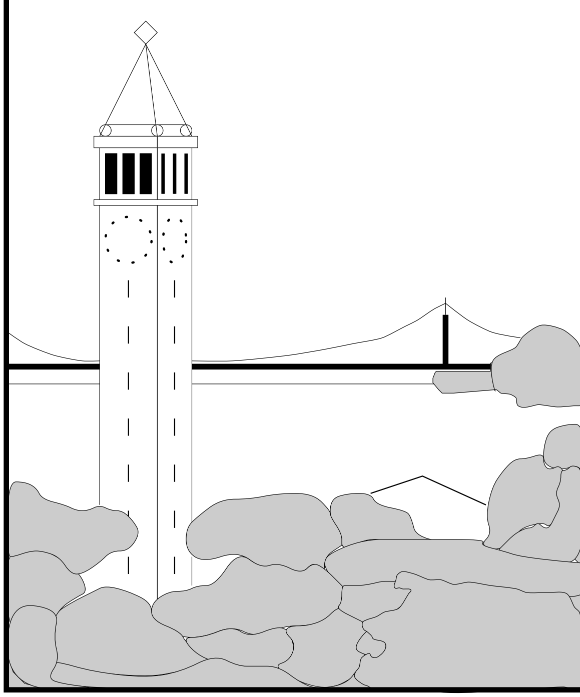
Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination

Shelley Q. Zhuang
shelleyz@eecs.berkeley.edu
CS Division, EECS Department, U.C.Berkeley



Report No. UCB/CSD-2-1170

January 2002

Computer Science Division (EECS)
University of California
Berkeley, California 94720

This technical report is supported by grant number DABT63-98-C-0038

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

Shelley Q. Zhuang
Computer Science Division
University of California at Berkeley

January 2002

1 Abstract

The demand for streaming multimedia applications is growing at a fast rate. In this report, we present Bayeux, an efficient application-level multicast system that scales to arbitrarily large receiver groups while tolerating failures in routers and network links. Bayeux also includes specific mechanisms for load-balancing across replicate root nodes and more efficient bandwidth consumption. Our simulation results indicate that Bayeux maintains these properties while keeping transmission overhead low (i.e., overlay routing latency is only 2-3 times of the physical shortest path latency and redundant packet duplication is a 85-fold improvement over naive unicast). To achieve these properties, Bayeux leverages the architecture of Tapestry, a fault-tolerant, wide-area overlay routing and location network.

2 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 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 report we present Bayeux, an efficient, source-specific, explicit-join, 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. Bayeux utilizes a prefix-based routing scheme that it inherits from an existing application-level routing protocol called Tapestry [36], a wide-area location and routing architecture used in the OceanStore [16] globally distributed storage system. 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 overlay latency and redundant packet duplication, for a variety of topology models.

In addition to the base multicast architecture, Bayeux leverages the Tapestry infrastructure to provide simple load-balancing across replicated root nodes, as well as reduced bandwidth consumption, by clustering receivers by identifier. The benefits of these optimizing mechanisms are shown in simulation results. Finally, Bayeux provides a variety of protocols to leverage the redundant routing structure of Tapestry. We evaluate one of them, *First Reachable Link Selection*, and show it to provide near-optimal fault-resilient packet delivery to reachable destinations, while incurring a low overhead in terms of membership state management.

In the rest of this report we discuss the architecture of Bayeux and provide simulation results. First, Section 3 describes the Tapestry routing and location infrastructure. Next, Section 4 describes the Bayeux architecture, followed by Section 5 which evaluates it. In Section 6, we explore novel scalability optimizations in Bayeux, followed by fault-resilient packet delivery in Section 7. We discuss related work in Section 8. Finally, we discuss future work and conclude in Section 10.

3 Tapestry Routing and Location

Our architecture leverages Tapestry, an overlay location and routing layer presented by Zhao, Kubiatowicz and Joseph in [36]. 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 [21]. 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 algorithms, its fault-tolerant mechanisms and simulation results can be found in [36].

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 sys-

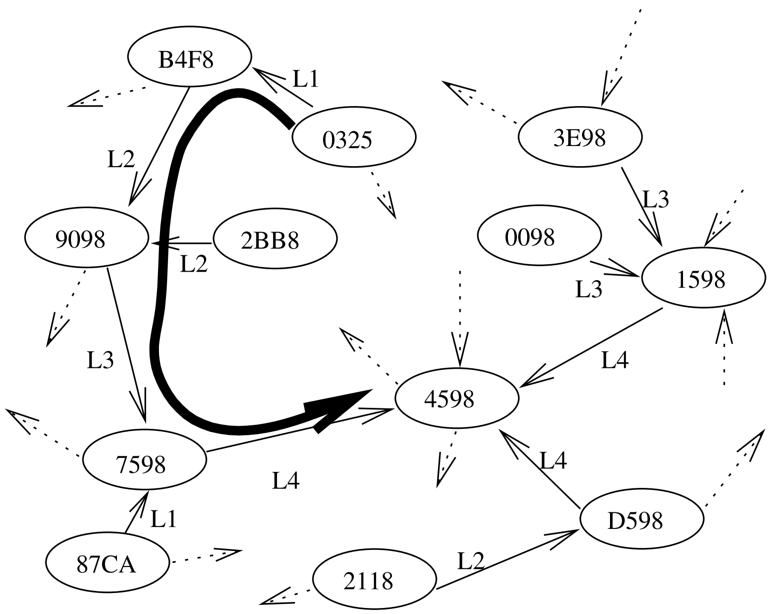


Figure 1: *Tapestry routing example.* Here we see the path taken by a message originating from node `0325` destined for node `4598` in a Tapestry network using hexadecimal digits of length 4 (65536 nodes in namespace).

tem 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 [25].

3.1 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 [24]. 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. Each entry in the neighbor map actually contains three entries that match the given suffix, where two secondary pointers are available if and when the primary route fails. These redundant routing paths are utilized by Bayeux protocols in Section 7.

3.2 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 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 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. Furthermore, by sorting distance to multiple replicas at intermediate hops, clients are likely to find the *nearest* replica of the desired object. These properties are analyzed and discussed in more detail in [36].

3.3 Benefits

Tapestry provides the following benefits:

- *Powerful Fault Handling*: Tapestry provides multiple paths to every destination. This mechanism enables application-specific protocols for fast failover and recovery.

- *Scalable*: Tapestry routing is inherently decentralized, and all routing is done using information from number of nodes logarithmically proportional to the size of the network. Routing tables also have size logarithmically proportionally to the network size, guaranteeing scalability as the network scales.
- *Proportional Route Distance*: It follows from Plaxton et al.'s proof in [21] that the network distance traveled by a message during routing is linearly proportional to the real underlying network distance, assuring us that routing on the Tapestry overlay incurs a reasonable overhead. In fact, experiments have shown this proportionality is maintained with a small constant in real networks [36].

3.4 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 forwarding packets according to suffixes of listener 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.

Note that unlike most existing application level multicast systems, not all nodes of the Tapestry overlay network are Bayeux multicast receivers. This use of dedicated infrastructure server nodes provides better optimization of the multicast tree and is a unique feature of the Bayeux/Tapestry system.

4 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 [13, 31, 33, 35]. A Bayeux multicast session is identified by the tuple <session name, UID>. A session name is a semantic name describing the content of the multicast, and the UID is a distinquishing ID that uniquely identifies a particular instance of the session.

4.1 Session Advertisement

We utilize Tapestry's data location services to advertise Bayeux multicast sessions. To announce a session, we take the tuple that uniquely names a multicast session, and use a secure one-way hashing

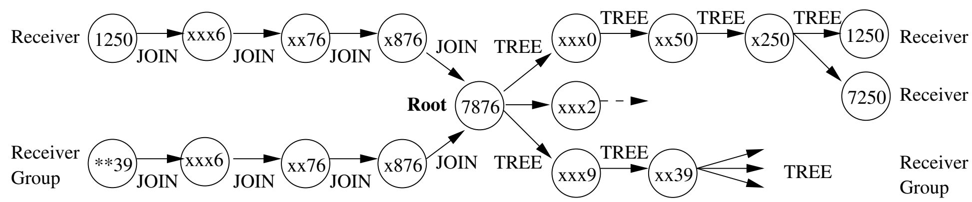


Figure 2: Tree maintenance

function (such as SHA-1 [25]) to map it into a 160 bit identifier. We then create a trivial file named with that identifier and place it on the multicast session's root node.

Using Tapestry location services, the root or source server of a session advertises that document into the network. Clients that want to join a session must know the unique tuple that identifies that session. They can then perform the same operations to generate the file name, and query for it using Tapestry. These searches result in the session root node receiving a message from each interested listener, allowing it to perform the required membership operations. As we will see in Section 6.1, this session advertisement scheme allows root replication in a way that is transparent to the multicast listeners.

