Achieving One-Hop DHT Lookup and Strong Stabilization by Passing Tokens

Ben Leong and Ji Li

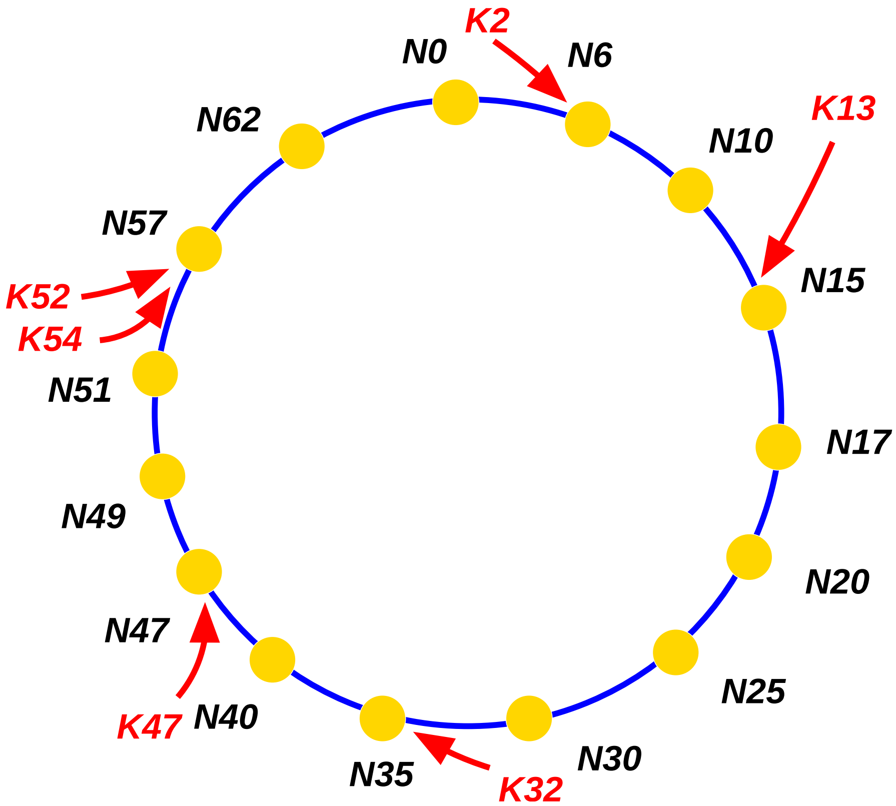
MIT Computer Science and Artificial Intelligence Laboratory

{benleong, jli}@mit.edu

Structured Peer-to-Peer Systems

- Large scale dynamic network
- Overlay infrastructure :
  - Scalable
  - Self configuring
  - Fault tolerant
- Every node responsible for some objects
- Find node having desired object
- Challenge: Efficient Routing at Low Cost

Address Space



- Most common — one-dimensional circular address space

Distributed Hash Tables (DHTs)

- A Distributed Hash Table (DHT) is a distributed data structure that supports a *put/get* interface.
- Store and retrieve {key, value} pairs efficiently over a network of (generally unreliable) nodes
- Early DHTs stored very little state ($O(\log n)$) to cope with network churn (Stoica et al., 2001; Ratnasamy et al., 2001; Zhao et al., 2001; Rowstron and Druschel, 2001).

Distributed Hash Tables (DHTs)

- Keep state stored per node small because of network churn $\Rightarrow$ minimize book-keeping & maintenance traffic
- Storage is cheap, so it is entirely reasonable to store a global lookup table at every node to achieve one-hop lookup (Gupta et al., 2004)
- Problem: Getting the routing information to all nodes and keeping it up-to-date

Anjali’s One-Hop Scheme (NDSI ’04)

- Hierarchical scheme
- Divide ring into slices – node at midpoint is the slice leader
- Slices further divided into units with leaders
- Slice leaders communicate with each other and with unit leaders in slice
- Information progagated along ring on *stay-a-live* messages

