Brocade: Landmark Routing on Overlay Networks

Ben Y. Zhao, Yitao Duan, Ling Huang
Anthony D. Joseph, and John D. Kubiatowicz
Computer Science Division, U. C. Berkeley
{ravenben, duan, hlion, adj, kubitron}@cs.berkeley.edu

1 Introduction

Existing peer-to-peer overlay infrastructures such as Tapestry [11], Chord [8], Pastry [6] and CAN [4] demonstrated the benefits of scalable, wide-area lookup services for Internet applications. These architectures make use of name-based routing to route requests for objects or files to a nearby replica. Applications built on such systems [2, 3, 7], depend on reliable and fast message routing to a destination node, given some unique identifier.

Due to the theoretical approach taken in these systems, however, they assume that most nodes in the system are uniform in resources such as network bandwidth and storage. This results in messages being routed on the overlay with minimum consideration to actual network topology and differences between node resources.

In *Brocade*, we propose a secondary overlay to be layered on top of these systems, that exploits knowledge of underlying network characteristics. The secondary overlay builds a location layer between "supernodes," nodes that are situated near network access points, such gateways to administrative domains. By associating local nodes with their nearby "supernode," messages across the wide-area can take advantage of the highly connected network infrastructure between these supernodes to shortcut across distant network domains, greatly improving point-to-point routing distance and reducing network bandwidth usage.

In this paper, we present the initial architecture of a brocade secondary overlay on top of a Tapestry network, and demonstrate its potential performance benefits by simulation. Section 2 briefly describes Tapestry routing and location, Section 3 describes the design of a Tapestry brocade, and Section 4 present preliminary simulation results. Finally, we discuss related work and conclude in Section 5.

2 Tapestry Routing and Location

Our architecture leverages Tapestry, an overlay location and routing layer presented by Zhao, Kubiatowicz and Joseph in [11]. Tapestry is one of several recent projects exploring the value of wide-area decentralized location services [4, 6, 8]. It allows messages to locate objects and route to them across an arbitrarily-sized network, while using a routing map with size logarithmic to the network namespace at each hop. We present here a brief overview of the relevant characteristics of Tapestry. A detailed discussion of its algorithms, fault-tolerant mechanisms and simulation results can be found in [11].

Each Tapestry node can take on the roles of *server* (where objects are stored), *router* (which forward messages), and *client* (origins of requests). Objects and nodes have names independent of their location and semantic properties, in the form of random fixed-length bit-sequences with a common base (e.g., 40 Hex digits representing 160 bits). The system assumes entries are roughly evenly distributed in both node and object namespaces, which can be achieved by using the output of secure one-way hashing algorithms, such as SHA-1.

2.1 Routing Layer

Tapestry uses local routing maps at each node, called *neighbor maps*, to incrementally route overlay messages to the destination ID digit by digit (e.g., `***8` $\Longrightarrow$ `**98` $\Longrightarrow$ `*598` $\Longrightarrow$ `4598` where *'s represent wildcards). This approach is similar to longest prefix routing in the CIDR IP address allocation architecture [5]. A node N has a neighbor map with multiple levels, where each level represents a matching suffix up to a digit position in the ID. A given level of the neighbor map contains a number of entries equal to the base of the ID, where the ith entry in the jth level is the ID and location of the closest node which ends in "i"+suffix(N, $j-1$). For example, the 9th entry of the 4th level for node `325AE` is the node closest to `325AE` in network distance which ends in `95AE`.

When routing, the nth hop shares a suffix of at least length n with the destination ID. To find the next router, we look at its $(n+1)$th level map, and look up the entry matching the value of the next digit in the destination ID. Assuming consistent neighbor maps, this routing method guarantees that any existing unique node in the system will be found within at most $Log_b N$ logical hops, in a system with an N size namespace using IDs of base b. Since every neighbor map level assumes that the preceding digits all match the current node's suffix, it only needs to keep a small constant size (b) entries at each route level, yielding a neighbor map of fixed size $b \cdot Log_b N$. Figure 1 shows an example of hashed-suffix routing.

2.2 Data Location

Tapestry employs this infrastructure for data location. Each object is associated with one or more *Tapestry location roots* through a distributed deterministic mapping function. To advertise or publish an object O, the server S storing the object sends a publish message toward the Tapestry location root for that object. At each hop along the way, the publish message stores location information in the form of a mapping <Object-ID(O), Server-ID(S)>. Note that these mappings are simply pointers to the server S where O is being stored, and not a copy of the object itself. Where multiple objects exist, each server maintaining a replica publishes its copy. A node N that keeps location mappings for multiple replicas keeps them sorted in order of distance from N.

