want to find the node in the PASTRY network with the NodeId closest to a given key
  - Note that NodeId and key are both 128 bit sequences
- Both NodeId and the key can be thought as sequence of digits with base 2^b

Routing idea

In each routing step, a node normally forwards the message to a node whose NodeId shares with the key a prefix that is at least one digit longer than than the key shares with the present node. If such a node is not known, the message is forwarded to a node that shares the same prefix of the actual node but its NodeId is numerically closer to the key,

State of a node

Each PASTRY node has a state consisting of:

- a routing table
  - used in the first phase of the routing (long distances)
- a neighborhood set M
  - contains the NodeId and IP addresses of the $|M|$ nodes which are closest (according to a metric) to the considered node
- a leaf set L
  - contains the NodeId and IP addresses of the $|L|/2$ nodes whose NodeId are numerically closest smaller than the present Nodeid, and the $|L|/2$ nodes whose NodeId are numerically closest larger than the present NodeId.

The routing table

- The routing table is a $\lceil \log_{2^b}(N) \rceil \times (2^b - 1)$ table
  - b is the configuration parameter
  - N is the number of PASTRY nodes in the network
- The $2^b - 1$ entries at row n each refers to a node whose NodeId shares the present node NodeId in the first n digits but whose $(n+1)$th digit has one of the $2^b - 1$ possible values other than $(n+1)$th digit in the present node id.

Routing table example

Assuming 16 bit NodeId, $b = 2$, number are expressed in base $2^b = 4$.

NodeId 10233102

0 2212102		2 2301203	3 1203203
	1 1 301233	1 2 230203	1 3 021022
10 0 31203	10 1 32102		10 3 23302
102 0 0230	102 1 1302	102 2 2302	
1023 0 322	1023 1 000	1023 2 121	
10233 0 01		10233 2 32	
		102331 2 0	

Unknown NodeId

Routing table dimension

- The choice of b and N determine the routing table size
- The size is approximatively $\lceil \log_{2^b} N \rceil \times (2^b - 1)$
- The maximum number of hops between any pair of nodes is $\lceil \log_{2^b} N \rceil$
- Larger b increases the routing table size but reduces the number of hops
- With 10^6 nodes and $b = 4$ we have around 75 table entries

Neighborhood set

- The Neighborhood set M contains the NodeIds and IP addresses of the $|M|$ nodes that are closest (according to a metric that usually depends on the network topology) to the local node
- This set is not normally used in the routing process
- It is useful in maintaining local properties

Leaf set

The leaf set contain the $|L|$ NodeIds closest to the current node's NodeId

NodeId 10233102

< >

10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

LEAF SET

0 2212102		2 2301203	3 1203203
	1 1 301233	1 2 230203	1 3 021022
10 0 31203	10 1 32102		10 3 23302
102 0 0230	102 1 1302	102 2 2302	
1023 0 322	1023 1 000	1023 2 121	
10233 0 01		10233 2 32	
		102331 2 0	

ROUTING TABLE

Routing algorithm: notation

- D: key to route
- R_ℓ^i the entry in the routing table R at column i with $0 \leq i \leq 2^b$ and row ℓ, $0 \leq \ell \leq \lfloor 128/b \rfloor$
- L_i the i-th closest nodeId in the leaf set L, $-\lfloor |L|/2 \rfloor \leq i \leq \lfloor |L|/2 \rfloor$
- D_ℓ the value of the l's digit in the key D
- $shl(A, B)$: the length of the prefix shared among A and B in digits
- A address of the current node

Routing algorithm

if $\mathbf{L}_{-\lfloor |L|/2 \rfloor} \leq D \leq \mathbf{L}_{+\lfloor |L|/2 \rfloor}$ **then**
    /* Route to a leaf */
    forward to L_i s.th. $|D - L_i|$ is minimal
end
else
    $\ell \leftarrow shl(D, A)$
    if $R_\ell^{D_\ell} \neq null$ **then**
        /* Route to a node in the routing table */
        forward to $R_\ell^{D_\ell}$
    end
    else
        /* Get as close as you can ... */
        forward to $T \in L \cup R \cup M$ s.th. $shl(T, D) \geq l$,
        $|T - D| < |A - D|$
    end

Example: how do we route?

NodeId 10233102

LEAF SET

<			>
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

ROUTING TABLE

0 2212102		2 2301203	3 1203203
	1 1 301233	1 2 230203	1 3 021022
10 0 31203	10 1 32102		10 3 23302
102 0 0230	102 1 1302	102 2 2302	
1023 0 322	1023 1 000	1023 2 121	
10233 0 01		10233 2 32	
		102331 2 0	

- $10233131 \Rightarrow 10233122$ (leaf)
- $10210221 \Rightarrow 10211302$
  - Target not in L because $10233102_4 - 10210221_4 = 22221_4$ and $10233102_4 - 1233000_4 = 102_4$ and $10233232_4 - 10233102_4 = 130_4$
  - $shl(10233102, 10210221) = 3$

Routing performance

Theorem (Expected number of routing steps)

The expected number of routing steps with PASTRY algorithm is $\lceil \log_{2^b} N \rceil$.