4.2 Tree Maintenance

Constructing an efficient and robust distribution tree to deliver data to session members is the key to efficient operation in application-level multicast systems. Unlike most existing work in this space, Bayeux utilizes dedicated servers in the network infrastructure (in the form of Tapestry nodes) to help construct more efficient data distribution trees.

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. Figure 2 shows an example where node `7876` is the root of a multicast session, and node `1250` tries to join. The `JOIN` message from node `1250` traverses nodes `xxx6`, `xx76`, `x876`, and `7876` via Tapestry unicast routing, where `xxx6` denotes some node that ends with 6. The root `7876` then sends a `TREE` message towards the new member, which sets up the forwarding state at intermediate application-level routers. Note that while both control messages are delivered by unicasting over the Tapestry overlay network, the `JOIN` and `TREE` paths might be different, due to the asymmetric nature of Tapestry unicast routing.

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.

5 Evaluation of Base Design

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.

5.1 Simulation Setup

To evaluate our protocol, we implemented Tapestry unicast routing and the Bayeux tree protocol as a packet-level simulator. Our measurements focus on distance and bandwidth metrics, and do not model the effects of any cross traffic or router queuing delays.

We use the Stanford Graph Base library [30] to access four different topologies in our simulations (AS, MBone, GT-ITM and TIERS). The AS topology shows connectivity between Internet autonomous systems (AS), where each node in the graph represents an AS as measured by the National Laboratory for Applied Network Research [18] based on BGP routing tables. The MBone graph presents the topology of the MBone as collected by the SCAN project at USC/ISI [28] on February 1999. To measure our metrics on larger networks, we turned to the GT-ITM [12] package, which produces transit-stub style topologies, and the TIERS [34] package, which constructs topologies by categorizing routers into LAN, MAN, and WAN routers. In our experiments, unicast distances are measured as the shortest path distance between any two multicast members.

5.2 Performance Metrics

We adopt the two metrics proposed in [6] 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.

5.3 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. RDP is measured for all pairwise connections between nodes in the network. Figure 3 plots the cumulative distribution of RDP on this network. The horizontal

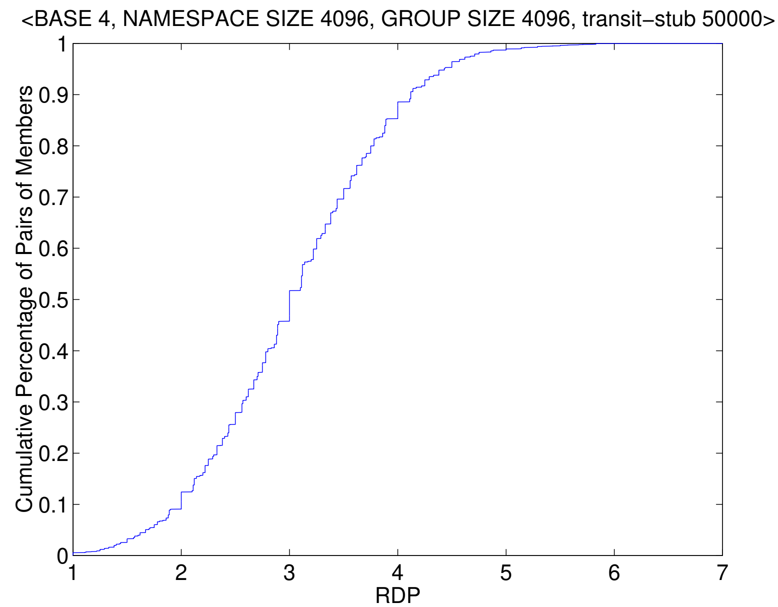


Figure 3: Cumulative distribution of RDP

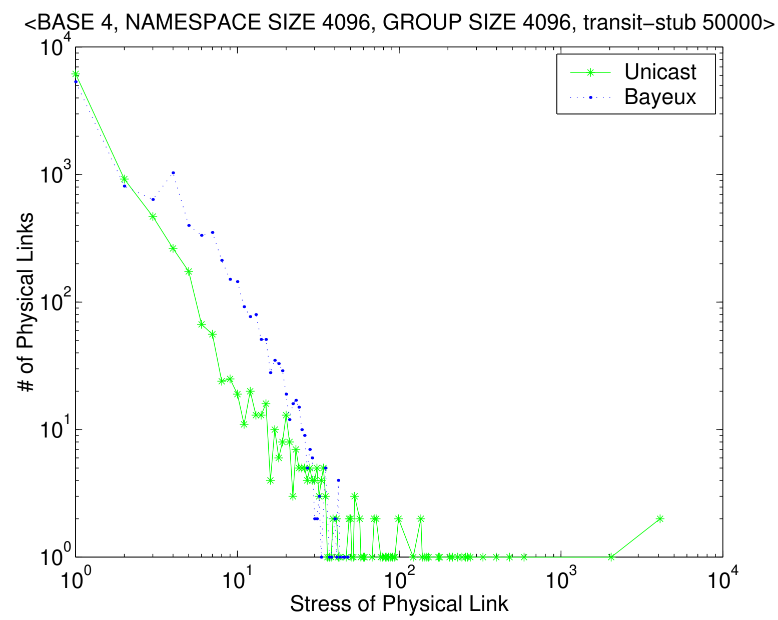


Figure 4: Comparing number of stressed links between naive unicast and Bayeux using Log scale on both axis.

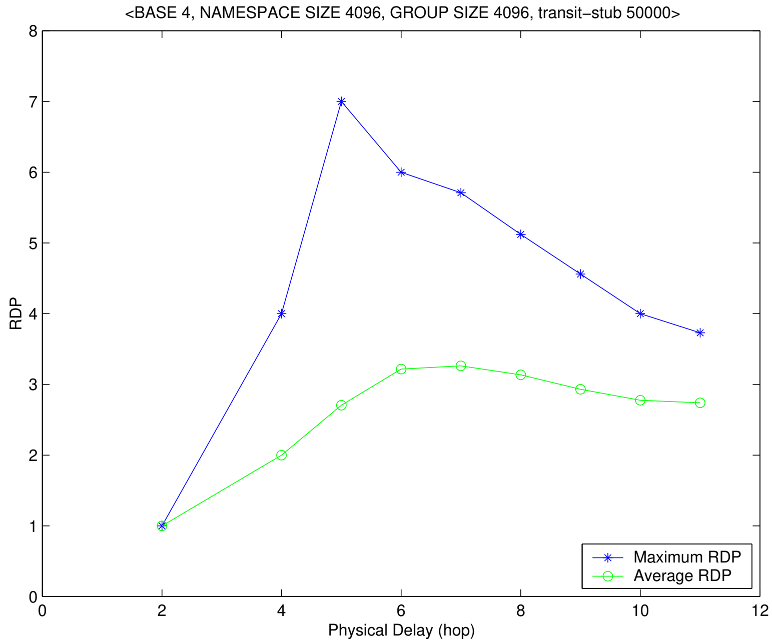


Figure 5: RDP vs. physical delay

axis represents a particular RDP and the vertical axis represents the cumulative fraction of sender-receiver pairs for which the RDP is less than this value. As we can see, the RDP for a large majority of connections is quite low. In fact, about 90% of pairs of members have a RDP less than 4.

A few sender-receiver pairs have a higher RDP, however, it can be seen in Figure 5 that the maximum RDP of seven corresponds to a sender-receiver pair with a small physical delay of five hops. This is because even though two nodes are physically close to each other, the digit-by-digit nature of Tapestry routing still produces a path of the same number of overlay hops, which can result in higher RDPs. However, the overlay delay between this sender-receiver pair is not very high, which can be seen from Figure 6.

In Figure 4, 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 [6] also exhibits low physical link stress, it only scales to receiver groups of hundreds.

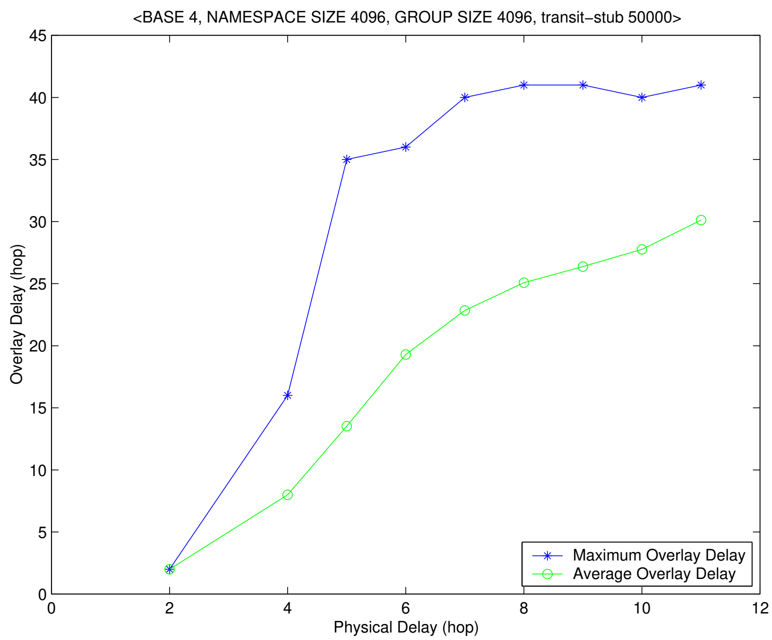


Figure 6: Overlay delay vs. physical delay

5.4 Effects of Tunable Parameters on Performance

In this section, we study the effects of varying parameters multicast group size, namespace size, topology size, and base on the performance of Bayeux. The namespace of Tapestry nodes is defined by fixed-length bit sequences represented by a common base. For instance, a Tapestry network can support 4096 nodes using 12 bit identifiers represented as 3 hexadecimal digits. For all results in the following sections, each data point is obtained by conducting 10 independent simulation experiments, and we plot the mean and the standard deviation.