During a location query, clients send messages directly to objects via Tapestry. A message destined for O is initially routed towards O's root from the client. At each hop, if the message encounters a node that contains the location mapping for O, it is redirected to the server containing the object. Otherwise, the message is forward one step closer to the root. If the message reaches the root, it is guaranteed to find a mapping for the location of O. Note that the hierarchical nature of Tapestry routing means at each hop towards the root, the number of nodes satisfying the next hop constraint decreases by a factor equal to the

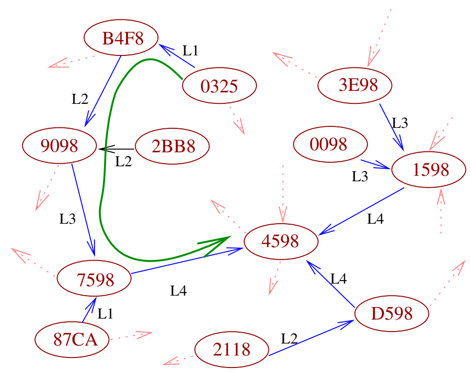

Figure 1: *Tapestry routing example.* Path taken by a message from node `0325` for node `4598` in Tapestry using hexadecimal digits of length 4 (65536 nodes in namespace).

identifier base (e.g. octal or hexadecimal) used in Tapestry. For nearby objects, client search messages quickly intersect the path taken by publish messages, resulting in quick search results that exploit locality. These and other properties are analyzed and discussed in more detail in [11].

3 Brocade Base Architecture

Here we present the overall design for the brocade overlay proposal, and define the design space for a single instance of the brocade overlay. We further clarify the design issues by presenting algorithms for an instance of a Tapestry on Tapestry brocade.

To improve point to point routing performance on an overlay, a brocade system defines a secondary overlay on top of the existing infrastructure, and provides a shortcut routing algorithm to quickly route to the local network of the destination node. This is achieved by finding nodes which have high bandwidth and fast access to the wide-area network, and tunnelling messages through an overlay composed of these "supernodes."

In overlay routing structures such as Tapestry [11], Pastry [6], Chord [8] and Content-Addressable Networks [4], messages are often routed across multiple autonomous systems (AS) and administrative domains before reaching their destinations. Each overlay hop often incurs long latencies within and across multiples AS's, consuming bandwidth along the way. To minimize both latency and network hops and reduce network traffic for a given message, brocade attempts to determine the network domain of the destination, and route directly to that domain. A "supernode" acts as a landmark for each network domain. Messages use them as endpoints of a tunnel through the secondary overlay, where messages would emerge near the local network of the destination node.

Before we examine the performance benefits, we address several issues necessary in constructing and utilizing a brocade overlay. We first discuss the construction of a brocade: how are supernodes chosen and how is the association between a node and its nearby supernode maintained? We then address issues in brocade routing: when and how messages find supernodes, and how they are routed on the secondary overlay.

3.1 Brocade Construction

The key to brocade routing is the tunnelling of messages through the wide area between landmark nodes (supernodes). The selection criteria are that supernodes have significant processing power (in order to route large amounts of overlay traffic), minimal number of IP hops to the wide-area network, and high bandwidth outgoing links. Given these requirements, gateway routers or machines close to them are attractive candidates. The final choice of a supernode can be resolved by an election algorithm between Tapestry nodes with sufficient resources, or as a performance optimizing choice by the responsible ISP.

Given a selection of supernodes, we face the issue of determining one-way mappings between supernodes and normal tapestry nodes for which they act as landmarks in Brocade routing. One possibility is to exploit the natural hierarchical nature of network domains. Each network gateway in a domain hierarchy can act as a brocade routing landmark for all nodes in its subdomain not covered by a more local subdomain gateway. We refer to the collection of these overlay nodes as the supernode's *cover set*. An example of this mapping is shown in Figure 2. Supernodes keep up-to-date member lists of their cover sets, which are used in the routing process, as described below.

A secondary overlay can then be constructed on supernodes. Supernodes can have independent names in the brocade overlay, with consideration to the overlay design, e.g. Tapestry location requires names to be evenly distributed in the namespace.

