Paper Review 2

Name: Bharath Venkatesh
Unity ID: bharat@ncsu.edu

Pastry: Scalable, Decentralized Object Location and Routing for Large Scale Peer-to-Peer Systems

Summary

The core idea of this paper is to introduce a scalable approach for object location and routing of messages in peer-to-peer applications. It seeks to minimize the distance that the messages travel based on *network locality* metrics like number of IP routing hops or the geographical distance between the nodes. Each node in the pastry overlay is assigned a unique *128 bit nodeID* randomly such that they form a uniform distribution in the 128 bit nodeID space. Each node maintains a state consisting of a triplet of leaf set, routing table and neighborhood set*[L,R,M]*. Routing is done based on mapping of this local routing information with the message and key tuple that has arrived. Pastry accounts for replication by *mapping each key to k nodes* which are closest to the key, instead of mapping it to only one node. The packet is routed at the application level to the neighbor that has a *nodeID closest* to the key. For a network of N pastry peers the expected number of hops to route the packet is *O(log N)*.

Pros: Why the Paper Should be accepted

- The key take away from this algorithm is the fact that it *accounts for network locality* properties, thus minimizing the distance that messages travel.
- The algorithm is *self-organizing* allowing for automatic arrival and departure of nodes in the network.
- Pastry approach ensures that the *route obtained is optimal* in nature. Experimental results show that for a network size of 100,000 nodes the maximum route length is just 5 i.e O(log N).
- The algorithm is also proven to be *fault tolerant* accounting for recovery of all missing entries even under 10% node failure.
- The algorithm works in a completely *decentralized* manner and thus can serve as a generic substrate for several peer-to-peer internet applications.

Cons: Why the paper may be rejected

- Pastry considers *network proximity as the only metric* for measuring network performance. Several other characteristics like the load on a peer, its bandwidth etc are not taken into consideration. In case of large file transfers it more desirable to have a peer with higher bandwidth process a request rather than that with close network proximity.
- Each peer is expected to *keep track of its immediate neighbors* through periodic message exchange. This can lead to significant overhead on the network overlay.
- Another key drawback of the algorithm is that, it accounts for very *poor load balancing* when requests arise from a cluster of nearby nodes. It leads to all requests being handled by the same peer.

- Finally, the paper assumes that each file is necessarily replicated across several peers. Will non-replicating applications suffer? A discussion on those line would help.

Ideas for Future Research

- Considering *requests currently being handled by a peer* as an additional metric alongside network proximity can lead to a more balanced load on all replicating peers.
- An extension of this algorithm to *handle malicious and misbehaving nodes* in the network would make the it more robust to failures.

Chord: A Scalable Peer-to-Peer Lookup protocol for Internet Applications

Summary

The paper presents a protocol for efficient content lookup in peer-to-peer distributed hash table[DHT] implementations. It uses consistent hashing techniques in assigning keys to chord nodes. Further it accomplishes a mapping onto values by *storing each key/value pair* at node to which the key maps, and resolves lookups via messages to other nodes either iteratively or recursively. Chord provides a completely *decentralized* mechanism that allows a node to find the data item at *O(log N)* time by tracking only *O(log N)* locations. Aside from locating the keys, the protocol describes *stabilization* operation that allows for any node to join or leave the network at any time. The paper also provides simulation and experimentation results that evaluates the above mentioned claims.

Pros: Why the Paper Should be accepted

- *Simplicity, provable correctness and provable performance* are three prime features that distinguish Chord from several other peer-to-peer lookup protocols.
- The algorithm is *dynamic in nature* and adapts very well to nodes joining and leaving the system. It is able to answer queries, albeit with some delay, even when the system is continuously changing.
- Chord distributes keys equally across peers thus leading to *efficient load balancing.*
- Simulation results prove that the algorithm is *scalable* as the states that each node needs to maintain increases only logarithmically with the number of nodes [O(log N)].
- The protocol has *no centralized control* and thus suitable for distributed environments.

Cons: Why the paper may be rejected

- One of the key shortcoming of the Chord algorithm is that it supports only *exact key lookup*. In practical applications we only find keywords and not DHT keys. Thus we require a layer of abstraction above it to convert the input keywords to DHT keys.

- Although the protocol ensures that there is equal distribution of keys across all the nodes, it doesnt take into account for the *size of the data items*. Furthermore, there might be some files which are very frequently downloaded and some that are rarely shared. An extension that supports additional constraints to handle these issues would be more useful in practical applications.
- With the proposed system, it is possible that a particular data item which is geographically close to the user might be assigned to a node far from the user hence *increasing delays*. This arises from the fact that chord doesnt take into account the *network locality.*
- The application using chord is responsible for providing any desired *authentication, caching and replication* which might not be desirable for all applications.

Ideas for Future Research

- Chord doesnt have any specific mechanism to *heal partitioned rings*. An approach to detect and correct such partitions would make the protocol more robust.
- Considering *security constraints*, the algorithm can be further extended to provide a layer of cryptographic authentication thus detecting malicious users.
- Considering the frequency of download and size of the file as additional metrics in determining key distribution can help in better load balancing.
- Partial Key Lookup and Wildcard matching techniques can be deployed to ease the constraint of exact key matching.