Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination

Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, John Kubiatowicz
Computer Science Division, University of California, Berkeley
{shelleyz, ravenben, adj, randy, kubi}@cs.berkeley.edu

Abstract- **The demand for streaming multimedia applications is growing at an incredible rate. In this paper, we propose Bayeux, an efficient application-level multicast system that scales to arbitrarily large receiver groups while tolerating failures in routers and network links. Our simulation results indicate that Bayeux maintains these properties while keeping transmission overhead low. Bayeux leverages the architecture of Tapestry, a fault-tolerant, wide-area overlay routing and location network.**

I. Introduction

The demand for streaming multimedia applications is growing at an incredible rate. Such applications are distinguished by a single writer (or small number of writers) simultaneously feeding information to a large number of readers. Current trends would indicate a need to scale to thousands or millions of receivers. To say that such applications stress the capabilities of wide-area networks is an understatement. When millions of receiving nodes are involved, unicast is completely impractical because of its redundant use of link bandwidth; to best utilize network resources, receivers must be arranged in efficient communication trees. This in turn requires the efficient coordination of a large number of individual components, leading to a concomitant need for resilience to node and link failures.

Given barriers to wide-spread deployment of IP multicast, researchers have turned to application-level solutions. The major challenge is to build an efficient network of unicast connections and to construct data distribution trees on top of this overlay structure. Currently, there are no designs for application-level multicast protocols that scale to thousands of members, incur both minimal delay and bandwidth penalties, and handle faults in both links and routing nodes.

In this paper we propose Bayeux, an efficient, application-level multicast system that has these properties. One of the novel aspects of Bayeux is that it combines randomness for load balancing with locality for efficient use of network bandwidth. Further, this protocol makes use of controlled redundancy to provide resilience to link and router failures. Bayeux utilizes a prefix-based routing scheme that it inherits from an existing application-level routing protocol called Tapestry [10], a wide-area location and routing architecture for OceanStore [3]. On top of Tapestry, Bayeux provides a simple protocol that organizes the multicast receivers into a distribution tree rooted at the source. Simulation results indicate that Bayeux scales well beyond thousands of multicast nodes in terms of the overlay latency and redundant packet duplication for a variety of topology models.

In the rest of this paper we discuss the architecture of Bayeux and provide simulation results. First, Section II describes the Tapestry routing and location infrastructure. Next, Section III describes the Bayeux architecture, followed by Section IV which evaluates it. In Section V, we explore some novel advantages of the Bayeux approach. We conclude in Section VI.

II. Tapestry Routing and Location Layer

Our architecture leverages Tapestry, a location and routing layer presented by Zhao, Kubiatowicz and Joseph in [10]. Bayeux uses the natural hierarchy of Tapestry routing to forward packets while conserving bandwidth. Multicast group members wishing to participate in a Bayeux session become (if not already) Tapestry nodes, and a data distribution tree is built on top of this overlay structure.

The Tapestry location and routing infrastructure uses similar mechanisms to the hashed-suffix mesh introduced by Plaxton, Rajaraman and Richa in [4]. It is novel in allowing 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. Tapestry provides a delivery time within a small factor of the optimal delivery time, from any point in the network. A detailed discussion of Tapestry and simulation results can be found in [10].

Each Tapestry node or machine can take on the roles of *server* (where objects are stored), *router* (which forward messages), and *client* (origins of requests). Also, objects and nodes have names independent of their location and semantic properties, in the form of random fixed-length bit-sequences represented by 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 hashing algorithms, such as SHA-1 [6].

A. 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. Because every single neighbor map at a node 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 constant size $b \cdot Log_b N$.

A way to visualize this routing mechanism is that every destination node is the *root node* of its own tree, which is a unique spanning tree across all nodes. Any leaf can traverse a number of intermediate nodes en route to the root node. In short, the hashed-suffix mesh of neighbor maps is a large set of embedded trees in the network, one rooted at every node. Figure 1 shows an example of hashed-suffix routing.

In addition to providing a scalable routing mechanism, Tapestry also provides a set of fault-tolerance mechanisms which allow routers to quickly route around link and node failures, while guaranteeing successful delivery of packets.

B. Data Location

Tapestry employs this infrastructure for data location in a straightforward way. Each object is associated with one or more *Tapestry location roots* through a deterministic mapping function. To advertise an object, a server sends a publish message toward the Tapestry location root for that object, which leaves a pointer behind at each hop along the way. Messages are then routed to the closest copy of the object by performing Tapestry routing toward the Tapestry location root until one of the pointers is encountered.

