SmartBoa: Constructing p2p Overlay Network in the Heterogeneous Internet Using Irregular Routing Tables[*]

Jingfeng Hu, Ming Li, Weimin Zheng, Dongsheng Wang, Ning Ning, Haitao Dong
Computer Science and Technology Department, Tsinghua University, Beijing, China

{hujinfeng00, lim01, nn02, dht02}@mails.tsinghua.edu.cn, {zwm-dcs, wds}@tsinghua.edu.cn

Abstract

The high heterogeneity of large-scale p2p system leads us to the philosophy that the size of a node's routing table and its updating cost should correspond to the node's capacity. With this philosophy, we design a novel structured overlay: SmartBoa. SmartBoa categorizes nodes into different levels according to their capacities. A node at level *k* has a routing table with $N/2^k$ entries (*N* is the system scale). An efficient non-redundant multicast algorithm is introduced to distribute nodes' changing reports, with which the routing table's updating cost is in proportion to its size. Node can change its level freely to adapt to the fluctuation of bandwidth. At the same cost as the Pastry-like overlay, SmartBoa maintains rather larger routing tables and has much higher routing efficiency. A low-bandwidth (64 kbps) node can maintain 10,000 routing entries at the cost of only 10 percent of its bandwidth. Without the high bandwidth requirement of one-hop overlay, SmartBoa is much more scalable.

1 Introduction

Observations in [5] show that there is great heterogeneity among p2p nodes: bandwidth of the most powerful node is 10^3~10^5 times higher than the weakest one. However, there is no p2p structured overlay fully adapting to this heterogeneity. By far there are two types of structured overlay, i.e. Pastry-like overlay (e.g. Pastry[4], Tapestry[7], Chord[6], CAN[3], SkipNet[2]) and one-hop overlay[1]. The size of Pastry-like overlay's routing table is $O(\log N)$ (Pastry, Tapestry, Chord, SkipNet) or $O(1)$ (CAN). Very limited bandwidth is required to maintain the routing table. Therefore, the extra bandwidth of powerful nodes cannot be utilized. On the other hand, the one-hop overlay's routing table consumes too much bandwidth for updating, which may overburden weak nodes. Its scalability is very poor. It is this reality that motivates us to design a new structured overlay, SmartBoa, in which each node's available bandwidth is adequately utilized. We consider an overlay of this kind the most effective and scalable.

The fundamental idea of SmartBoa is that the size of a node's routing table should be in proportion to the available bandwidth. Nodes with higher bandwidth have larger routing table and faster routing speed. A direct approach to realize it is to control the routing table size of Pastry-like overlay. This can be achieved by letting different nodes have different *b*[1]. Routing table is updated through periodical probing. In this manner, a weak node can handle no more than 2,000 routing

[*] This work is supported by The National High Technology Research and Development Program of China (G2001AA111010), Chinese National Basic Research Priority Program (G1999032702) and National Natural Science Foundation of China (G60273006)

[1] In Pastry, the size of routing table is $\left(2^b-1\right)\cdot\log_{2^b} N$. *N* is the system scale. *b* is a constant which is 4 typically.

entries at the cost of its entire bandwidth. In a p2p system having more than 1,000,000 nodes, this improvement is too trivial to make substantial sense.

A novel idea brought out by one-hop overlay is that a node can maintain a much larger routing table through event reporting than periodical probing. However, even through this effective mechanism, the maintaining cost of one-hop overlay is still too high to be handled by normal node. This greatly constrains its scalability. An applicable idea to improve the scalability is manipulating routing tables into different sizes according to nodes' capacities. The weaker the node is, the lower the maintaining cost is.

To manipulate routing tables of one-hop overlay into different sizes, how to design the report multicast algorithm is the key issue. It encounters following challenges: 1) When a node joins or departures, it is difficult to determine the multicast scope of the changing report (we call this scope the report's "*target group*"). 2) The report should be received by nodes in its target group once and only once. That is to say, the report algorithm should be non-redundant. Otherwise, the bandwidth used to update the routing table will not be in proportion to its size. 3) There are weak nodes that do not know all the other nodes in the target group. They should not be required to send reports to nodes outside their routing tables.