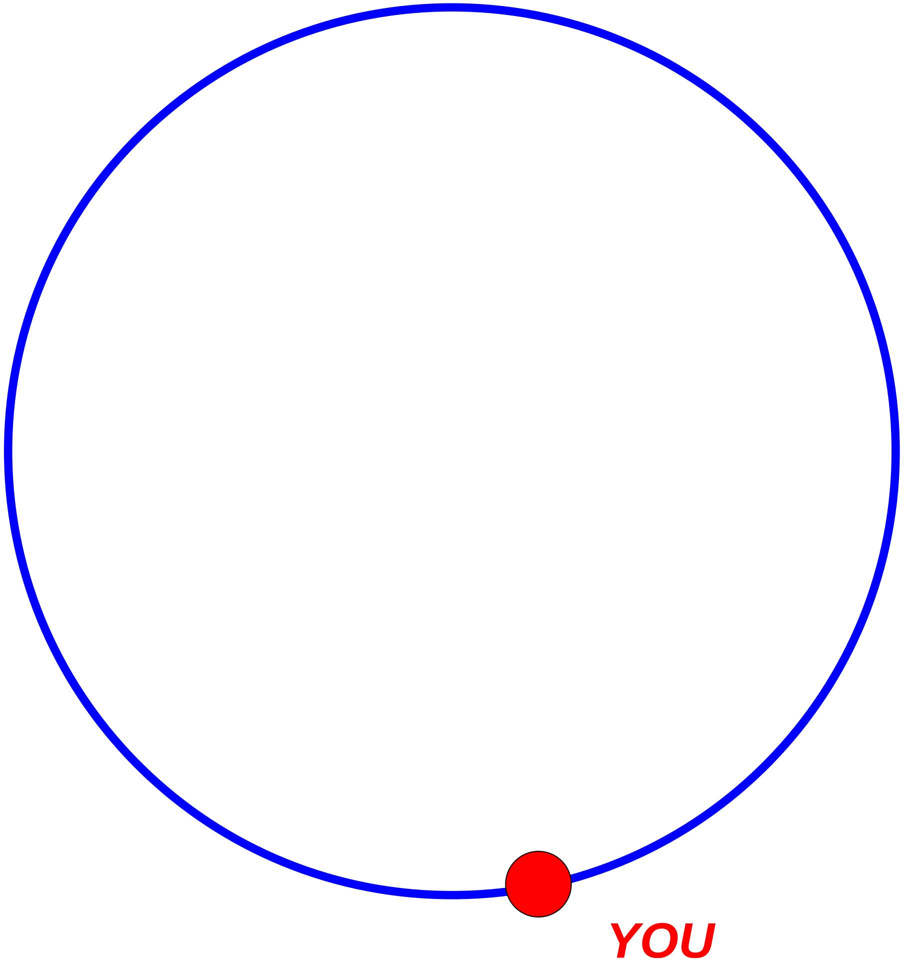
Anjali’s One-Hop Scheme

- Problems:
  - Slice leaders significantly more bandwidth than the other nodes
  - Several parameters — tuning requires knowledge of steady state network size
- Natural question: why don’t we just use per-event ad hoc broadcast trees?