C. Benefits

Tapestry provides the following benefits:

- *Powerful Fault Handling*: Tapestry provides multiple paths to every destination. This enables fast failover and recovery.
- *Scalable*: Tapestry routing is inherently decentralized, and all routing is done using local information. Routing tables have a small logarithmic size, guaranteeing scalability as the network scales.

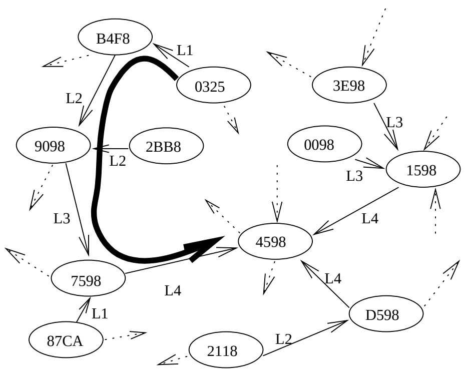

Fig. 1. *Tapestry routing example.* Here we see the path taken by a message originating from node `0325` destined for node `4598` in a Tapestry network using hexadecimal digits of length 4 (65536 nodes in namespace).

- *Proportional Route Distance*: It follows from Plaxton et al.'s proof in [4] that the network distance traveled by a message during routing is proportional to the real underlying network distance, assuring us that routing on the Tapestry overlay incurs a reasonable overhead.

D. Multicast on Tapestry

The nature of Tapestry unicast routing provides a natural ground for building an application-level multicasting system. Tapestry overlay assists efficient multi-point data delivery by following suffixes of node IDs. The node ID base defines the fanout factor used in the multiplexing of data packets to different paths on each router. Because randomized node IDs naturally group themselves into sets sharing common suffixes, we can use that common suffix to minimize transmission of duplicate packets. A multicast packet only needs to be duplicated when the receiver node identifiers become divergent in the next digit. In addition, the maximum number of overlay hops taken by such a delivery mechanism is bounded by the total number of digits in the Tapestry node IDs. For example, in a Tapestry namespace size of 4096 with an octal base, the maximum number of overlay hops from a source to a receiver is 4. The amount of packet fan-out at each branch point is limited to the node ID base. This fact hints at a natural multicast mechanism on the Tapestry infrastructure.

III. BAYEUX BASE ARCHITECTURE

Bayeux provides a source-specific, explicit-join multicast service. The source-specific model has numerous practical advantages and is advocated by a number of projects [2], [7], [8], [9]. A Bayeux multicast session is identified by the tuple <root node ID, session name>, where an ID is a fixed-length bit sequence in the Tapestry node namespace that identifies the root/source node. The session name helps distinguish between multiple simultaneous sessions hosted by a single root node.

A. *Session Advertisement*

We leverage Tapestry's data location algorithm for advertising Bayeux sessions. An advertisement is a tuple containing the root node ID and the multicast session name. The root or source server of a session publishes advertisement (<root node ID, session name>) using the Tapestry publication mechanism. Clients can rely on Tapestry to quickly locate the advertisement block given the session name.

B. *Tree Maintenance*

Self organization of session members into an efficient robust distribution tree is the key to efficient operation in any application-level multicasting system. The tree building protocol in a Bayeux session begins at the root server associated with each session.

There are four types of control messages in building a distribution tree: `JOIN, LEAVE, TREE, PRUNE`. A member joins the multicast session by sending a `JOIN` message towards the root, which then replies with a `TREE` message. When a router receives a `TREE` message, it adds the new member node ID to the list of receiver node IDs that it is responsible for, and updates its forwarding table. For example, consider node xx50 on the path from the root node to node 1250. Upon receiving the `TREE` message from the root, node xx50 will add 1250 into its receiver ID list, and will duplicate and forward future packets for this session to node x250. Similarly, a `LEAVE` message from an existing member triggers a `PRUNE` message from the root, which trims from the distribution tree any routers whose forwarding states become empty after the leave operation.

IV. Evaluation of Base Architecture

Here, we compare the basic Bayeux algorithm against IP multicast and naive unicast. By naive unicast we mean a unicast star topology rooted at the source that performs one-to-one transmission to all receivers. In Section V, we introduce Tapestry-enabled enhancements to Bayeux and demonstrate their effect on performance. [Reviewers: the full version discusses the experimental setup in detail.]