SmartBoa develops a novel multicast algorithm, which solves above problems successfully. Nodes in SmartBoa are categorized into different levels according to their capacities. Size and content of a node's routing table are related to its ID and level. Reports flow from powerful nodes to weak nodes. When node M's state change, whether node N should know about it is determined by M's ID, N's ID and N's level. No additional information is required to determine the target group. In the target group, the upper level nodes' routing tables contain those of the lower level nodes. It means that the upper level nodes know, and can control where the lower level nodes send the report. Therefore, if a report flows strictly from upper level nodes to lower level nodes, the multicast can be confined in the target group without redundancy.

Under current network environments, in SmartBoa even the weakest node (with a bandwidth of 64 kbps) can maintain thousands of routing entries at the cost of 10 percent of its bandwidth. Routing efficiency in SmartBoa is very high. On the other hand, when the system's scale or changing frequency increases to a degree that a node cannot handle, the node can freely debase its level to decrease the maintenance cost. This gives SmartBoa remarkable scalability. The long joining period, which is typical in one-hop overlay, can be avoided through *warm up*, a process in which the node's level rises gradually when joining.

2 Core Designs

2.1 Routing entries

In SmartBoa, nodes in different levels have routing tables of different size and having different number of routing entries (Entries in routing table are pointers to remote nodes: ID + IP address + port).

As in Pastry and Chord, each node in SmartBoa is assigned a 128-bit node identifier, which indicates the node's position in a circular key space. ID is generated randomly (for example, by SHA-1 hashing function) and supposed to scatter evenly in the ID ring. A message with a 128-bit key is sent to nodes whose ID is closest to the key in the ID ring. We call this node the key's *holder*.

SmartBoa categorizes nodes into different levels (from level 0 to no more than level 127) according to their bandwidth. A node in level *k* maintains routing entries whose ID's *k*-bit length suffix is the same to local node. For example, a node in level 2 with ID of 101~1$\boxed{10}$ maintains routing entries whose ID is xxx~x10. A node in level *k* maintains $N/2^k$ routing entries if all the nodes' IDs scatter evenly in the ID ring (*N* is the system scale). Nodes in level 0 maintain routing entries covering all the other nodes in the system, just like nodes in one-hop overlay. The lower the node's level is, the fewer routing entries it has to maintain.

We define the *k*-bit length suffix of a node M[2] in level *k* as the node's "*label*", denoted by α_M. Labels of nodes in level 0 are empty, which is denoted by *Φ*. All nodes with the same label α form a set, which is denoted by $\{\alpha\}$.

2.2 Multicast algorithm。

Before discussing the multicast algorithm, we first make the following definitions:

[Definition 1] If node A's label α_A is a suffix of node B's label α_B, and $\alpha_A \neq \alpha_B$, then say A is *superior to B*, or A is B's *super node*, denoted as $A > B$. Obviously, B's routing table is a subset of A's.

[Definition 2] If node A has no super node in the system, *A* is called a *top node*.

[Definition 3] If $A > B$ or $\alpha_A = \alpha_B$, and *A* is a top node, then say A is *B's top node*.

In normal conditions, the set of top nodes is $\{\Phi\}$. But if the scale of system is gigantic, there may be no node powerful enough to stay at level 0. At that time, the set of top nodes may split into $\{"0"\}$ and $\{"1"\}$ or sets in even lower level.

In SmartBoa, we call the set of nodes whose routing table contains pointer to node *M* as *M*'s "*target group*", which is the union of $\{\Phi\}$, $\{"M_1"\}$, $\{"M_2M_1"\}$, $\cdots$, $\{"M_{128}M_{127}\cdots M_2M_1"\}$ (M_i denotes the *i*-th to last bit of M). Obviously, in the target group the upper-level nodes have routing tables entirely containing those of the lower-level nodes.

```
rcv_bcast(ID m, Step s):
        //Receive event notification related to node m at the step s.
    Rs = getTargetGroup(routing_entries, m)
        //Get target group of M from local routing entries
    For i := s+1 to 128 do
        Rn := getSuffix(Rs, i−1)
            //Get set of nodes in Rs whose ID's (i-1)bit length suffix is the same as local ID's, but the i-th to last bit is different.
        If Rn = null then
            continue
        fi
        P := getHighestLevel(Rn)
            //Select one of the highest level nodes
        send_bcast(P, m, i)
            //Send the message to P, mark it as the i-th step.
    End do
```

Figure 1: Pseudo-code for SmartBoa's multicast algorithm. Describing what a node to do when it receives a message about node m's changing event.