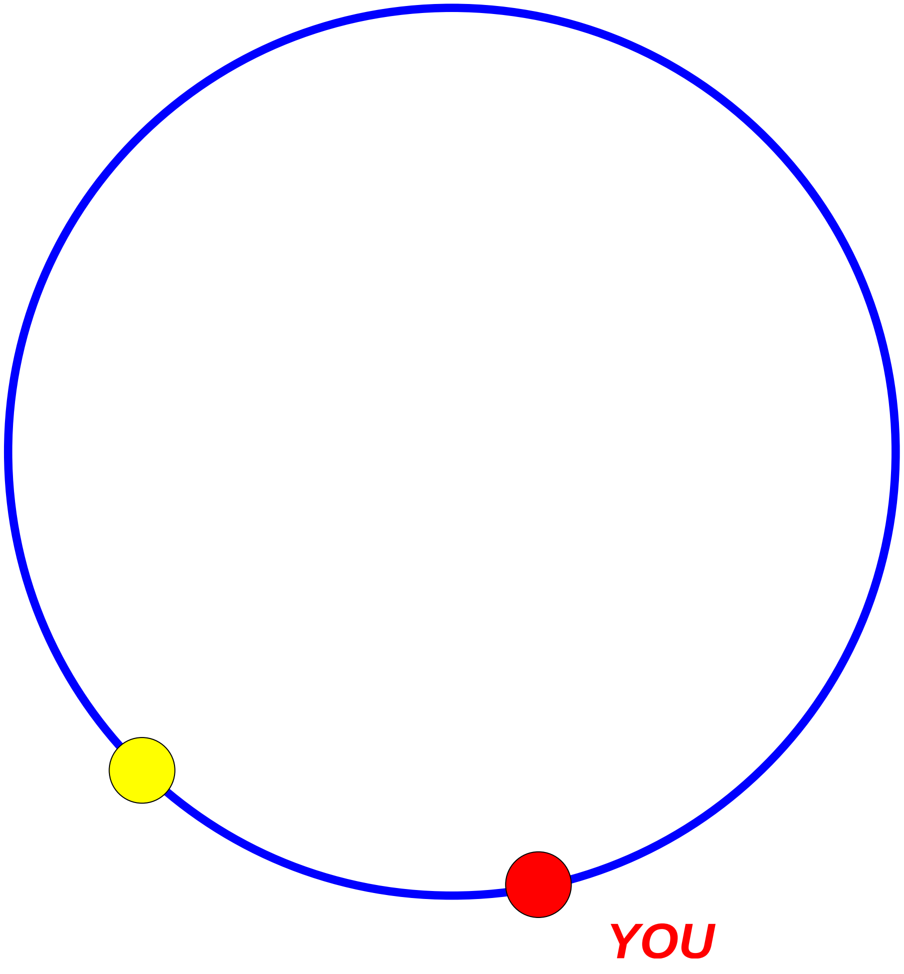
Token Passing Algorithm

- Token – message containing join or leave information (IP address, id).
- Also has a range of propagation, specified by destination n_d.
- When a node receives a token, it can:
  - Pass the token to its predecessor; or
  - Generate q secondary tokens that cover the remaining propagation range.