A. *Performance Metrics*

We adopt the two metrics proposed in [1] to evaluate the effectiveness of our application-level multicast technique:

- *Relative Delay Penalty*, a measure of the increase in delay that applications incur while using overlay routing. For Bayeux, it is the ratio of Tapestry unicast routing distances to IP unicast routing distances. Assuming symmetric routing, IP Multicast and naive unicast both have a RDP of 1.
- *Physical Link Stress*, a measure of how effective Bayeux is in distributing network load across different physical links. It refers to the number of identical copies of a packet carried by a physical link. IP multicast has a stress of 1, and naive unicast has a worst case stress equal to number of receivers.

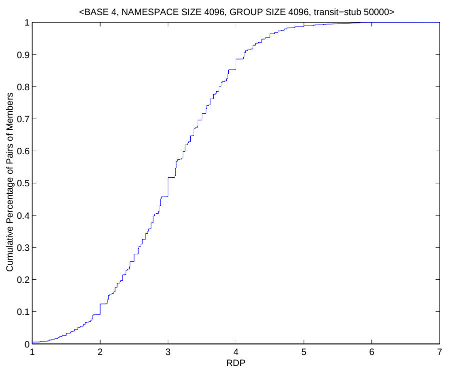

Fig. 2. Cumulative distribution of RDP

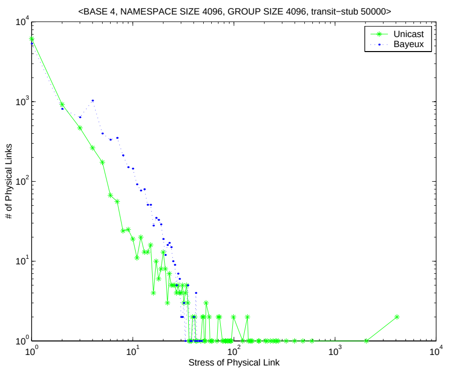

Fig. 3. Comparing number of stressed links between naive unicast and Bayeux using Log scale on both axis.

B. *Snapshot Measurements*

In this experiment, we used a topology generated by the transit-stub model consisting of 50000 nodes, with a Tapestry overlay using node namespace size of 4096, ID base of 4, and a multicast group size of 4096 members. Figure 2 plots the cumulative distribution of RDP on this network. RDP is measured for all pairwise connections between nodes in the network. As we can see, the RDP for a large majority of connections is quite low.

In Figure 3, we compare the variation of physical link stress in Bayeux to that under naive unicast. We define the stress value as the number of duplicate packets going across a single physical link. We pick random source nodes with random receiver groups, and measure the worst stress value of all links in the tree built. We plot the number of links suffering from a particular stress level on the Y-axis, against the range of stress levels on the X-axis. We see that relative to unicast, the overall distribution of link stress is substantially lower. In addition, naive unicast exhibits a much longer tail, where certain links experience stress levels up to 4095, whereas the Bayeux measurement shows no such outliers. This shows that Bayeux distributes the network load evenly across physical links, even for large multicast groups. While End System Multicast [1] also exhibits low

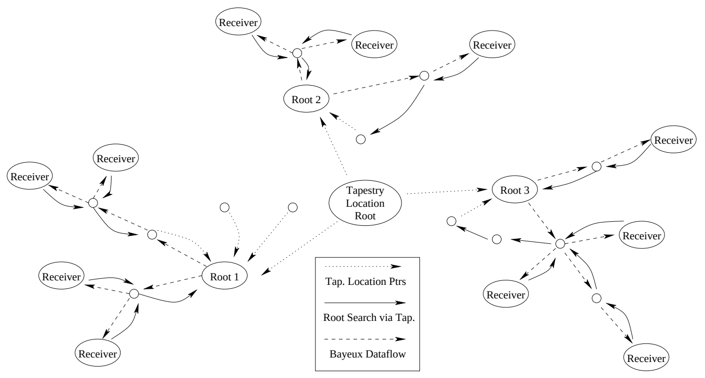

Fig. 4. Receivers self-configuring into Tree Partitions

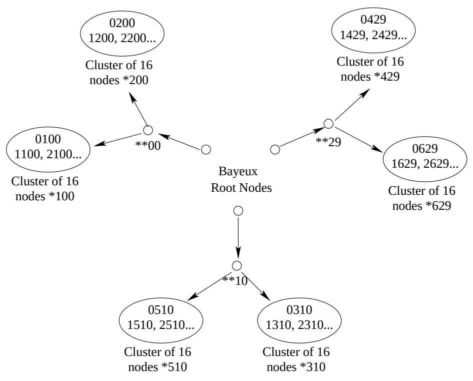

Fig. 5. Receiver ID Clustering according to network distance

physical link stress, it only scales to receiver groups of hundreds.