5.4.1 Group Size

In this experiment, we use topologies from the AS, MBone, TIERS, and transit-stub models, a Tapestry namespace size of 4096, and a base of 4. Figure 7 plots the 90th percentile RDP versus increasing group size for these four topologies. All the curves are close to each other except the AS topology, which shows slightly higher RDPs. This is because the connectivity of a topology directly affects the properties of the Tapestry overlay network built on top of it. Consider the difference between AS and MBone topologies. The MBone is composed of islands that can directly support IP multicast, where the islands are linked by virtual point-to-point tunnels whose endpoints have support for IP multicast. The MBone topology is a combination of mesh at the backbone and star at each regional network, however, the connectivity in the mesh is manually configured and ad-hoc. In contrast, the AS topology is more structured and much better connected with increasing amount of peering relationships in the recent years. Therefore, more nodes have higher fanouts in the AS topology, which means that there are plenty of freedom in choosing the optimal route in shortest

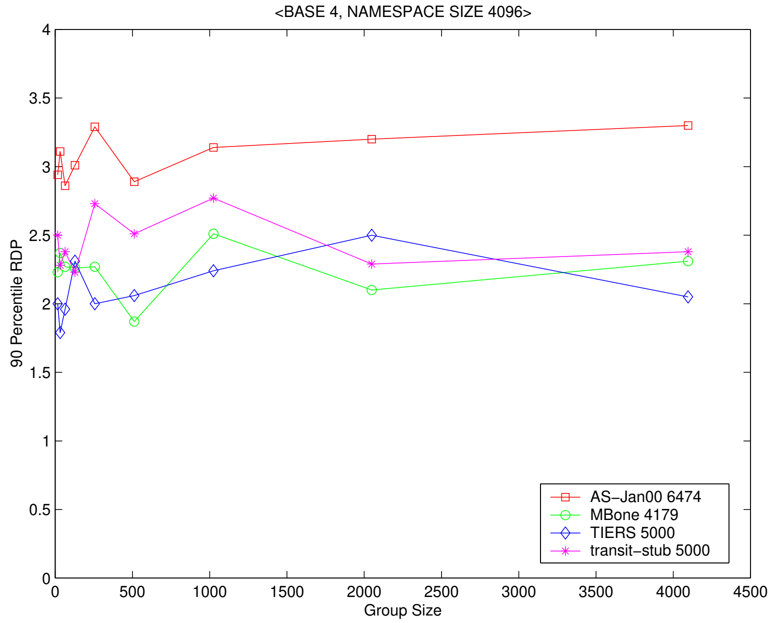


Figure 7: 90 percentile RDP vs. group size for topologies from four models

path unicast routing. However, unicast routing in Tapestry is somewhat contrained in the sense that routes have to follow the destination node identifiers, and thus cannot fully leverage the choice of routes offered by the underlying topology, which offers some intuition why the AS topology tend to have higher RDPs. Now we look at the overall variation in RDP as the group size increases from 16 to 4096. Figure 7 shows that the 90 percentile RDP remained more or less constant, which is expected because increasing the group size only increases the fanouts of branching points, but does not increase the height of the Bayeux tree, thus not affecting the RDP.

Next we study the effect of varying group size on worst case physical link stress. We only consider the generated transit-stub model of 50000 nodes because the results are skewed in other real topologies of about 5000 nodes since the multicast session density becomes too high for a group size of 4096. Figure 8 plots the variation of the worst case physical link stress for the transit-stub model. The worst case physical link stress increases sub-linearly as the group size increases from 16 to 4096. While for large group sizes of thousands, worst case stress may be higher, it is still much lower than naive unicast.

5.4.2 Namespace Size

In this experiment, we examine the effect of varying the size of the Tapestry network on the RDP and the worst case physical link stress. We use the topologies from the AS, MBone, TIERS, and transit-stub models, and a Tapestry base of 4. Because we are only interested in the variation of performance with respect to namespace size, we use a multicast group of 64 members to decrease the amount of simulation time. Figure 9 and 10 plot the variations on RDP and worst case stress as the Tapestry namespace size increases from 64 to 4096. For all topologies, we see a slight

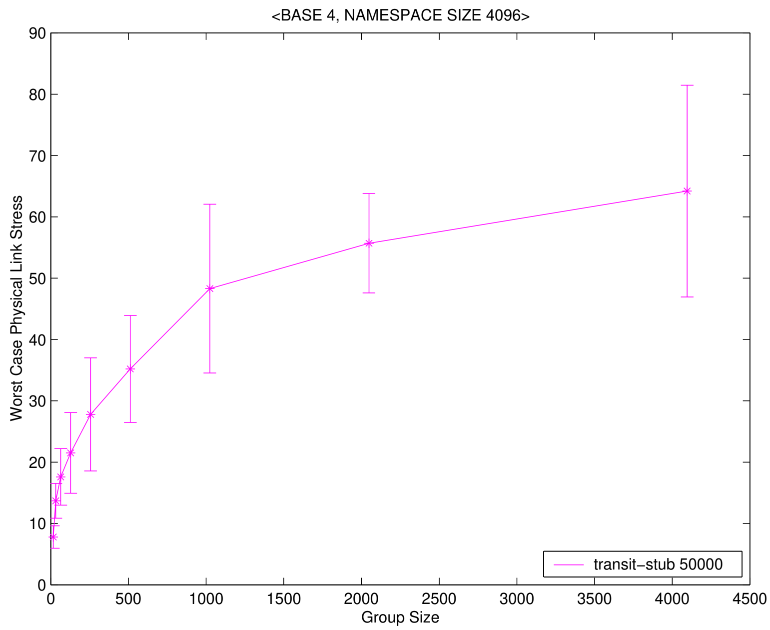


Figure 8: Worst case physical link stress vs. group size for transit-stub 50000

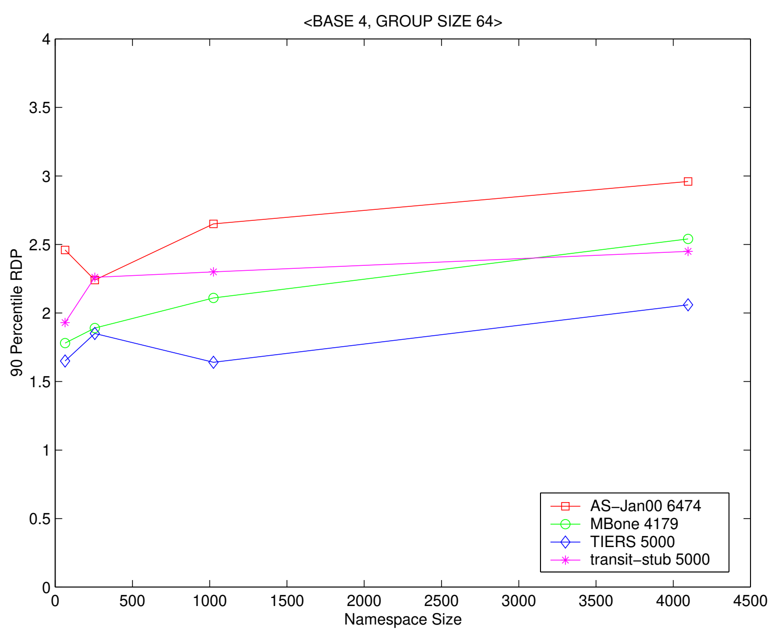


Figure 9: 90 percentile RDP vs. Tapestry network size for topologies

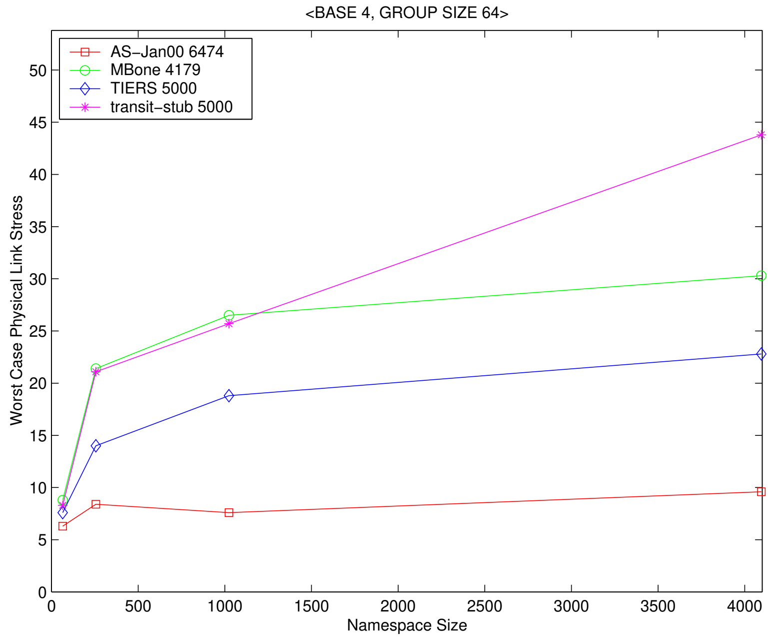


Figure 10: Worst case physical link stress vs. Tapestry network size for topologies