Proof

- If the target node is reached using the routing table, each step reduces the set of possible target of 2^b
- If the target node is in L, then we need 1 step
- The third case is more difficult to treat. It is unlikely to happen, experimental results with uniform NodeId, give:
  - If $|L| = 2^b$, probability < 0.02
  - If $|L|2^{b+1}$, probability $= < 0.006$

When case 3 happens it adds an additional step

Reliability

- In the event of many simultaneous node failures the number of routing steps may be at worst linear with N (loose upper bound)
- Message delivery is guaranteed unless $\lfloor |L|/2 \rfloor$ nodes with consecutive NodeIds fails simultaneously. (Very rare event)

PASTRY API (simplified version)

PASTRY exports the following operations:

- **nodeId = pastryInit(Credentials, Application)**
  - Join a PASTRY network or create a new one
  - Credentials: needed to authenticate the new node
  - Application: handle to the application that requires the services
- **route(msg,key)**
  - PASTRY routes message *msg* to the node with NodeId numerically closest to *key*

Application API (simplified version)

An application that uses PASTRY services must export the following operations:

- **deliver(msg,key)**
  - PASTRY calls this method to deliver a message arrived to destination
- **forward(msg,key,nextId)**
  - PASTRY calls this method before forwarding a message. The application may change the message, or *nextId*. Setting *nextId* to null terminates the delivering.
- **newLeafs(leafSet)**
  - Used by PASTRY to inform the application about a change in the leaf set

Scenario and assumptions

- Node X wants to join a PASTRY network
- X's NodeId is computed by the application
    - E.g. may be a SHA-1 of its IP address or its public key
- X knows a close (according to the proximity metric) node A

Join message

- Node X sends to A a message of *join* whose *key* is X's NodeId
- The messages is treated by A like all the other messages
  - A tries to deliver the message to send the message to node Z whose NodeId is closest to *key*, i.e., closest to X's NodeId
- Each node in the path from A to Z sends its state tables to X
- X may require additional information to other nodes
- X builds its own tables
- The interested nodes update their state tables

Neighbourhood set and leaf set

- A is assumed to be close to X so X uses A's neighbourhood set to initialise its own
- Z leaf set is used as base leaf set of X

Building the routing table

- Let $\ell = shl(X, A) \geq 0$
- Rows from 0 to ℓ of A become rows from 0 to ℓ of X
- Row $\ell + 1$ of X is row $\ell + 1$ of B, where B is the node after A in the path to Z
- X sends M, L and the routing table to each node from A to Z. These update their states
- Simultaneous arrivals cause contention solved using timestamp
- Messages sent for a node join are $\mathcal{O}(\log_{2^b} N)$

Dealing with node dapartures

- Node can fail or depart from the network without warnings
- A node is considered failed when its immediate neighbours (in NodeId space) cannot communicate with it:
- In this case the state of the nodes that refer to the failed node must be updated

Repairing the leaf set

Scenario:

- Node X fails
- Node A has X in the leaf set

Actions performed by A to repair its leaf set:

- If $\text{NodeId}_A > \text{NodeId}_X$ then A requires the leaf set of the leaf node with lowest NodeId
- If $\text{NodeId}_A < \text{NodeId}_A$ then A requires the leaf set of the leaf node with highest NodeId
- A uses the received set to repair its own

Repairing the routing table

Scenario:

- Node X fails
- Node A has X as target in the routing table in position R_ℓ^d

Actions performed by A to repair its routing table:

- A asks the entry R_ℓ^d for each target in its routing table R_ℓ^i with $i \neq d$
- If none answers with a live node then it passes to row $R_{\ell+1}$ and repeats the procedure
- If a node exists this procedure finds it with high probability

Repairing the neighbourhood set

- Note that the neighbourhood set is not used in the routing, yet it plays a pivotal role in improving the performance of PASTRY algorithm
- A PASTRY node periodically tests if the nodes in M are live
- When a node does not answer the polling node asks for the neighbourhood set of the other nodes in its M. Then it replaces the failed node with the closest (according to the proximity metric) live one.

Main idea

- PASTRY routing algorithm may result inefficient because few steps in the routing procedure may require long time
- The distribution of NodeIds does not take in account locality
  - Close NodeIds may be geographically far $\Rightarrow$ long delays for message delivering
- The neighbourhood set is used to improve the performance

Assumptions and goal

Assumptions:

- Scalar proximity metric
  - E.g.: number of routing hops, geographic distance
- The proximity space given by the proximity metric is Euclidean
  - Triangulation inequality holds
- If the metric is not Euclidean PASTRY routing keeps working but it may be not optimized

Goal:

- The nodes in the path of a message delivery from A to B are close according to the proximity metric.

Locality in the routing table

Scenario:

- Assume a network satisfies the required property
- We show that when a new node X joins the network the property is maintained
- X knows A that is assumed to be close to X

Idea:

- R_0 of A is used for X. If the property holds for A and A is close to X **then** the property holds for S
- R_1 of X is R_1 of B, i.e., the node reached from A. Why can B be considered close to X? The distance should be weighted on the number of possible targets!
- The same argument applies to the other routing table rows

Further improvements

- The quality of the described approximation may decade due to cascade errors
- PASTRY incorporates a second stage in building the locality route tables
  - Node X joining the networks requires the state from each of the nodes mentioned in the routing table and in the neighbourhood set
  - Node X replaces in its state the nodes in case it received better information
  - E.g. R_ℓ^d of X may be replaced if node addressed by R_ℓ^i has a closest address (according to the proximity metric) that fits in R_ℓ^d.

Locality property

- PASTRY locality features grant that a good route is found but not that the best route is found
- The process approximates the best routing to the destination
- The routing decisions are taken locally!
- Recall that a resource is present in the network with k replicas. But the addressed one could be not the closest (according to the proximity metric)