The pseudo code of the multicast algorithm is in Figure 1. The basic principle is that at step *k*, the node receiving the report forwards it to another node whose ID's last *k* bits are identical to local node but the (*k*+1)th to last bit is different. Figure 2 illustrates how a report flows among the target group. Figure 3 shows those nodes that have received the report after each step. From Figure 2 and Figure 3 we can see that reports flow from powerful nodes to weak nodes, and every node in the target group receives the report once and only once.

[2] In order to simplify discussion, we name a node with nodeId M as node M.

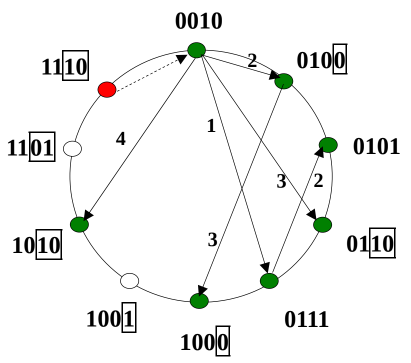
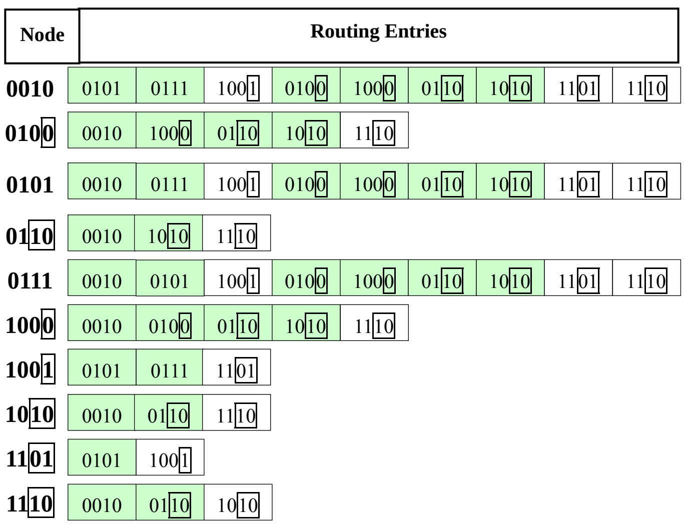


Figure 2.1: An example of multicast process. Red point is the changing node. Green points are nodes in the target group of the changing report. Arrows show the flow of report. Numbers besides the arrows are step numbers. Framed bits are labels.



Node	Routing Entries								
0010	🟩 0101	🟩 0111	100$\boxed{1}$	🟩 010$\boxed{0}$	🟩 100$\boxed{0}$	🟩 01$\boxed{10}$	🟩 10$\boxed{10}$	11$\boxed{01}$	11$\boxed{10}$
010$\boxed{0}$	🟩 0010	🟩 100$\boxed{0}$	🟩 01$\boxed{10}$	🟩 10$\boxed{10}$	11$\boxed{10}$				
0101	🟩 0010	🟩 0111	100$\boxed{1}$	🟩 010$\boxed{0}$	🟩 100$\boxed{0}$	🟩 01$\boxed{10}$	🟩 10$\boxed{10}$	11$\boxed{01}$	11$\boxed{10}$
01$\boxed{10}$	🟩 0010	🟩 10$\boxed{10}$	11$\boxed{10}$						
0111	🟩 0010	🟩 0101	100$\boxed{1}$	🟩 010$\boxed{0}$	🟩 100$\boxed{0}$	🟩 01$\boxed{10}$	🟩 10$\boxed{10}$	11$\boxed{01}$	11$\boxed{10}$
100$\boxed{0}$	🟩 0010	🟩 010$\boxed{0}$	🟩 01$\boxed{10}$	🟩 10$\boxed{10}$	11$\boxed{10}$				
100$\boxed{1}$	🟩 0101	🟩 0111	11$\boxed{01}$						
10$\boxed{10}$	🟩 0010	🟩 01$\boxed{10}$	11$\boxed{10}$						
11$\boxed{01}$	🟩 0101	100$\boxed{1}$							
11$\boxed{10}$	🟩 0010	🟩 01$\boxed{10}$	10$\boxed{10}$						

Figure 2.2: Routing entries of nodes in Figure 2.1. The nodeIds with green background are in the target group of node 1110.