increase in the RDP and worst case stress. This is because we do not gain additional benefits by adding more Tapestry nodes beyond the number of members in the multicast group. In fact, the performance degrades because as the namespace size increases and the base is kept constant, a node needs to traverse a longer overlay path in order to reach another node, which increases the end-to-end latencies, and also causes unnecessary packet duplications. With respect to varying topologies, we note from Figure 10 that the AS topology exhibits the lowest worst case stress. This is due to the same reasons as why the AS topology has a higher RDP than the other topologies. In other words, because of the higher fanout of nodes in the AS topology, more links share the responsibility of multicast forwarding such that the amount of load on each individual link becomes lower, attaining a load balancing effect.

5.4.3 Topology Size

In this section, we use a Tapestry namespace size of 64, a base of 4, and a multicast group size of 64. We generate topologies from the transit-stub model of sizes varying from 100 nodes to 50000 nodes, and evaluate the impact on Bayeux's performance. Figure 11 plots the RDP against the topology size, and Figure 12 plots the worst case stress against the topology size. We observe that both the RDP and worst case stress decrease in general as the topology size increases. This is because in a fixed physical space, the number of links increases as the topology becomes larger, which results in better routes becoming available.

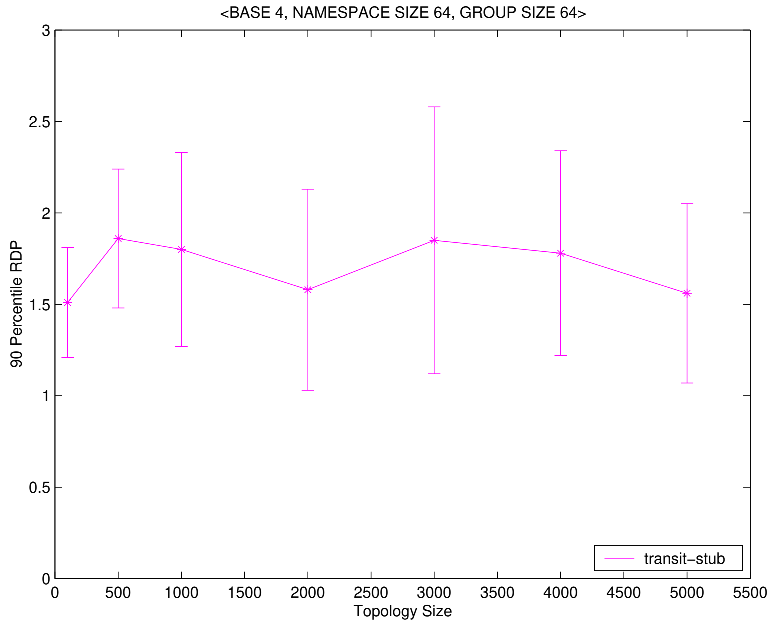


Figure 11: 90 percentile RDP vs. topology size for topologies from the transit-stub model

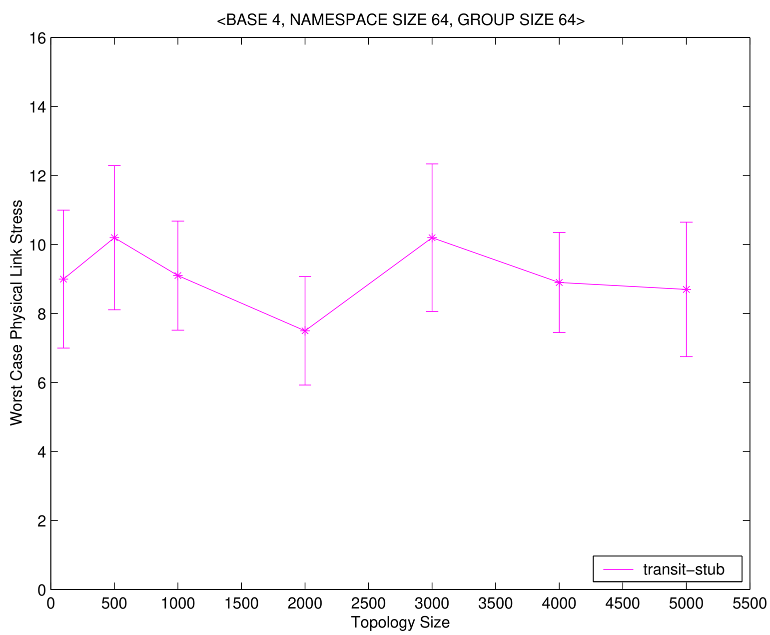


Figure 12: Worst case physical link stress vs. topology size for topologies from the transit-stub model

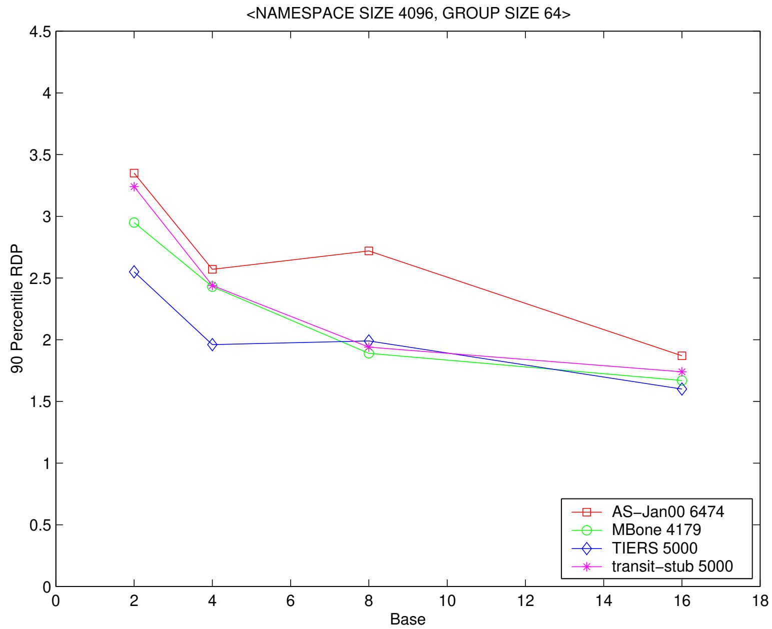


Figure 13: 90 percentile RDP vs. base for topologies from the four models

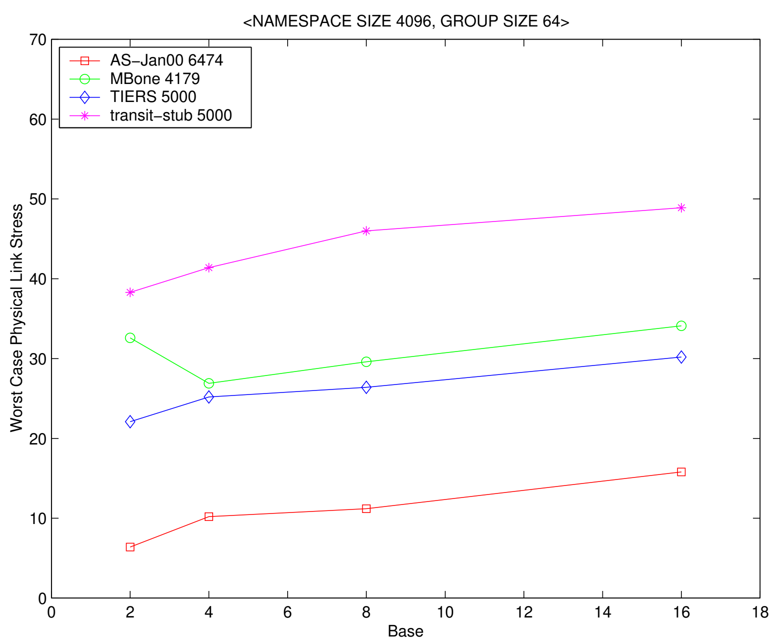


Figure 14: Worst case physical link stress vs. base for topologies from the four models

5.4.4 Base

Finally, we study the effect of variation of the Tapestry base, which determines the range of the overlay fanout in the Bayeux tree. We consider topologies from the AS, MBone, TIERS, and transit-stub models, a Tapestry namespace size of 4096, and a group size of 64. Figure 13 and 14 plot the variations in RDP and worst case stress as Tapestry base increases. When the base increases and the namespace is kept constant, the Bayeux tree height decreases, which causes RDP to decrease. On the other hand, the overlay fanout increases as the base increases, which causes physical link stress to increase because physical links near the branching nodes need to be shared by an increasing number of overlay links.

