Viceroy: A Scalable and Dynamic Emulation of Butterfly

Dahlia Malkhi Moni Naor David Ratajczak

Outline

. Introduction

. The Viceroy Network

. Viceroy Construction and Routing

. Terminology and Notation

. Identity and Level Selection, Lookup

. Drawbacks

. Conclusion

Introduction - Viceroy

. Functions as DHT

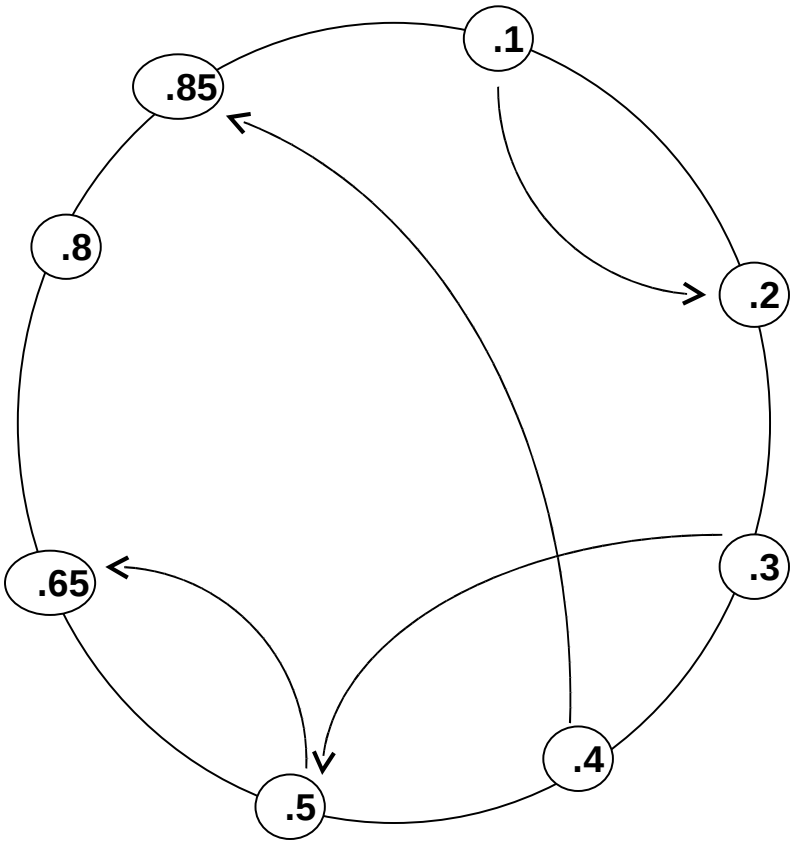
. Constant Degree Routing network with logarithmic diameter

. Randomized Construction

. Distributed and scalable lookup service suitable for deployment in P2P network.

DHT - lookup

- Problem definition:
  - join
  - leave
  - lookup
  - scalable and dynamic
- Goal: Build an overlay routing network, with:
  - Efficient lookup: logarithmic
  - Reasonable join/leave cost: small constant
  - Completely decentralized and dynamic
  - Load balance



Framework

Just like chord, a particular key-value pair resides on the server that is closest to the key.

Viceroy networks uses links between successor and predecessors on the ring for short distances. Moreover they augment the ring construction with a constant number of long range contacts chosen appropriately so that a localized routing strategy produces short path.

These few long range contacts should be chosen randomly with a particular bias towards closer points.

Unlike Kleinbergs work, the resulting topology has only logarithmic dilation.