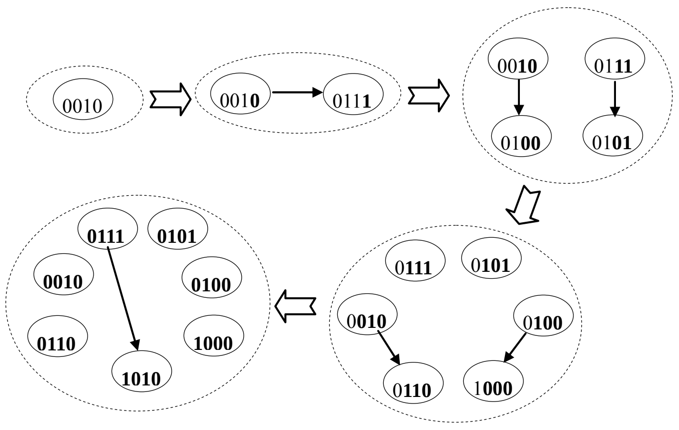


Figure 3: Nodes having received report in step 0 to step 4. Bold bits of nodeIds show that in step k the k-bit length suffix of any the nodes having received report is different from that of any others.

Considering the limited space, formal proving of the completeness and non-redundancy of the multicast algorithm will not be presented in this paper.

In order to broadcast the reports, every node in SmartBoa maintains *top entries* pointing to some top nodes. Top entries are maintained by lazy update because powerful nodes are stable.

The new joining node or a node changing its level sends the report to a top node by itself. But a node may depart or break down silently without warning. To detect nodes' silent departure, node M probes its right neighbor node N in set $\{\alpha_M\}$ periodically. If M finds N's departure, it sends report to one of its top nodes (obviously, M and N have the same top nodes).

With the multicast algorithm, a weak node in SmartBoa can maintain a quite large routing table. For example, a modem-linked node whose bandwidth is only 64 kbps can maintain almost 15,000 routing entries. The result is drawn from following calculations. Assuming 10 percent of the node's bandwidth, i.e. 6.4 kbps, is used for updating its routing table. The size of a report is no more than 500 bits. Thus, the node can receive 12 reports per second. Assuming a node's average online period is one hour[5]. In one of its life circle, a node may cause 3 reports (joining, departure and level changing in warm up). Suppose the size of routing table is r. r nodes will trigger $3r/3600 = r/1200$ reports per second. A node can receive 12 reports per second, $r/1200 = 12$, $r = 14{,}400$. When the system scale is about 14,000, even the weak node can maintain routing table containing all the nodes in the system, and the routing can be done in one hop.

2.2 Routing

SmartBoa adopts greedy routing algorithm. A message with key M is sent to the closest (in ID space) node in local routing table in every routing step. Unlike the broadcast algorithm, the routing algorithm has no bias to powerful nodes, in order to not over burden them. From the following discussions we can see that even without bias to powerful nodes, the routing efficiency is good enough.

Because not all the nodes are powerful enough to maintain one-hop routing tables, SmartBoa introduces *leaf set* to ensure the convergence of routing. Leaf set records *l* (*l* is 16 or 32 normally) nearest nodes in ID space on each side of local node. SmartBoa maintains leaf set through heartbeat messages, just as Pastry does. In a system with $r \times l/2 = 14{,}400 \times 16 \approx 230{,}000$ nodes, generally a message can be route by weak node to its destination in two hops (first hop via routing entries, and second hop via leaf set).

When system scale is gigantic, in weak nodes' routing tables there are too many nodes between nearby entries to be covered by leaf set. Then after the first hop via routing entries, a message has to make several hops via leaf set. To accelerate routing, SmartBoa introduces *finger entries*. Finger entries are bisearch pointers between local node *M* and its right (left) neighbor node *N* in routing entries. They are pointers to node $(M+N)/2$, node $(M+(M+N)/2)/2$, … (until overlapping with leaf set). Finger entries are maintained by heartbeat messages.