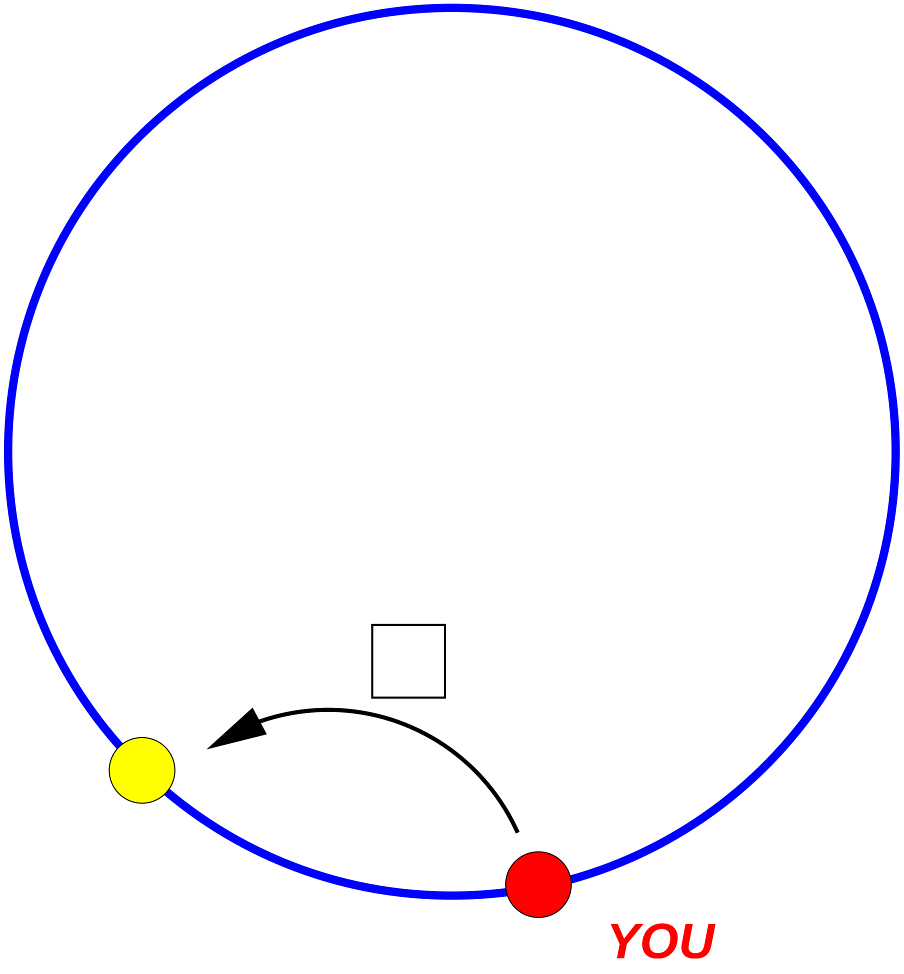
Token Passing Algorithm



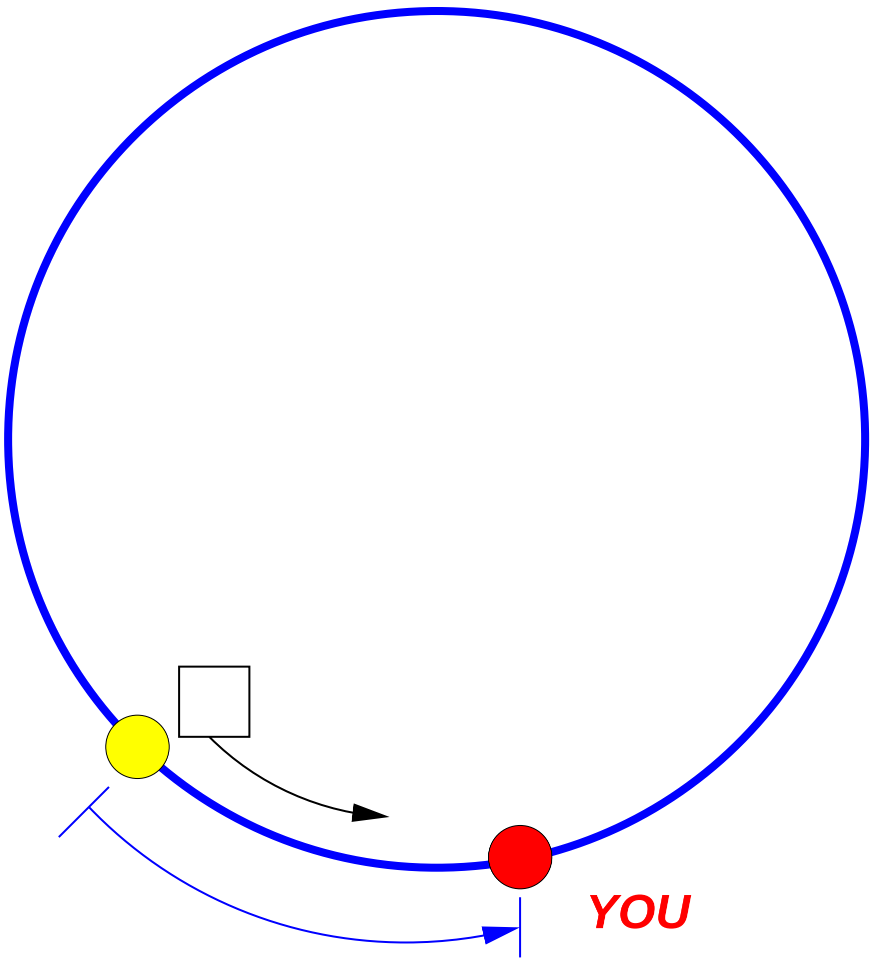