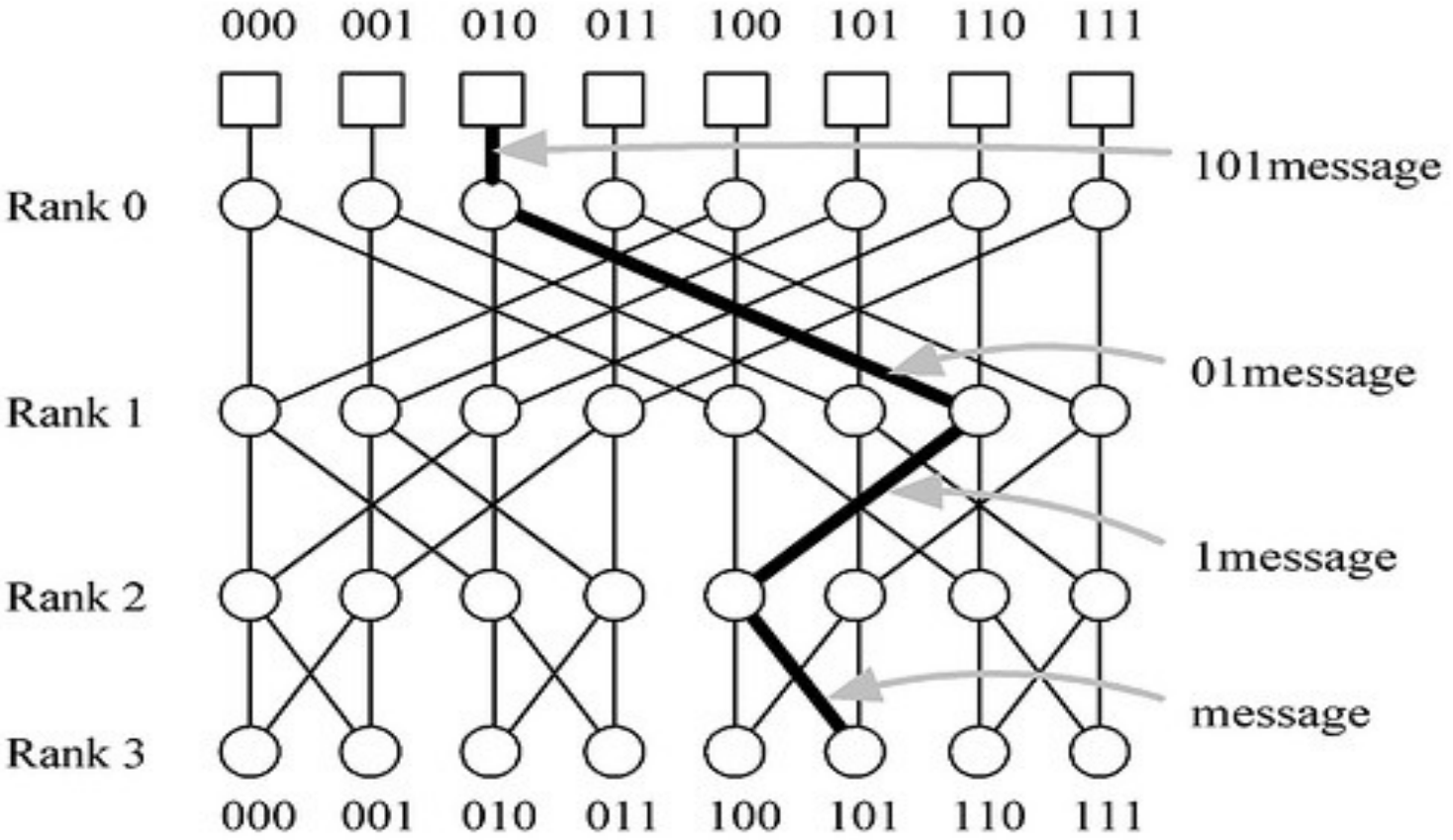
Performance Metrics

Congestion : No server should be bottleneck on the performance of the service. The load should be distributed among participating lookup servers.

Cost of join/leave : When servers join and leave only a small number of servers should change their state.

Lookup Path Length : The forwarding path of a lookup should involve as few machines as possible. Also referred as **Dilation**.