In a system having 1,000,000 weak nodes, the routing efficiency of SmartBoa is much better than Pastry. In SmartBoa the first hop of a message is via routing entries. If IDs scatter evenly, after the first hop there are at most $N/2r$ nodes(averagely $N/4r$ nodes) between the message's current position and its target node. The following *f* hops are via finger entries. The last hop is via leaf set. Each side of leaf set has $l/2$ entries, $(N/4r)/2^f = l/2$, $f = \log_2(N/2rl)$. The total number of hops is $h = f + 2 = \log_2(2N/rl)$. When $N = 1{,}000{,}000$, $h \approx 2.12$. This is much fewer than that of typical 16-based Pastry ($\log_{16} N \approx 4.98$).

2.3 Joining & Warm up

A node X's joining process is as follows. 1) X contacts one existing node B, which called X's "bootstrap". Suppose the level of B is k_B, and the bandwidth used to update B's routing table is W_B. Thus the highest level of X is $k_{\max} = \lceil k_B + \log_2(W_B / W_X) \rceil$. 2) X gets its *k* top entries from one of *B*'s top nodes. 3) X downloads its own routing table from its top nodes. The download may consume too much bandwidth. To relieve the pressure on top nodes, the downloading can be redirected to another supper node or even different supper nodes concurrently.

A node can change its level freely to adapt to the bandwidth fluctuation. A node can debate its level by reducing routing entries and reporting to top nodes. Changing to an upper level requires the node to download some routing entries from its supper nodes additionally.

One serious drawback of one-hop overlay is its long starting up process. In a system having 100,000 nodes, a modem-linked node has to take 5 minutes to download its routing table even using up all of its 64 kbps bandwidth. If the system scale reaches 1,000,000, above process takes 50 minutes.

SmartBoa adopts a "*warm up*" process to hide this boring period. When a node joins, it can select a lower level in which the downloading can be done in a few seconds. Then the node runs in this level temporarily with the downloading going on in background, which may take several minutes. After the downloading is accomplished, the node elevates its level to normal.

3 Conclusion

Considering the great heterogeneous between nodes in p2p systems, we want to achieve the highest efficiency by making full utilize of every node's permitting bandwidth. SmartBoa is a beginning step, which relates the routing table's size and maintenance overhead to the node's capacity. Without a formalized proving, we cannot determine the difference between routing efficiencies of SmartBoa and the optimum algorithm. However, SmartBoa do have much higher routing efficiency and better scalability than previous overlays.

Generally speaking, SmartBoa has following good qualities:

a) Fully utilize nodes' available bandwidth.
b) Without the uniform bandwidth requirement, any node can join the overlay.
c) Nodes can change their levels freely to adapt to the fluctuation of network condition.
d) Simple but effective routing algorithm
e) Routing does not overburden any section of nodes
f) Remarkable scalability
g) Using warm up process to hide the long starting up period.
h) Can provide information about nodes' capabilities to upper applications.

There are still some open problems with SmartBoa:

a) Incentive mechanism. Without an incentive, Users tend to remain in lower level because of the high maintenance overhead in higher level. In fact, this is an open problem to the whole p2p realm.
b) How to pack messages in multicast to decrease the overhead of IP address and UDP message head.
c) Relate routing entries to the network layer to increase the efficiency of every hop.

References

[1] Anjali Gupta, Barbara Liskov, Rodrigo Rodrigues. One Hop Lookups for Peer-to-Peer Overlays. HOTOS IX. May 2003.

[2] Nicholas J.A. Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, Alec Wolman. SkipNet: A Scalable Overlay Network with Practical Locality Properties. USITS 2003. March 2003.

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In Proc. of ACM SIGCOMM, Aug. 2001.

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

[5] SAROIU, S., GUMMADI, P. K., AND GRIBBLE, S. D. A Measurement Study of Peer-to-Peer File Sharing Systems. In Proceedings of MMCN'02. San Jose, CA, Jan. 2002.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the ACM SIGCOMM '01 Conference, San Diego, California, August 2001.

[7] Ben Zhao, John Kubiatowicz, and Anthony Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley. April 2001.