Token Passing Algorithm



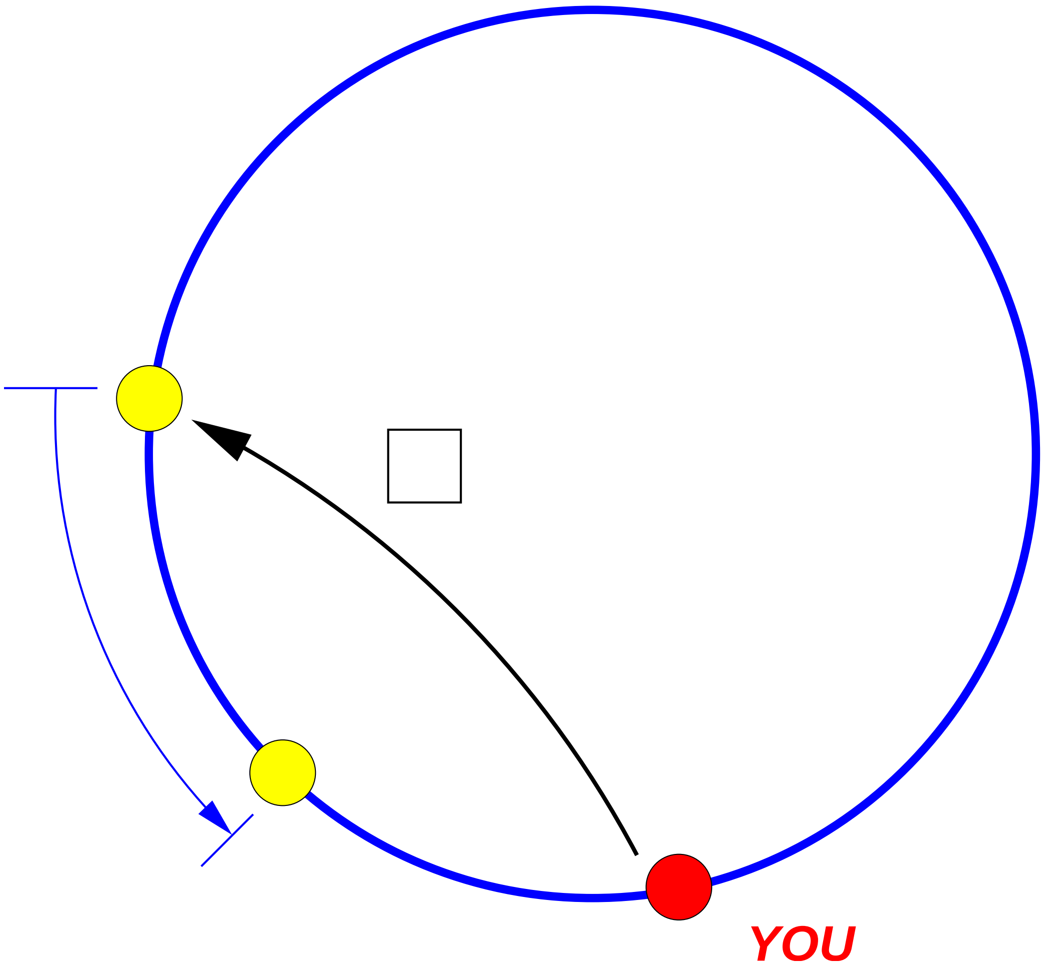
Token Passing Algorithm