Butterfly network



The Viceroy Topology

. Collection of Butterfly network and the connected ring of predecessor and successor links

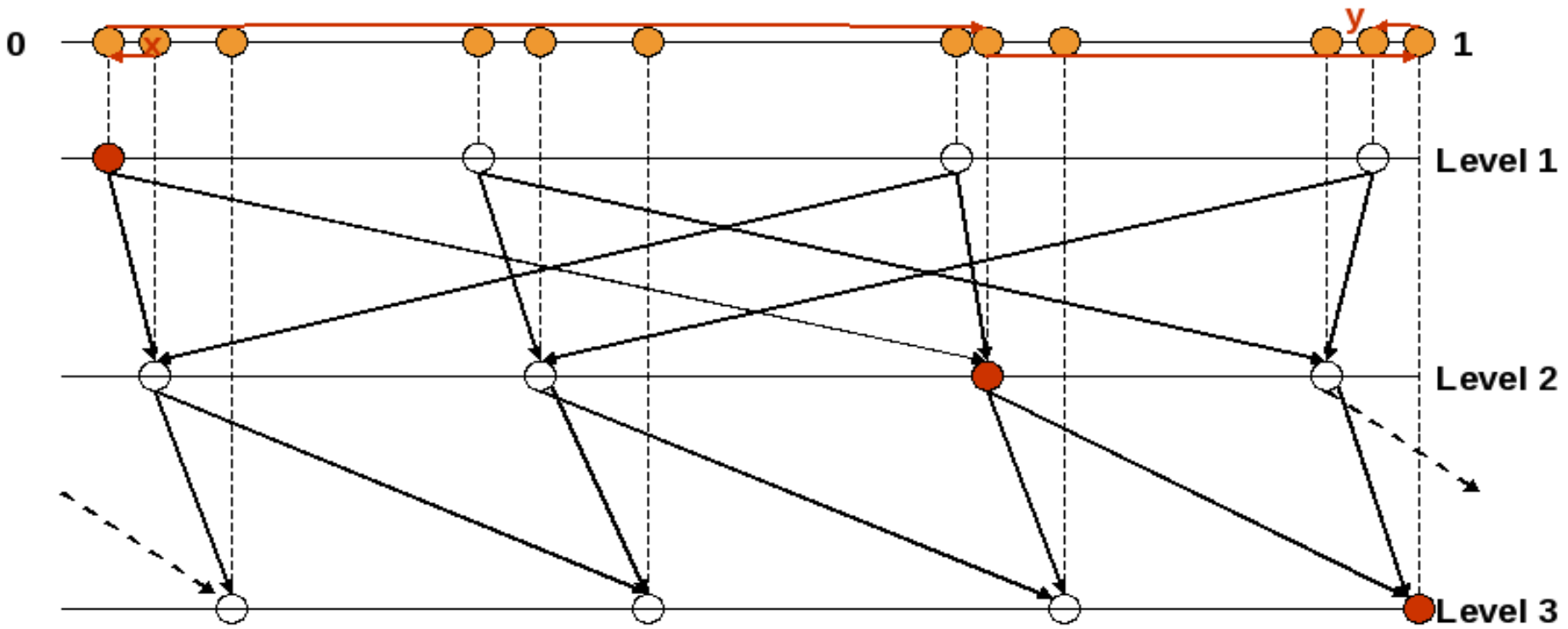
. Each node selects a level at random in such a way that one of (log n) levels is selected with nearly equal probability.

. For a level 'L' node, a “down right” edge is added to a long range contact at level 'L+1' at a distance roughly 1/2^l away, and a “down left” edge at a close distance on the ring level 'L+1'.

. An 'up' edge to a close by node at level 'L-1' is included if L > 1.

. Level ring links are added to the next and previous nodes of same level L

Viceroy topology



The Viceroy Network

. General ring : Where each node is connected to its successor and predecessor.

. Level rings : Where nodes of the same level are connected to each other in a ring.

. Butterfly : For a level 'L' node, a “down right” edge is added to a long range contact at level 'L+1' at a distance roughly 1/2^l away, and a “down left” edge at a close distance on the ring level 'L+1'. An 'up' edge to a close by node at level 'L-1' is included if L > 1.

Routing in Viceroy

Step I : Climb using 'up' connection to a 'L-1' node

Step II : Routing proceeds down the levels of the tree using the down links; moving from L to L+1, one follows either the edge to the close by down link or the far away down link, depending on whether x is at distance greater than 1/2^L or not. This continues until a node is reached with no down links, which presumably is in the vicinity of the target.