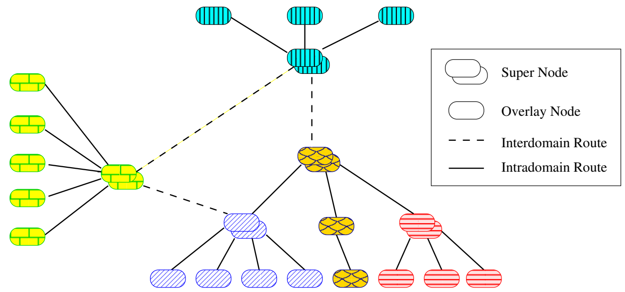

Figure 2: Example of Supernode Organization

3.2 Brocade Routing

Here we describe mechanisms required for a Tapestry-based brocade, and how they work together to improve long range routing performance. Given the complexity and latency involved in routing through an additional overlay, three key issues are: how are messages filtered so that only long distance messages are directed through the brocade overlay, how messages find a local supernode as entry to the brocader, and how a message finds the landmark supernode closest to the message destination in the secondary overlay.

3.2.1 Selective Utilization

The use of a secondary overlay incurs a non-negligible amount of latency overhead in the routing. Once a message reaches a supernode, it must search for the supernode nearest to the destination node before routing to that domain and resuming Tapestry routing to the destination. Consequently, only messages that route outside the reach of the local supernode benefit from brocade routing.

We propose a naive solution by having each supernode maintain a listing of all Tapestry nodes in its cover set. We expect the node list at supernodes to be small, with a maximum size on the order of tens of thousands of entries. When a message reaches a supernode, the supernode can do an efficient lookup (via hashtable) to determine whether the message is destined for a local node, or whether brocade routing would be useful.

3.2.2 Finding Supernodes

For a message to take advantage of brocade routing, it must be routed to a supernode on its way to its destination. How this occurs plays a large part in how efficient the resulting brocade route is. There are several possible approaches. We discuss three possible options here, and evaluate their relative performance in Section 4.

Naive A naive approach is to make brocade tunnelling an optional part of routing, and consider it only when a message reaches a supernode as part of normal routing. The advantage is simplicity. Normal nodes need to do nothing to take advantage of brocade overlay nodes. The disadvantage is that it severely limits the set of supernodes a message can reach. Messages can traverse several overlay hops before encountering a supernode, reducing the effectiveness of the brocade overlay.

IP-snooping In an alternate approach, supernodes can "snoop" on IP packets to determine if they are Tapestry messages. If so, supernodes can parse the message header, and use the destination ID to determine if brocade routing should be used. The intuition is that because supernodes are situated near the edge of local networks, any Tapestry message destined for an external node will likely cross its path. This also has the advantage that the source node sending the message need not know about the brocade supernodes in the infrastructure. The disadvantage is difficulty in implementation, and possible limitations imposed on regular traffic routing by header processing.

Directed The most promising solution is for overlay nodes to find the location of their local supernode, by using DNS resolution of a well-known name, e.g. `supernode.cs.berkeley.edu`, or by an expanding ring search. Once a new node joins a supernode's cover set, state can be maintained by periodic beacons. To reduce message traffic at supernodes, nodes keep a local *proximity cache* to "remember" local nodes they have communicated with. For each new message, if the destination is found in the proximity cache, it is routed normally. Otherwise, the node sends it directly to the supernode for routing. This is a proactive approach that takes advantage of any potential performance benefit brocade can offer. It does, however, require state maintenance, and the use of explicit fault-tolerant mechanisms should a supernode fail.

3.2.3 Landmark Routing on Brocade

Once an inter-domain message arrives at the sender's supernode, brocade needs to determine the supernode closest to the message destination. This can be done by organizing the brocade overlay as a Tapestry network. As described in Section 2.2 and [11], Tapestry location allows nodes to efficiently locate objects given their IDs. Recall that each supernode keeps a list of all nodes inside its cover set. In the brocade overlay, each supernode advertises the IDs on this list as IDs of objects it "stores." When a supernode tries to route an outgoing inter-domain message, it uses Tapestry to search for an object with an ID identical to the message destination ID. By finding the object on the brocade layer, the source supernode has found the message destination's supernode, and forwards the message directly to it. The destination supernode then resumes normal overlay routing to the destination.

Note these discussions make the implicit assumption that on average, inter-domain routing incurs much higher latencies compared to intra-domain routing. This, in combination with the distance constraints in Tapestry, allows us to assert that intra-domain messages will never route outside the domain. This is because the destination node will almost always offer the closest node with its own ID. This also means that once a message arrives at the destination's supernode, it will quickly route to the destination node.