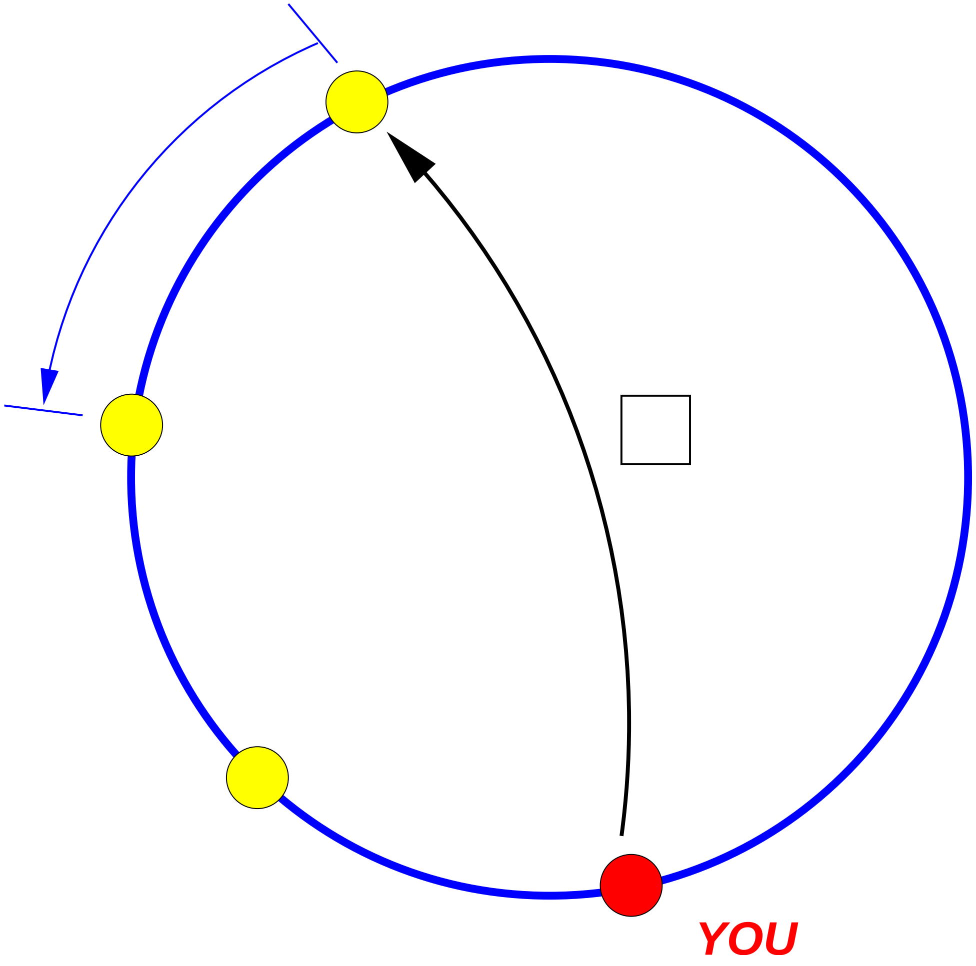
Token Passing Algorithm



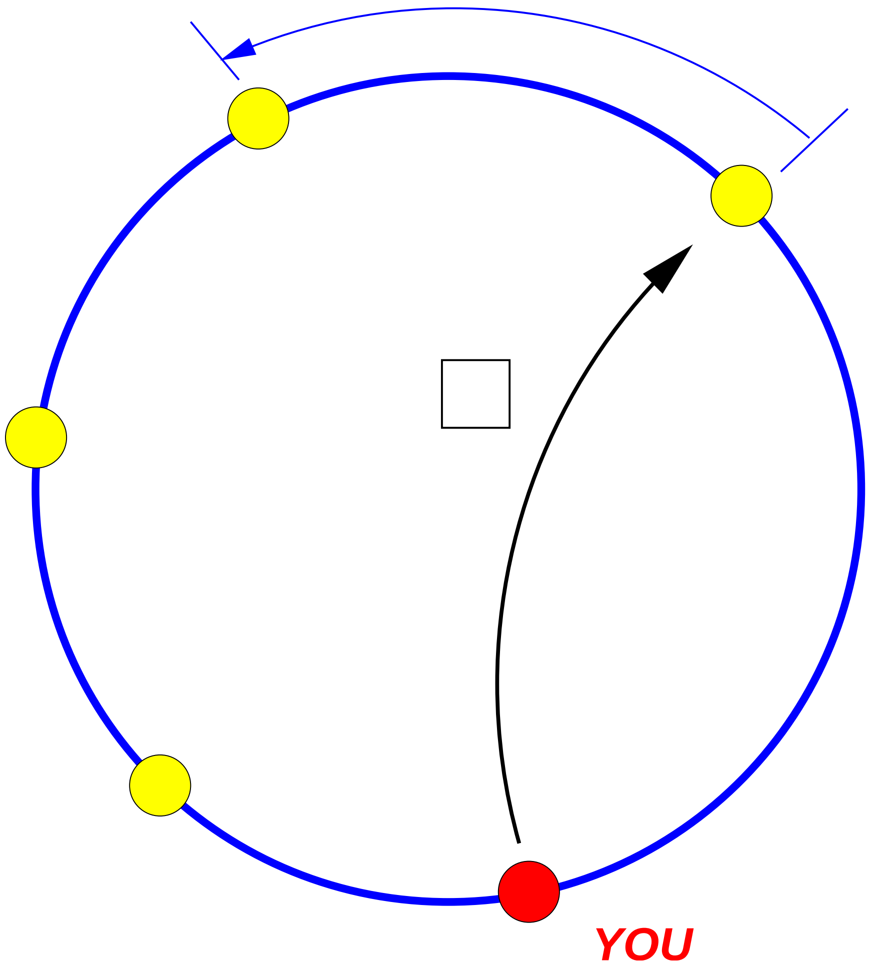