5.5 Summary of Results

In this section, we summarize the evaluation results that we have presented in earlier sections.

Across a range of topology models, Bayeux achieves a low RDP for a wide range of group sizes. Figure 7 shows that the 90 percentile RDP remained more or less constant as the group size increases from 16 to 4096.

In addition, Bayeux results in a low worst case stress for a wide range of group sizes. Figure 8 shows that the worst case stress increases sub-linearly as the group size increases from 16 to 4096 for the transit-stub model of 50000 nodes. While for larger group sizes, worst case stress may be higher, it is still much lower than unicast. For example, for a group of 4096 members, Bayeux reduces worst case stress by a factor of 85 compared to unicast.

6 Scalability Enhancements

In this section, we demonstrate and evaluate optimizations in Bayeux for load-balancing and increased efficiency in bandwidth usage. These enhancements, *Tree Partitioning* and *Receiver Clustering*, leverage Tapestry-specific properties, and are unique to Bayeux.

6.1 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, the non-symmetric routing in Bayeux implies that the root node must handle all `join` and `leave` requests from session members. Second, only the session root node can send data in a source-specific service model. Although the root can act as a reflector for supporting multiple senders [13], all messages have to go through the root, and a network partition or root node failure will compromise the entire group's ability to receive data.

To remove the root as a scalability bottleneck and point of failure, Bayeux includes a *Tree Partitioning* mechanism that leverages the Tapestry location mechanism. The idea is to create multiple

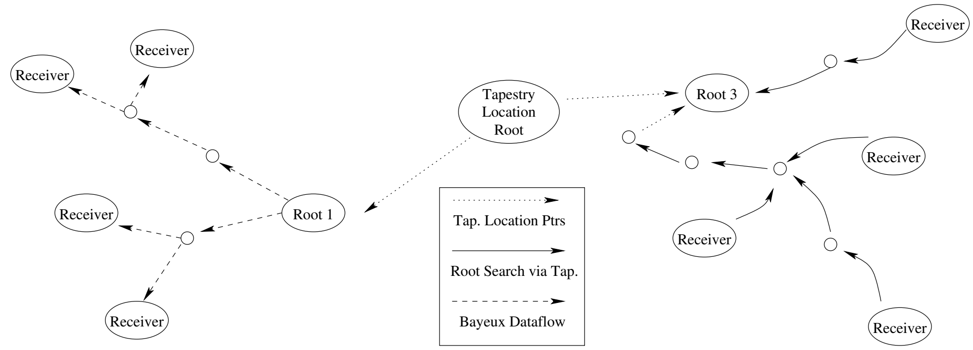


Figure 15: Receivers self-configuring into Tree Partitions

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 advertises 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 services 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 15 shows the path of various messages in the tree partitioning algorithm. Each member M sends location requests up to the Tapestry location root. Tapestry location services guarantee M will find the closest such root with high probability [21, 36]. 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 efficient tool for balancing load between root nodes and reducing first hop latency to receivers when roots are placed near listeners. Bayeux's technique of root replication is similar in principle to root replication used by many existing IP multicast protocols such as CBT [3] and PIM [7, 8]. Unlike other root replication mechanisms, however, we do not send periodic advertisements via the set of root nodes, and members can transparently find the closest root given the root node identifier.

We performed evaluation of our root replication algorithms by simulation. Our simulation results on four topologies (AS, MBone, Transit-stub and TIERS) are quite similar. Here we only show the Transit-stub results for clarity. We simulate a large multicast group that self-organizes into membership partitions, and examine how replicated roots impact load balancing of membership operations such as `join`. Figure 16 plots the mean and the 5th and 95th percentiles of the number of `join` requests handled per root as members organize themselves around more replicated roots.

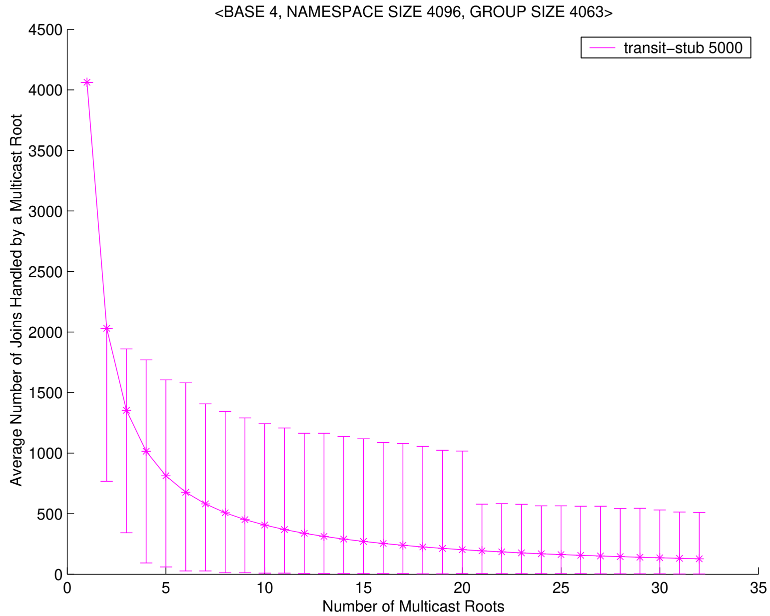


Figure 16: Membership Message Load Balancing by Roots

While the mean number of requests is deterministic, it is the 5th and 95th percentiles which show how evenly `join` requests are load-balanced between different replicated roots. As the number of roots increases, the variation of the number of `join` requests handled among the roots decreases inversely, showing that load-balancing does occur, even with randomly distributed roots, as in our simulation. One can argue that real-life network administrators can do much better by intelligently placing replicated roots to evenly distribute the load.

6.2 Receiver Identifier Clustering

To further reduce packet duplication, Bayeux introduces the notion of receiver node ID clustering. Tapestry delivery of Bayeux packets approaches the destination ID digit by digit, and one 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. Multiples of these 16-node groups can be further organized into larger groups, constructing a clustered hierarchy. Figure 17 shows such an example. While group sizes matching the Tapestry ID 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.