Step III : A vicinity search is performed using the ring and level-ring links until the target is reached.

Terminology and Notation

. **Configuration** : Active set of servers. Each server has a real identifier chosen uniformly at random from the range [0,1) before they join.
. **stretch(x,y)** : Clockwise region between x and y non inclusive.
. **Distance d(x,y)** : Clockwise distance from x to y.
. **density q(x,y)** : Number of servers present in stretch(x,y)
. **Succ(x)** : Clockwise neighbor of x.
. **Pred(x)** : Counter clockwise neighbor of x.
. **Nlevel(x)** : Server of level i that is closest in the clockwise direction.
. **Plevel(x)** : Server of level i that is closest in the counter-clockwise direction to x.
. **NextonLevel(x)** : Closest server of the same level as x.
. **PrevonLevel(x)** : Closest server of the same level as x in the counterclockwise direction.

Identity and Level Selection

. Each server picks its identifier independently and uniformly from [0,1) and this id does not change while it is active in the system.

. Select_Level(s):
- Let n' = 1 / d(s, succ(s))
- Select a level L among [1.....[log n']]

Sanity : each node has maximum level between log (n / 2log(n)) .. 3log(n)

Goodness : finding a node on level k takes expected O(log(n)) nodes

O(log^2(n)) w.h.p.

Finding a node on **some** sane level takes O(log(n)) nodes w.h.p.

Join and Leave Operation

. Select an identity s according to identity selection mechanism
. Find using the Lookup method SUCC(s). Update all the corresponding values.
. Transfer from the successor all key value pairs with key between s.predecessor and s.
. Select a level s.level according to the level selection mechanism. Update all corresponding values.
. Find corresponding s.left, s.right and s.up.

When a server leaves it will have to remove all of its outbound connections and notify all of the servers with the inbound connection that they must find a replacement. It must also transfer its resources to its successors.

Lookup

Lookup(x,y):
Initialization: Set cur to y.
Proceed to root: if cur.level = 1 goto traverse-tree phase.
Else
set cur = cur.up if it exists or else, cur=cur.successor and repeat phase.

Traverse tree: if d(cur, x) < 1/2^cur.level then cur = cur.left (go down left) if it exists.
if d(cur, x) >= 1/2^cur.level then cur = cur.right (go down right) if it exists.
If the reqd down link does not exist or if it exists and it overshoots the target, then go to traverse ring phase. Else repeat this phase.

Traverse ring: if cur is the clockwise-closest server to x then compute a response to the lookup. Else, if cur.nextonlevel element of stretch(cur, x) then cur = cur.nextonlevel similarly for prevonlevel else set cur = cur.predecessor, or cur = cur.successor whichever is closer to x and repeat phase.

Drawbacks

. Locality is exploited less effectively because there are a few choices for a neighbor
. The network is vulnerable to being partitioned because each node only has a constant degree.
. The constants in the expected running time are higher.
. Viceroy DHT provides constant expected degree, however its high probability bound is O(log n). For example, it involves estimating the size of network to select the various levels for nodes in system
. The paper doesn't talk about fault tolerance.

Conclusion

. Managing atomic data in distributed hash tables
. Main algorithm: dynamic emulation of butterfly
. Separate fault tolerance issue from routing algorithm

	Node Degree	Dilation	Congestion	Topology
Chord	log(n)	log(n)	log(n)/n	hypercube
Tapestry	log(n)	log(n)	log(n)/n	hypercube
CAN	D	D*(n^1/D)	D*(n^1/D)/D	D-dim torus
Small World	O(1)	Log^2 n	(Log^2 n)/n	Cube connected cycle
Viceroy	7	log(n)	log(n)/n	Butterfly

Questions?

References

. D. Malkhi, M. Naor, D. Ratajczak : Viceroy: a scalable and dynamic emulation of the butterfly. In Proceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC '02)

. http://www.comnet.technion.ac.il/clubnet/presentations/viceroy.ppt