4 Evaluation of Base Design

In this section, we present some analysis and initial simulation results showing the performance improvement possible with the use of brocade. In particular, we simulate the effect brocade routing has on point to point routing latency and bandwidth usage. For our experiments, we implemented a two layer brocade system inside a packet-level simulator that used Tapestry as both the primary and secondary overlay structures. The packet level simulator measured the progression of single events across a large network without regard to network effects such as congestion or retransmission.

Before presenting our simulation results, we first offer some back-of-the-envelope numerical support for why brocade supernodes should scale with the size of AS's and the rate of nodes entering and leaving the Tapestry. Given the size of the current Internet around 204 million nodes[1], and 20000 AS's, we estimate the size of an average AS to be around

[1] Source: `http://www.netsizer.com/`

10,000 nodes. Also, our current implementation of Tapestry on a PIII 800Mhz node achieves throughput of 1000 messages/second. In a highly volatile AS of 10000 nodes, where 10% of nodes enter or leave every minute, roughly 1.7% of the supernode processing power is used for handling the "registration" of new nodes.

We used in our experiments GT-ITM [10] transit stub topologies of 5000 nodes. We constructed Tapestry networks of size 4096, and marked 16 transit stubs as brocade supernodes. We then measured the performance of pair-wise communication paths using original Tapestry and all three brocade algorithms for finding supernodes (Section 3.2.2). We include four total algorithms: 1. original Tapestry, 2. naive brocade, 3. IP-snooping brocade, 4. directed brocade. For brocade algorithms, we assume the sender knows whether the destination node is local, and only uses brocade for inter-domain routing.

We use as our key metric a modified version of Relative Delay Penalty (RDP) [1]. Our modified RDP attempts to account for the processing of an overlay message up and down the protocol stack by adding 1 hop unit to each overlay node traversed. Each data point is generated by averaging the routing performance on 100 randomly chosen paths of a certain distance. In the RDP measurements, the sender's knowledge of whether the destination is local explains the low RDP values for short distances, and the spike in RDP around the average size of transit stub domains.

We measured the hop RDP of the four routing algorithms. For each pair of communication endpoints A and B, hop RDP is a ratio of # of hops traversed using brocade to the ideal hop distance between A and B. As seen in Figure 3, all brocade algorithms improve upon original Tapestry point to point routing. As expected, naive brocade offers minimal improvement. IP snooping improves the hop RDP substantially, while directed brocade provides the most significant improvement in routing performance. For paths of moderate to long lengths, directed brocade reduces the routing overhead by more than 50% to near optimal levels (counting processing time). The small spike in RDP for IP snooping and directed brocade is due to the Tapestry location overhead in finding landmarks for destinations in nearby domains.

Figure 3 makes a simple assumption that all physical links have the same latency. To account for the fact that interdomain routes have higher latency, Figure 4 shows an RDP where each interdomain hop counts as 3 hop units of latency. We see that IP snooping and directed brocade still show the drastic improvement in RDP found in the simplistic topology results. We note that the spike in RDP experienced by IP snooping and directed brocade is exacerbated by the effect of higher routing time in interdomain traffic making Tapestry location more expensive. We also ran this test on several transit stub topologies with randomized latencies direct from GT-ITM, with similar results.

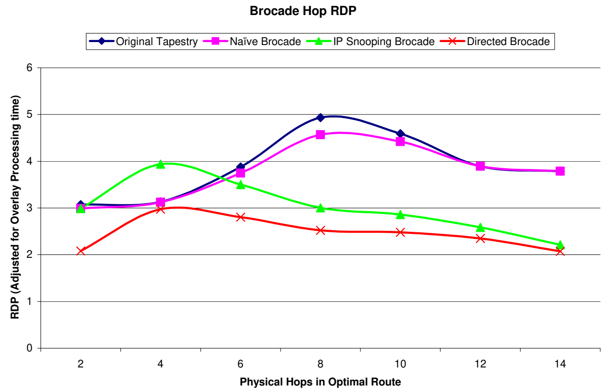

Figure 3: Hop-based RDP

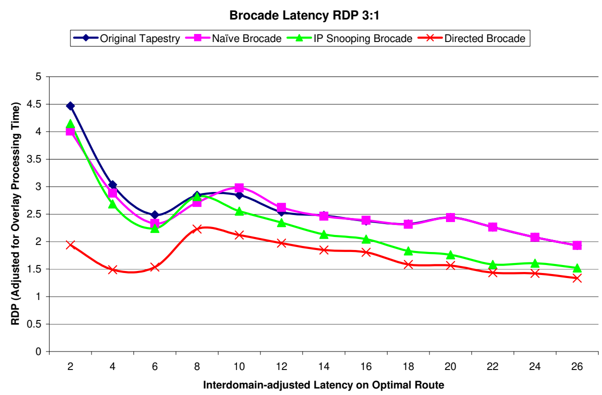

Figure 4: Weighted latency RDP, ratio 3:1

Finally, we examine the effect of brocade on reducing overall network traffic, by measuring the aggregate bandwidth taken per message delivery, using units of (sizeof(Msg) * hops). The result in Figure 5 shows that IP snooping brocade and directed brocade dramatically reduce bandwidth usage per message