V. Tapestry-enabled Mechanisms and Benefits

In this section, we demonstrate the advantages Bayeux gains from leveraging Tapestry-specific properties, in the form of optimizations *Tree Partitioning*, *Receiver Clustering*, and *Multicast Fault-resilience*. We further quantify the resulting benefits via simulation.

A. Tree Partitioning

The source-specific service model has several drawbacks. First, the root of the multicast tree is a scalability bottleneck, as well as a single point of failure. Unlike existing multicast protocols, due to the non-symmetric routing in Bayeux, the root must handle all join and leave requests from session members. Second, only the root can send data in a source-specific service model. Although the root can act as a reflector for supporting multiple senders [2], all messages have to go through the root, and a network partition or root failure will compromise the entire group's ability to receive data.

To remove the root as a scalability bottleneck and point of failure, we propose a *Tree Partition* mechanism in Bayeux that leverages the Tapestry location mechanism. The idea is to create multiple root nodes, and partition receivers into disjoint membership sets, each containing receivers closest to a local root in network distance. Receivers organize themselves into these sets as follows: 1. Integrate Bayeux root nodes into a Tapestry network. 2. Name an object O with the hash of the multicast session name, and place O on each root. 3. Each root advertise O in Tapestry, storing pointers to itself at intermediate hops between it and the Tapestry location root, a node deterministically chosen based on O. 4. On `JOIN`, new member M uses Tapestry location to find and route a `JOIN` message to the nearest root node R. 5. R sends `TREE` message to M, now a member of R's receiver set.

Figure 4 shows the path of various messages in the tree

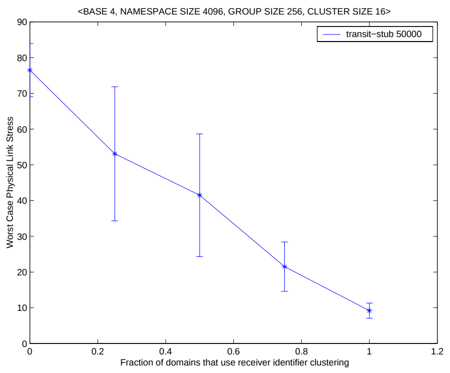

Fig. 6. Worst case physical link stress vs. fraction of domains that use receiver ID clustering for the transit-stub model

partitioning algorithm. Each member M sends location requests up to the Tapestry location root. Tapestry location guarantees M to find the closest such root [10], [4]. Root nodes then use Tapestry routing to forward packets to downstream routers, minimizing packet duplication where possible. The self-configuration of receivers into partitioned sets means root replication is an effcient tool to balance load between root nodes, and to reduce first hop latency to receivers when roots are placed near listerers.

B. Receiver Identifier Clustering

To further reduce packet duplication in Bayeux, we introduce the notion of receiver node ID clustering. Tapestry routing approaches the destination ID digit by digit, and a single packet is forwarded for all nodes sharing a suffix. Therefore, a naming scheme that provides an optimal packet duplication tree is one that allows local nodes to share the longest possible suffix. For instance, in a Tapestry 4-digit hexadecimal naming scheme, a group of 16 nodes in a LAN should be named by fixing the last 3 digits (`XYZ`), while assigning each node one of the 16 result numbers (`0XYZ, 1XYZ, 2XYZ,` etc.) This means upstream routers delay packet duplication until reaching the LAN, minimizing bandwidth consumption and reducing link stress. These groups of 16 nodes can be further organized into hierarchical groups of 16. Figure 5 shows such an example. While group sizes matching the Tapestry ID

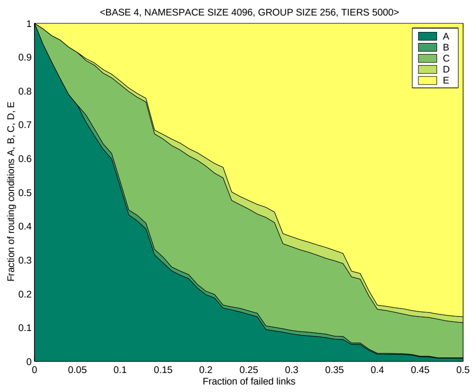

Fig. 7. Reachability Measures vs. Fraction of Failed Links in Physical Network

base are unlikely, clustered receivers of any size will show similar benefits. Also note that while Tapestry routing assumes randomized naming, organized naming on a small scale will not impact the efficiency of a wide-area system.

