Tapestry

I. INTRODUCTION

Tapestry is a peer-to-peer(P2P) overlay network designed to simplify deployment of new distributed applications. Tapestry provides decentralized object location and routing (DOLR) interface that focuses on routing messages to endpoints. It virtualizes resources since endpoints are named by identifiers that do not share anything about their physical location. This virtualization enables message delivery to endpoints in the presence of instability in the underlying infrastructure. DOLR therefore, makes it easy for developers to ignore the dynamics of the underlying network and simply focus on optimization of their distributed applications when deploying them. Tapestry also provides location-independent routing of messages to "close" endpoints using only localized resources. Tapestry also uses adaptive algorithms with soft state to maintain fault tolerance in the face of changing node membership and network faults.

II. RELATED WORK

The first generation of P2P systems included: Napster *which used centralized server to locate files*; Gnutella *which provided distributed service using scoped broadcast queries limiting scalability*; MojoNation *used online economic model to encourage resource sharing*; Freenet *is a file-sharing network designed to resist censorship*.

The second generation of P2P systems are structured P2P overlay networks that included: Tapestry, Chord, Pastry and CAN. They all implement key-based routing (KBR) that supports deterministic routing of messages. They also support DHTs and DOLR and are thus, highly scalable.

Pastry and Tapestry construct locally optimal routing tables when initialized and maintain them throughout, as opposed to Chord and CAN, which do not take network distances into account when constructing routing overlay and may end up having an overlay hop that spans the diameter of the network.

Pastry and Tapestry resemble Plaxton et al. routing algorithm for a static network. Recent works include systems like Kademlia which uses XOR for overlay routing and Viceroy which provides logarithmic hops through nodes with constant degree routing tables. SkipNet uses a multidimensional skip-list data structure to support overlay routing while maintaining both a DNS-based namespace for operational locality and randomized namespace for network locality.

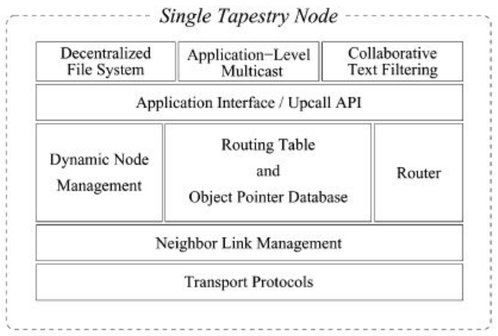
III. TAPESTRY ALGORITHMS

Tapestry nodes participate in overlay network and are assigned nodeIDs uniformly at random from an identifier space of 160-bit values with globally defined radix. Application specific end-points are assigned a globally unique identifiers (GUIDs) from the same space. Tapestry assumes these identifiers are evenly distributed in namespace which can be achieved by secure hashing algorithm. Efficiency of Tapestry improves with network size, therefore multiple applications to share a large Tapestry overlay network. Each message is assigned an Application specific identifier which is used to select a process/message delivery at the destination. Thus, there exists a four-part DOLR networking API: PUBLISHOBJECT (best effort attempt to make available object on the local node); UNPUBLISHOBJECT (best effort attempt to remove location mapping for given object); ROUTETOOBJECT (route messages to specified object location); ROUTETONODE (route message to application on node).

Tapestry dynamically maps each identifier to a unique live node called identifier's root. Each node maintains a routing tables of its neighbors called neighbor maps, thus, messages are forwarded across these links to nodes that are progressively closer in the ID space. The network environment is inherently dynamic and hence the challenge lies in being able to route reliably even when intermediate links re changing or faulty. Tapestry thus, exploits network path diversity in the form of redundant routing paths to provide resilience.

Tapestry uses some mechanisms to maintain routing table consistency and ensure object availability. When a new Node (N with N_{id}) insertion begins at its surrogate S (root node that maps N_{id} to the network) which then finds length of longest prefix its ID shares with N_{id} and sends out acknowledged multicast message to all nodes that share the same prefix, these nodes then add N to their routing tables. These nodes then contact N and become initial neighbor set used to construct its routing table. To all these nodes do not fail to notify each other about their existence, every node A in the multicast keeps state on every node B that is still multicasting down one of its neighbors. If a node N leaves, it tells the nodes in its backpointers about it along with a replacement node from its own routing table. The notified nodes send republish traffic to both N and its replacement. Tapestry also builds redundancy into routing tables and object location references to ensure resilience. Nodes also use periodic beacons to detect outgoing link and node failures triggering repair of routing mesh and initiate redistribution and replication of object location references.

IV. ARCHITECTURE


Single Tapestry Node
Decentralized File System
Application−Level Multicast
Collaborative Text Filtering
Application Interface / Upcall API
Dynamic Node Management
Routing Table and Object Pointer Database
Router
Neighbor Link Management
Transport Protocols

Transport layer provides abstraction of communication channels from one overlay node to other and use TCP/IP or UDP. Neighbor link layer provides secure but unreliable datagram facilities to layer above it including fragmentation and reassembly of large messages. Links are opened on demand by higher layers. It provides fault detection through soft-state keep alive messages, plus latency and loss rate estimation. This layer also optimizes message processing by parsing the message headers and only deserializing message content. Node authentication and MACs can be integrated into this layer for added security. Router layer contains the routing tables and local object pointers. The router examines destination GUID of message passed to it and determines the next hop using this table and local object pointers. Messages are then passed back to neighbor link layer for delivery. The interaction between Tapestry and application handles occur through three primary calls: Deliver (asynchronous; invoked on incoming messages) ; Forward (incoming upcall-enabled messages) ; and Route (invoked by application handler to forward a message on to NextHopNode).

V. CONCLUSION

An implementation of Tapestry using Java has been tested under a variety of conditions and has proven to be highly resilient under dynamic conditions. A variety of applications have been implemented and deployed on Tapestry infrastructure: OceanStore, a global-scale storage utility; Mnemosyne, a steganographic file system; Bayeux, an efficient self-organizing application multicast system; and SpamWatch, a decentralized spam-filtering system utilizing a similarity search engine implemented on Tapestry.