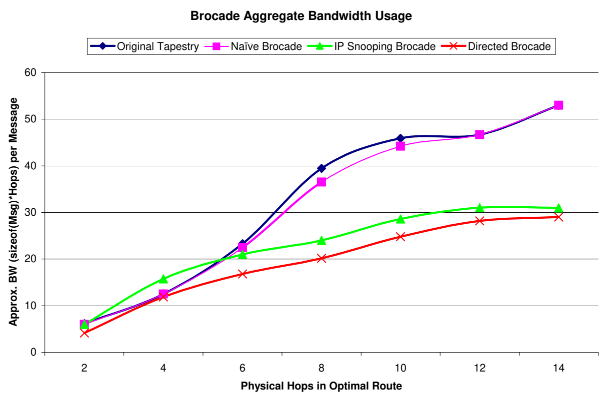

Figure 5: Aggregate bandwidth used per message

delivery. This is expected, since brocade forwards messages directly to the destination domain, and reduces message forwarding on the wide-area.

While certain decisions in our design are Tapestry specific, we believe similar design decisions can be made for other overlay networks [4, 6, 8], and these results should apply to brocade routing on those networks as well.

5 Related Work and Status

In related work, the Cooperative File System [2] leverages nodes with more resources by allowing them to host additional virtual nodes in the system, each representing one quantum of resource. This quantification is directed mostly at storage requirements, and CFS does not propose a mechanism for exploiting network topology knowledge. Our work is also partially inspired by the work on landmark routing [9], where packets are directed to a node in the landmark hierarchy closest to the destination before local routing.

While we present an architecture here using Tapestry at the lower level, the brocade overlay architecture can be generalized on top of any peer-to-peer network infrastructure. The presented architecture works as is on top of the Pastry [6] network. We are currently exploring brocades on top of CAN [4] and Chord [8]. We are implementing brocade in the Tapestry/OceanStore code base, and are experimenting with alternative efficient mechanisms for locating landmark nodes.

In conclusion, we have proposed the use of a secondary overlay network on a collection of well-connected "supernodes," in order to improve point to point routing performance on peer-to-peer overlay networks. The brocade layer uses Tapestry location to direct messages to the supernode nearest to their destination. Simulations show that brocade significantly improves routing performance and reduces bandwidth consumption for point to point paths in a wide-area overlay. We believe brocade is an interesting enhancement that leverages network knowledge for enhanced routing performance.

References

[1] CHU, Y., RAO, S. G., AND ZHANG, H. A case for end system multicast. In *Proceedings of ACM SIGMETRICS* (June 2000), pp. 1–12.

[2] DABEK, F., KAASHOEK, M. F., KARGER, D., MORRIS, R., AND STOICA, I. Wide-area cooperative storage with CFS. In *Proceedings of SOSP* (October 2001), ACM.

[3] KUBIATOWICZ, J., ET AL. OceanStore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS* (November 2000), ACM.

[4] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SCHENKER, S. A scalable content-addressable network. In *Proceedings of SIGCOMM* (August 2001), ACM.

[5] REKHTER, Y., AND LI, T. An architecture for IP address allocation with CIDR. RFC 1518, `http://www.isi.edu/in-notes/rfc1518.txt`, 1993.

[6] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of IFIP/ACM Middleware 2001* (November 2001).

[7] ROWSTRON, A., AND DRUSCHEL, P. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of SOSP* (October 2001), ACM.

[8] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM* (August 2001), ACM.

[9] TSUCHIYA, P. F. The landmark hierarchy: A new hierarchy for routing in very large networks. *Computer Communication Review 18*, 4 (August 1988), 35–42.

[10] ZEGURA, E. W., CALVERT, K., AND BHATTACHARJEE, S. How to model an internetwork. In *Proceedings of IEEE INFOCOM* (1996).

[11] ZHAO, B. Y., KUBIATOWICZ, J. D., AND JOSEPH, A. D. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, UC Berkeley, EECS, 2001.