To quantify the effect of clustered naming, we measure link stress versus the fraction of local LANs that utilize clustered naming. We simulate 256 receivers on a Tapestry network using ID base of 4 and IDs of 6 digits. The simulated physical network is a transit stub modeled network of 50000 nodes, since it best represents the natural clustering properties of physical networks. Receivers are organized as 16 local networks, each containing 16 members. Figure 6 shows the dramatic decrease in worst cast link stress as node names become more organized in the local area. By correlating node proximity with naming, the duplication of a single source packet is delayed until the local router, reducing bandwidth consumption at all previous hops. The result shows an inverse relationship between worst cast link stress and local clustering.

C. Fault Resilience

Finally, we examine how Bayeux leverages Tapestry's routing redundancy to maintain delivery despite node and link failures. Each entry in the Tapestry neighbor map maintains secondary neighbors in addition to the closest primary neighbor. When a Bayeux node detects a next-hop node or link failure, it reroutes packets to one of its secondary neighbors along with an updated list of session members. This session state is soft-state, and times out if no data is forwarded within a timeout period. Note that the session state transmitted to each secondary neighbor is small. The worst case session state transmission occurs when the primary neighbor or link next to the root of the multicast tree fails. In a Tapestry network with 160-bit hexadecimal node IDs, the worst case session state for a multicast session with 1 million members is slightly over 1MB. However, by pre-staging the session state at the first-level secondary neighbors, we can reduce session state transmitted by a factor of 16 for each level pre-staged. Due to the hierarchical nature of Tapestry routing [10], packets forwarded to the secondary path will quickly converge to the original path.

Figure 7 shows Bayeux's fault-tolerance properties compared to IP routing. We used a topology generated by the TIERS model consisting of 5000 nodes and 7084 links. Results are similar for other topologies. We used a Tapestry node identifer namespace size of 4096, a base of 4, and a multicast group size of 256 members. Links are randomly dropped, and we monitor the reachability of IP and Tapestry routing. As link failures increase, region A shows probability of successful IP and Tapestry routing. Region C shows cases where IP fails and Tapestry succeeds. Region E represents cases where the destination is physically unreachable. Finally, region B shows instances where IP succeeds, and Tapestry fails; and region D shows where both protocols fail to route to a reachable destination.

The results show that when IP fails to route to a reachable node, Tapestry routing has an extremely high success rate. This can be explained by Tapestry routing "around" failures using secondary pointers.

VI. Conclusion

We have presented an architecture for Internet content distribution that leverages Tapestry, an existing fault-tolerant routing infrastructure. Bayeux achieves scalability, efficiency, and high degree of fault-tolerance by leveraging Tapestry-enabled mechanisms.

References

[1] Chu, Y. H., Rao, S. G., and Zhang, H. A case for end system multicast. In *Proceedings of SIGMETRICS* (June 2000).

[2] Holbrook, H. W., and Cheriton, D. R. Ip multicast channels: EXPRESS support for large-scale single-source applications. In *Proceedings of SIGMETRICS* (August 1999).

[3] Kubiatowicz, J., et al. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ASPLOS* (November 2000).

[4] Plaxton, C. G., Rajaraman, R., and Richa, A. W. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures(SPAA)* (June 1997).

[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] Robshaw, M. J. B. MD2, MD4, MD5, SHA and other hash functions. Tech. Rep. TR-101, RSA Labs, 1995. version 4.0.

[7] Source-specific multicast (SSM) working group at IETF. `http://sith.maoz.com/SSM`.

[8] Stoica, I., Ng, T. S. E., and Zhang, H. REUNITE: A recursive unicast approach to multicast. In *Proceedings of INFOCOM* (March 2000).

[9] Yano, K., and McCanne, S. The breadcrumb forwarding service: A synthesis of PGM and EXPRESS to improve and simplify global IP multicast. *ACM Comp. Comm. Review 30*, 2 (2000).

[10] Zhao, B. Y., Kubiatowicz, J., and Joseph, A. D. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Submitted for publication to SIGCOMM, `http://www.cs.berkeley.edu/~ravenben/tapestry.pdf`, 2001.