Tapestry: A Fault-tolerant Wide-area Application Infrastructure

Ben Y. Zhao, John D. Kubiatowicz, Anthony D. Joseph
Computer Science Division
University of California, Berkeley
{ravenben,kubitron,adj}@cs.berkeley.edu

This abstract describes progress of the Tapestry wide-area overlay network infrastructure, current experimental results on routing efficiency, locality and fault-tolerance, along with the design and implementation of a wide-area multicast system called Bayeux. Tapestry is an infrastructure approach to solving several key challenges in building novel large-scale network applications, including scalability, fault-tolerance, and adaptability in the wide-area.

Tapestry solves these issues as an overlay network application infrastructure. It provides network location services in the form of mapping globally unique object IDs to server locations, and message routing services given the unique ID of the destination node. Identifiers are randomly assigned 160 bit strings. Messages are routed using a hop-by-hop incremental matching of node IDs to the destination node. Objects are distributed across random embedded trees, and a function maps each object-ID to the node-ID of its "root node." Insertion involves using inter-node routing pointers to find the root, while placing pointers to the object on intervening hops. Queries also use routing pointers to find the destination root, but terminate when they intersect hops with the desired object pointers. For each possible suffix length, each node keeps routing links to its "closest" neighbors with the common suffix that differs in the next digit. This results in good locality and distribution, and routes with at most Log_b(# of Nodes) hops, where b is the base. This scheme simplifies corrupted pointer detection, scales via random object distribution, and has natural redundancy resulting in resilience against intermediate node failures and small network partitions.

Compared to similar overlay systems like Chord [3] and Content-addressable Networks [2], Tapestry performs similarly or better in the key metrics of routing table size and expected logical hops between endpoints, shown in Table 1. The key difference between Tapestry and CAN and Chord is that Tapestry builds in an explicit correlation between overlay topology distance and physical network latency. In both CAN and Chord, it is possible to take one overlay hop to a neighbor node, and traverse an unbounded number of hops in the physical network (both systems compensate by using heuristics). In contrast, Tapestry's insertion algorithms select for each overlay hop the nearest nodes in network distance which satisfy the criteria. As a result, overlay distances correspond to physical network distances, and local searches do not incur long network traversals. In addition, Tapestry's hierarchical cache of object pointers makes a probabilistic argument, that a client searching for local copies of an object finds the nearest copy, and traverses a distance linearly proportional to its distance to that copy.

In addition to providing efficient overlay routing and object location, Tapestry provides redundant mechanisms that leverage the increasing availability of resources such as computational power, magnetic storage and network bandwidth. In object location, redundancy occurs when incoming objects are hashed to produce multiple unique names, and references to the object are inserted using all names. Queries are issued on the same set of names to provide high availability even in the presence of network partitions. Redundancy in routing occurs in the form of multiple outgoing pointers per routing entry. Experiments show that using a simple first available pointer algorithm (FRLS), Tapestry delivers packets with near optimal success rate as link failures on the network increase. Furthermore, simulations show that packets which take a secondary route and diverge from the primary path quickly converge back to the primary path within 2 hops. This means that fault-tolerant routing protocols can proactively duplicate packets to be sent on multiple outgoing links, and with duplicate packet suppression, incur only a small overhead in bandwidth usage.

Metric	Tapestry	Chord	C.A.N.
Key Parameters	Base B	None	Dimension D
Logical Hops	$Log_b(N)$	$Log_2(N)$	$O(d * N^{1/d})$
Routing State	$b * Log_b(N)$	$Log_2(N)$	$O(d)$

Table 1: Key Metric Comparison: N = nodes in network

Finally, Tapestry is designed as a fault-tolerant routing and location application infrastructure. By building applications such as the Bayeux [4] wide-area multicast system, we have demonstrated how a Tapestry application can transparently inherent its properties of scalability and fault-tolerant packet delivery with minimal effort in application design. We are currently in the process of deploying a dynamic Tapestry network for use within the OceanStore [1] global-scale storage system. For more information, please refer to the Tapestry documents available online at: `http://www.cs.berkeley.edu/~ravenben/tapestry`.

References

[1] KUBIATOWICZ, J., BINDEL, D., CHEN, Y., EATON, P., GEELS, D., GUMMADI, R., RHEA, S., WEATHERSPOON, H., WEIMER, W., WELLS, C., AND ZHAO, B. OceanStore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS* (November 2000), ACM.

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

[3] 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.

[4] ZHUANG, S. Q., ZHAO, B. Y., JOSEPH, A. D., KATZ, R. H., AND KUBIATOWICZ, J. D. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of the 11th International Workshop on Network and Operating System Support for Digital Audio and Video* (June 2001), ACM.