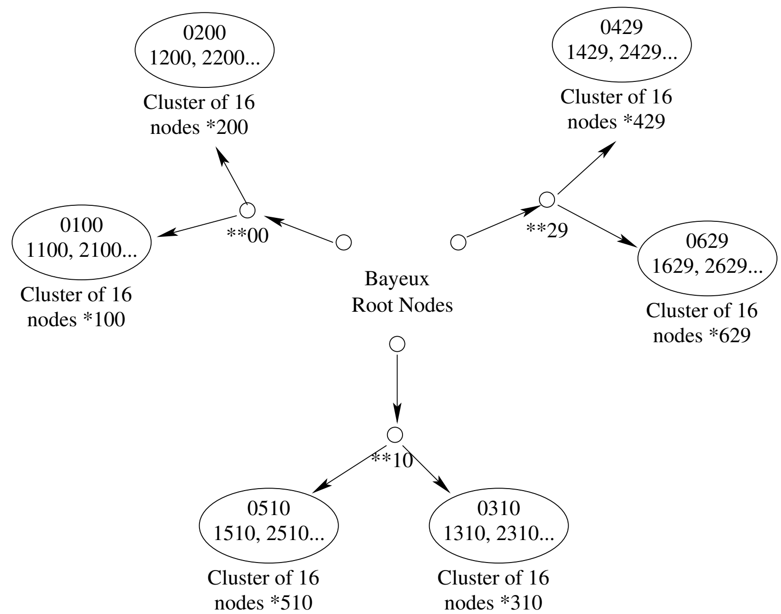


Figure 17: Receiver ID Clustering according to network distance

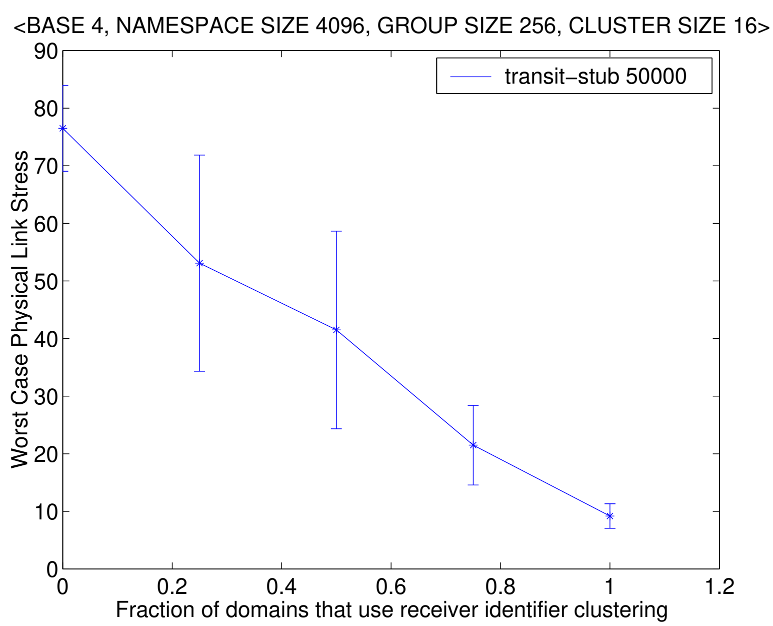


Figure 18: Worst case physical link stress vs. fraction of domains that use receiver ID clustering for the transit-stub model

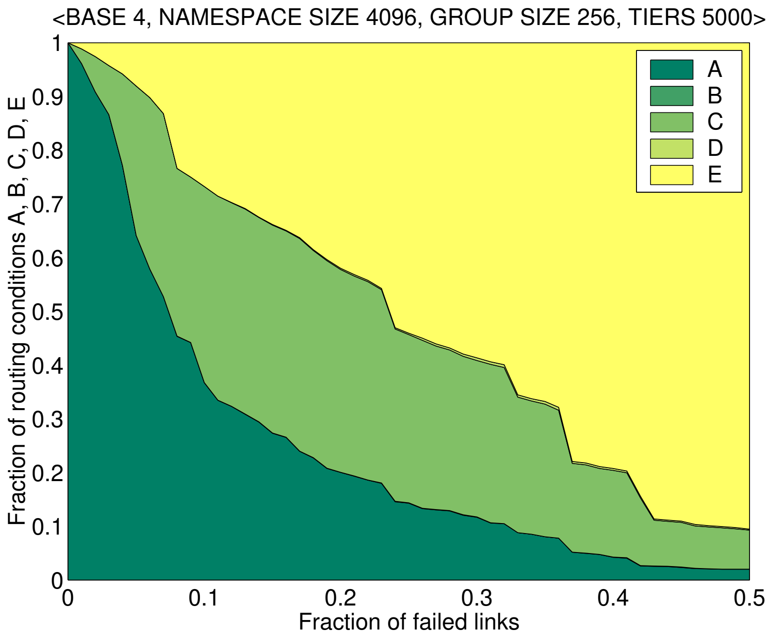


Figure 19: Maximum Reachability via Multiple Paths vs. Fraction of Failed Links in Physical Network

To quantify the effect of clustered naming, we measured link stress versus the fraction of local LANs that utilize clustered naming. We simulated 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 18 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 case link stress and local clustering.

7 Fault-resilient Packet Delivery

In this section, we examine how Bayeux leverages Tapestry's routing redundancy to maintain reliable delivery despite node and link failures. Each entry in the Tapestry neighbor map maintains secondary neighbors in addition to the closest primary neighbor. In Bayeux, membership state is kept consistent across Tapestry nodes in the primary path from the session root to all receivers. Routers on potential backup routes branching off the primary path do not keep member state. When a backup route is taken, the node where the branching occurs is responsible for forwarding on the necessary member state to ensure packet delivery.

We explore in this section approaches to exploit Tapestry's redundant routing paths for efficient fault-resilient packet delivery, while minimizing the propagation of membership state among

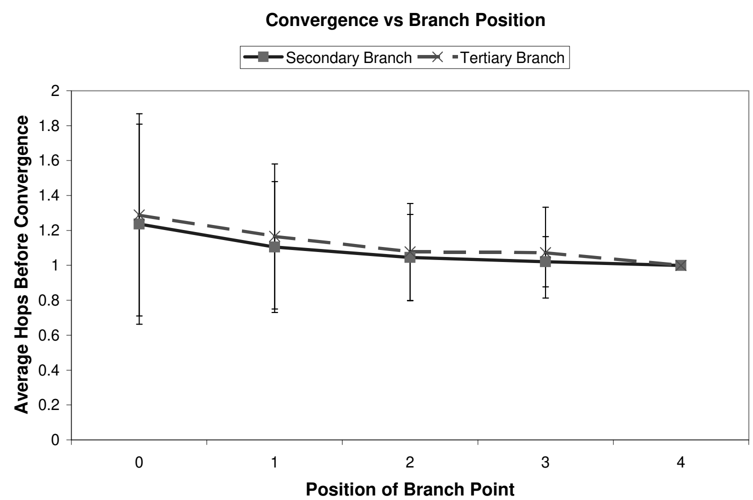


Figure 20: Average Hops Before Convergence vs. Position of Branch Point

Tapestry nodes. We first examine fault-resilient properties of the Tapestry hierarchical and redundant routing paths, then present several possible protocols and present some simulation results.

7.1 Infrastructure Properties

A key feature of the Tapestry infrastructure is its backup routers per path at every routing hop. Before examining specific protocols, we evaluate the maximum benefit such a routing structure can provide. To this end, we used simulation to measure maximum connectivity based on Tapestry multi-path routes. At each router, every outgoing logical hop maintains two backup pointers in addition to the primary route.

Figure 19 shows maximum connectivity 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. Note that regions B and D are almost invisible, since the multiple paths mechanism in Tapestry finds a route to the destination with extremely high probability, if such a route exists. This result shows that by using two backup pointers for each routing map entry, Tapestry achieves near-optimal maximum connectivity.

Another notable property of the Tapestry routing infrastructure is its hierarchical nature [36]. All possible routes to a destination can be characterized as paths up to a tree rooted at the destination. With a random distribution of names, each additional hop decreases the expected number of next hop candidates by a factor equal to the base of the Tapestry identifier. This property means that with evenly distributed names, paths from different nodes to the same destination converge within an expected number of hops equal to $Log_b(D)$, where b is the Tapestry digit base, and D is number of nodes between the two origin nodes in the network.

This convergent nature allows us to intentionally fork off duplicate packets onto alternate paths. Recall that the alternate paths from a node are sorted in order of network proximity to it. The expectation is that a primary next hop and a secondary next hop will not be too distant in the network. Because the number of routers sharing the required suffix decreases quickly with each additional hop, alternate paths are expected to quickly converge with the primary path. We confirm this hypothesis via simulation in Figure 20. On a transit-stub topology of 5000 nodes, Tapestry IDs with base 4, where the point to point route has 6 logical hops, we see that convergence occurs very quickly. As expected, an earlier branch point may incur more hops to convergence, and a secondary route converges faster than a tertiary route.

7.2 Fault-resilient Delivery Protocols

We now examine more closely a set of Bayeux packet delivery protocols that leverages the redundant route paths and hierarchical path reconvergence of Tapestry. While we list several protocols, we only present simulation results for one, and continue to work on simulation and analysis of the others. The protocols are presented in random order as follows:

1. *Proactive Duplication:* Each node forwarding data sends a duplicate of every packet to its first backup route. Duplicate packets are marked, and routers on the secondary path cannot duplicate them, and must forward them using their primary routers at each hop.

   The hypothesis is that duplicates will all converge at the next hop, and duplication at each hop means any single failure can be circumvented. While incurring a higher overhead, this protocol also simplifies membership state propagation by limiting traffic to the primary paths and first order secondary nodes. Membership state can be sent to these nodes before the session. This protocol trades off additional bandwidth usage for circumventing single logical hop failures.

2. *Application-specific Duplicates:* Similar to previous work leveraging application-specific data distilling [20], this protocol is an enhancement to *Proactive Duplication*, where an application-specific lossy duplicate is sent to the alternate link. In streaming multimedia, the duplicate would be a reduction in quality in exchange for smaller packet size. This provides the same single-failure resilience as protocol 1, with lower bandwidth overhead traded off for quality degradation following packet loss on the primary path.

3. *Prediction-based Selective Duplication:* This protocol calls for nodes to exchange periodic UDP probes with their next hop routers. Based on a moving history window of probe arrival success rates and delay, a probability of successful delivery is assigned to each outgoing link, and a consequent probability calculated for whether a packet should be sent via each link. The weighted expected number of outgoing packets per hop can be varied to control the use of redundancy (e.g., between 1 and 2).

When backup routes are taken, a copy of the membership state for the next hop is sent along with the data once. This protocol incurs the overhead of periodic probe packets in exchange for the ability to adapt quickly to transient congestion and failures at every hop.

4. *Explicit Knowledge Path Selection:* This protocol calls for periodic updates to each node from its next hop routers on information such as router load/congestion levels and instantaneous link bandwidth utilization. Various heuristics can be employed to determine a probability function which choose the best outgoing path for each packet. Packets are not duplicated.

5. *First Reachable Link Selection:* This protocol is a relatively simple way to utilize Tapestry's routing redundancy. Like the previous protocol, a node receives periodic UDP packets from its next hop routers. Based on their actual and expected arrival times, the node can construct a brief history window to predict short-term reliability on each outgoing route. Each incoming data packet is sent on the shortest outgoing link that shows packet delivery success rate (determined by the history window) above a threshold. No packet duplication takes place. When a packet chooses an alternate route, membership state is sent along with the data. This protocol is discussed more in Section 7.3.

Note that several of these protocols (1, 2, 3) may send additional packets down secondary or tertiary routes in addition to the original data. As we have shown in Figure 20, the bandwidth overhead of those protocols is limited, since the duplicates quickly converge back on to the primary path, and can be suppressed. This gives us the ability to route around single node or link failures. Duplicate packet supression can be done by identifying each packet with a sequential ID, and keeping track of the packets expected but not received (in the form of a moving window) at each router. Once either the original or the duplicate packet arrives, it is marked in the window, and the window boundary moves if appropriate. All packets that have already been received are dropped.

7.3 First Reachable Link Selection

Each of the above protocols has advantages and disadvantages, making them best suited for a variety of different operating conditions. We present here our evaluation of First Reachable Link Selection (FRLS), by first examining its probability of successful packet delivery, and then simulating the increasing latency associated with sending membership state along with the data payload.

Figure 21 shows that FRLS delivers packets with very high success rate despite link failures. The regions are marked similarly to that of Figure 19, where region A represents successful routing by IP and Tapestry, region B is where IP succeeds and Tapestry fails, region C is where IP fails and Tapestry succeeds, region D is where a possible route exists but neither IP nor Tapestry find it, and region E is where no path exists to the destination. When compared to Figure 19, we see that by choosing a simple algorithm of taking the shortest predicted-success link, we gain almost all of the potential fault-resiliency of the Tapestry multiple path routing. The end result is that FRLS delivers packets with high reliability in the face of link failures.

FRLS delivers packets with high reliability without packet duplication. The overhead comes in the form of bandwidth used to pass along membership state to a session's backup routers. FRLS keeps the membership state in each router on the primary path that the packets traverse. The size

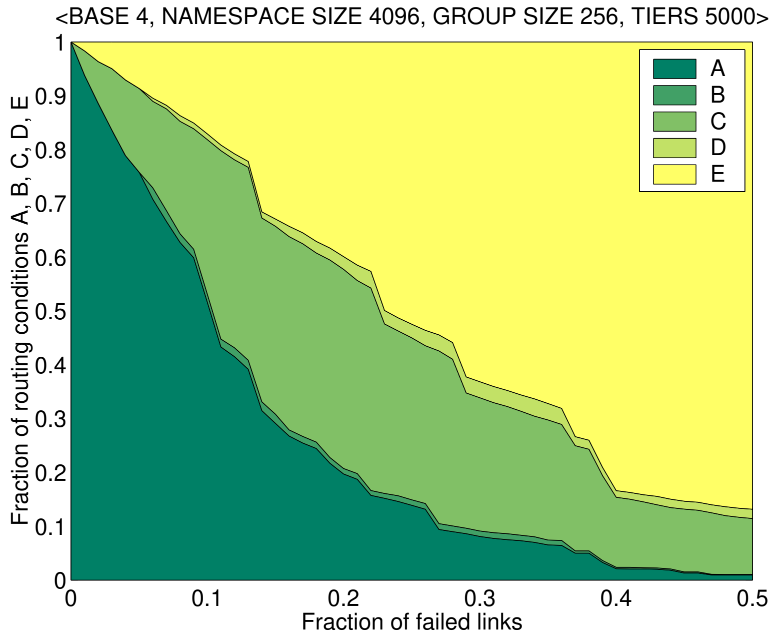


Figure 21: Fault-resilient Packet Delivery using First Reachable Link Selection

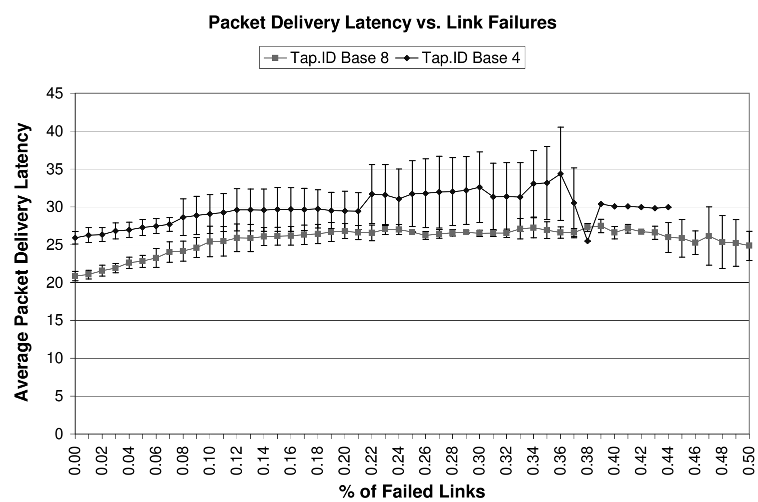


Figure 22: Bandwidth Delay Due to Member State Exchange in FRLS

of membership state transmitted decreases for routers that are further away from the data source (multicast root). For example, a router with ID "`475129`" that is two hops away from the root keeps a list of all members with Tapestry IDs ending in `29`, while another router `420629` two hops down the multicast tree will keep a list of all members with IDs ending in `0629`. When a backup route is taken and routing branches from the primary path, the router at the branching point forwards the relevant portion of its own state to the branch taken, and forwards it along with the data payload. This causes a delay for the multicast data directly proportional to the size of member state transmitted.

We plot a simulation of average delivery latency in FRLS, including the member state transmission delay, on a transit-stub 5000 node topology, using both base 4 and base 8 for Tapestry IDs. Note that average time to delivery does not include unreachable nodes as failure rate increases. Figure 22 shows that as link failures increase, delivery is delayed, but not dramatically. The standard deviation is highest when link failures have forced half of the paths to resort to backup links, and it spikes again as the number of reachable receivers drops and reduces the number of measured data points.

8 Related Work

There are several projects that share the goal of providing the benefits of IP multicast without requiring direct router support ([5, 6, 10, 14, 19, 23, 27]). End System Multicast [6] is one such example targeted towards small-sized groups such as audio and video conferencing applications, where every member in the group is a potential source of data. However, it does not scale to large-sized multicast groups because every member needs to maintain a complete list of every other member in the group. The Scattercast work by Chawathe et al. [5] is similar to the End System Multicast approach except in the explicit use of infrastructure service agents, SCXs. Both Scattercast and End System Multicast build a mesh structure across participating nodes, and then construct source-rooted trees by running a standard routing protocol. On the other hand, Yallcast [10] directly builds a spanning tree structure across the end hosts without any intermediate mesh structure, which requires expensive loop detection mechanisms, and is also extremely vulnerable to partitions. The CAN multicast work by Ratnasamy et al. [23] and the SCRIBE work by Rowstron et al. [27] are similar to Bayeux in that they achieve scalability by leveraging the scalable routing infrastructure provided by systems like CAN [22], Pastry [26], and Tapestry respectively. However, these systems have not focused on fault-tolerant packet delivery as a primary goal.

In terms of the service model, EXPRESS [13] also adopts a source-specific paradigm, and augments the multicast class D address with a unicast address of either the core or the sender. This eliminates the address allocation problem and provides support for sender access control. In contrast, Bayeux goes one step further and eliminates the class D address altogether. Using only the UID and session name to identify the group makes it possible to provide additional features, such as native incremental deployability, and load balancing at the root.

The idea of root replication shows a promising approach of providing anycast service at the application level. Recently, IP-anycast has been proposed as an infrastructure service for multicast routing. For example, Kim et al. use anycast to allow PIM-SM to support multiple rendezvous points per multicast tree [15]. However, there is a lack of a globally deployed IP-anycast service. There are several proposals for providing an anycast service at the application layer ([4, 9, 11, 17, 29]), which attempt to build directory systems that return the nearest server when queried with a service name and a client address. Although our anycast service is provided at the application layer, server availability is discovered by local Tapestry nodes and updated naturally as a part of the Tapestry routing protocol. Therefore, our mechanism may potentially provide an anycast service that is easier to deploy than IP-anycast, yet avoids several complications and scalability problems associated with directory-based application layer anycast. We believe that the application layer anycast provided by the Tapestry overlay network described herein forms an interesting topic for future research.

Finally, there are several recent projects focusing on similar goals as Tapestry. Among them are Chord [32] from MIT/Berkeley, Content-Addressable Networks (CAN) [22] from AT&T/ACIRI and Pastry [26] from Rice and Microsoft Research. These research projects have also produced decentralized wide-area location and routing services with fault-tolerant properties, but only Tapestry provides explicit correlation between overlay distance and underlying network distance.

9 Future Work

In this report, we have studied the properties of the First Reachable Link Selection (FRLS) protocol, it will be worthwhile to explore and understand the performance and tradeoffs involved in the alternative fault-resilient delivery protocols discussed in Section 7. In particular, it will be useful to look at the effect of different parameters on each protocol, and their performance under varying operating conditions.

The Streaming Media Systems Group at HP Labs has developed a multiple state video encoder/deoder and a path diversity transmission system [1, 2], which sends different subsets of packets over different paths. The multiple state video codec seem to fit well with our packet duplication techniques onto alternate paths, and is an interesting area for future research.

Finally, it will be worthwhile to conduct large scale Internet experiments with emphasis on studying the dynamics of Bayeux, and effects of packet loss and cross-traffic.

10 Conclusion

In conclusion, we have presented an architecture for Internet content distribution that leverages Tapestry, an existing fault-tolerant routing infrastructure. Simulation results show that Bayeux achieves scalability, efficiency, and highly fault-resilient packet delivery. We believe Bayeux shows that an efficient network protocol can be designed with simplicity while inheriting desirable properties from an underlying application infrastructure.

11 Acknowledgements

I would like to thank Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John D. Kubiatowicz, who were co-authors on the NOSSDAV 2001 paper on which this technical report extended upon. I would also like to thank Almudena Konrad, Lakshminarayanan Subramanian, Helen Wang, and Koichi Yano for their many helpful suggestions and discussions.

This work was supported in part by grants from Ericsson, Intel, Sprint, and Motorola, by DARPA under contract DABT63-98-C-0038, by the State of California under the MICRO program, and by NSF Research Infrastructure grant CDA-94-01156. Shelley Zhuang is supported by a Department of Defense NDSEG grant.

References

[1] APOSTOLOPOULOS, J. G. Reliable video communication over lossy packet networks using multiple state encoding and path diversity. In *Proceedings of Visual Communications and Image Processing* (2001), IEEE.

[2] APOSTOLOPOULOS, J. G., AND WEE, S. J. Unbalanced multiple description video communication using path diversity. In *Proceedings of International Conference on Image Processing* (October 2001), IEEE.

[3] BALLARDIE, A. Core based trees (CBT) multicast routing architecture. Internet Request for Comments RFC 2201, September 1997. `http://www.landfield.com/rfcs/rfc2201.html`.

[4] BHATTACHARJEE, S., AMMAR, M., ZEGURA, E., SHAH, N., AND FEI, Z. Application layer anycasting. In *Proceedings of IEEE INFOCOM* (June 1997).

[5] CHAWATHE, Y., MCCANNE, S., AND BREWER, E. A. An architecture for internet content distribution as an infrastructure service. `http://www.cs.berkeley.edu/~yatin`, 1999.

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

[7] ESTRIN, D., FARINACCI, D., HELMY, A., THALER, D., DEERING, S., HANDLEY, M., JACOBSON, V., LIU, C., SHARMA, P., AND WEI, L. Protocol independent multicast - sparse mode (pim-sm): Protocol specification. Internet Request for Comments RFC 2117, June 1997.

[8] ESTRIN, D., FARINACCI, D., JACOBSON, V., LIU, C., WEI, L., SHARMA, P., AND HELMY, A. Protocol independent multicast - dense mode (pim-dm): Protocol specification.

[9] FEI, Z., BHATTACHARJEE, S., AMMAR, M. H., AND ZEGURA, E. W. A novel server technique for improving the response time of a replicated service. In *Proceedings of IEEE INFOCOM* (June 1998).

[10] FRANCIS, P. Yallcast: Extending the internet multicast architecture, September 1999. `http://www.yallcast.com`.

[11] FRANCIS, P., JAMIN, S., PAXON, V., ZHANG, L., GRYNIEWICZ, D. F., AND JIN, Y. An architecture for a global host distance estimation service. In *Proceedings of IEEE INFOCOM* (June 1998).

[12] Georgia tech internet topology model. `http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html`.

[13] HOLBROOK, H. W., AND CHERITON, D. R. Ip multicast channels: EXPRESS support for large-scale single-source applications. In *Proceedings of SIGMETRICS* (August 1999).

[14] JANNOTTI, J., GIFFORD, D. K., JOHNSON, K. L., KAASHOEK, M. F., AND JAMES W. O'TOOLE, J. Overcast: Reliable multicasting with an overlay network. In *Proceedings of OSDI* (October 2000).

[15] KIM, D., MEYER, D., KILER, H., AND FARINACCI, D. Anycast RP mechanism using PIM and MSDP, 2000. Internet-Draft, `http://www.join.uni-muenster.de/drafts/draft-ietf-mboned-anycast-rp-05\%.txt`.

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

[17] MYERS, A., DINDA, P., AND ZHANG, H. Performance characteristics of mirror servers on the internet. In *Proceedings of IEEE INFOCOM* (June 1999).

[18] National laboratory for applied network research. `http://moat.nlanr.net/Routing/rawdata/`.

[19] PENDARAKIS, D., SHI, S., VERMA, D., AND WALDVOGEL, M. ALMI: An application level multicast infrastructure. In *Proceedings of USITS* (March 2001).

[20] PERKINS, C. S., HUDSON, O., AND HARDMAN, V. Network adaptive continuous-media applications through self-organised transcoding. In *Proceedings of Network and Operating Systems Support for Digital Audio and Video* (Cambridge, UK., July 1998), ACM.

[21] 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).

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

[23] RATNASAMY, S., HANDLEY, M., KARP, R., AND SCHENKER, S. Application-level multicast using content-addressable networks. In *Proceedings of NGC* (November 2001), ACM.

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

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

[26] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, decentralised object location and routing for large-scale peer-to-peer systems. Middleware, 2001.

[27] ROWSTRON, A., KERMARREC, A.-M., CASTRO, M., AND DRUSCHEL, P. Scribe: The design of a large-scale event notification infrastructure. In *Proceedings of NGC* (November 2001), ACM.

[28] The SCAN project. `http://www.isi.edu/scan/`.

[29] SESHAN, S., STEMM, M., AND KATZ, R. SPAND: Shared passive network performance discovery. In *Proceedings of USITS* (March 1997).

[30] The stanford graph base (SGB) package. `ftp://labrea.stanford.edu/pub/sgb/`.

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

[32] 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.

[33] STOICA, I., NG, T. S. E., AND ZHANG, H. REUNITE: A recursive unicast approach to multicast. In *Proceedings of INFOCOM* (March 2000).

[34] Tiers. `http://www.isi.edu/haldar/topogen/tiers1.0.tar.gz`.

[35] 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).

[36] 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, University of California at Berkeley, Computer Science Division, April 2001.

A Appendix

In this appendix, we discuss the setup of our simulator used to carry out the experimental analysis described earlier. We implemented Tapestry unicast routing and the Bayeux tree protocol by extending the Stanford Graph Base library (SGB) [36], which is a platform for combinatorial computing. The SGB library contains routines to manipulate graph structures, such as file formats, input/output functions and shortest path calculations. We describe the various components of the simulator in the following sections.

A.1 SGB Modification

gb_graph.w 9 extra vertex utility fields and 4 extra arc utility fields are added. This file need to be put into the SGB source code directory before installing SGB.

A.2 Generic Functions

cluster.c functions that implement the Receiver Clustering scalability enhancement discussed in Section 6.2.

fault.{c,h} functions that inject link and node failures into the underling physical network

graph.{c,h} functions that interacts with the SGB graph structures

hop.{c,h} functions that measure routing delays

max_conn.c functions that measure Maximum Reachability via Multiple Paths discussed in Section 7.1.

nodeid.{c,h} functions that implement various Tapestry node ID conversions

pick.{c,h} functions that pick vertices from the Tapestry network

protocol.c functions that implement the FRLS protocol discussed in Section 7.3

route.{c,h} functions that build the Tapestry routing table for every node

stack.{c,h} functions that implement the stack data structure

stat.{c,h} functions that implement various statistic routines

stress.{c,h} functions that measure stress on physical links

tree_partition.c functions that implement the Tree Partitioning scalability enhancement discussed in Section 6.1.

util.{c,h} functions that implement various utility routines

A.3 Experimental Main Loop

exp.{c,h} functions that implement the initialization and running of the experiments

main.c functions that setup and run the experiments

A.4 Post Processing

read.{c,h} functions that read delay and stress values for post processing

post_proc.c functions that post process experimental results