Token Passing Algorithm



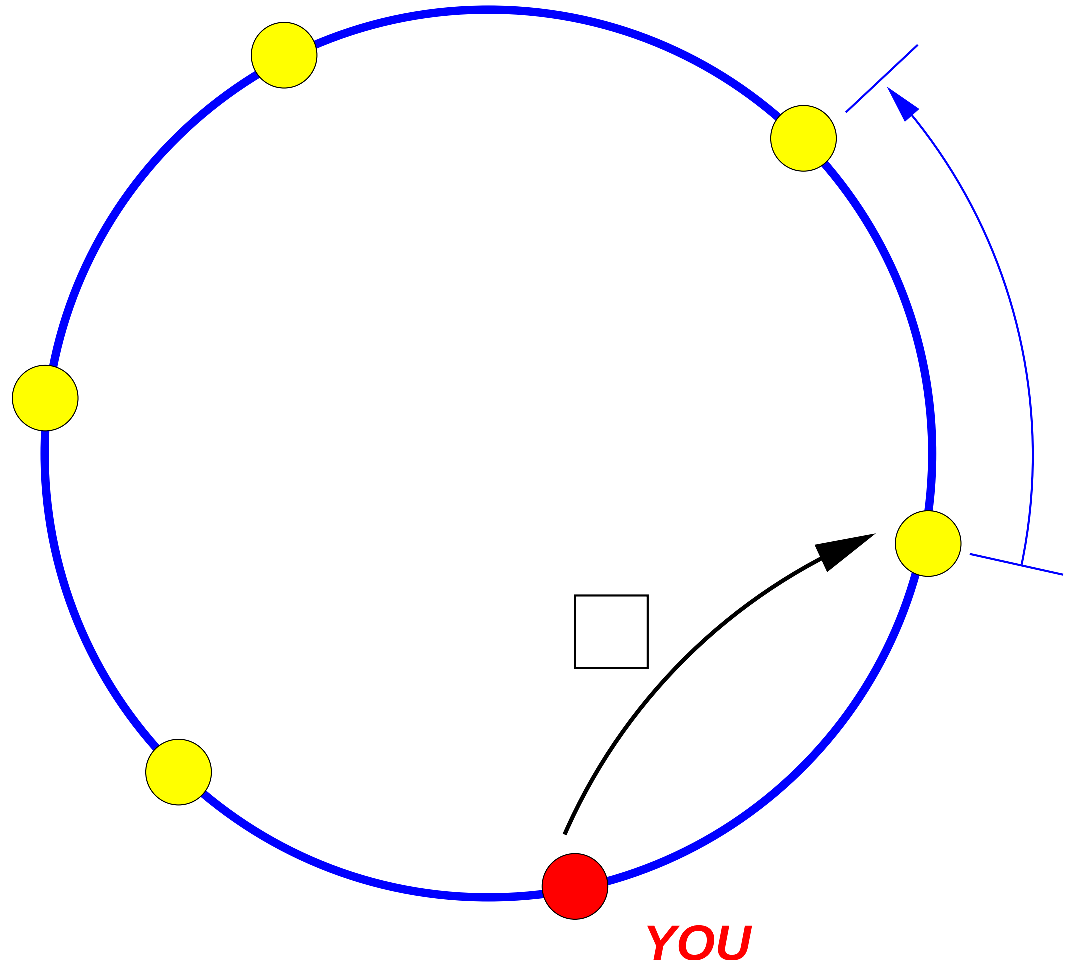
Token Passing Algorithm



Token Passing Algorithm

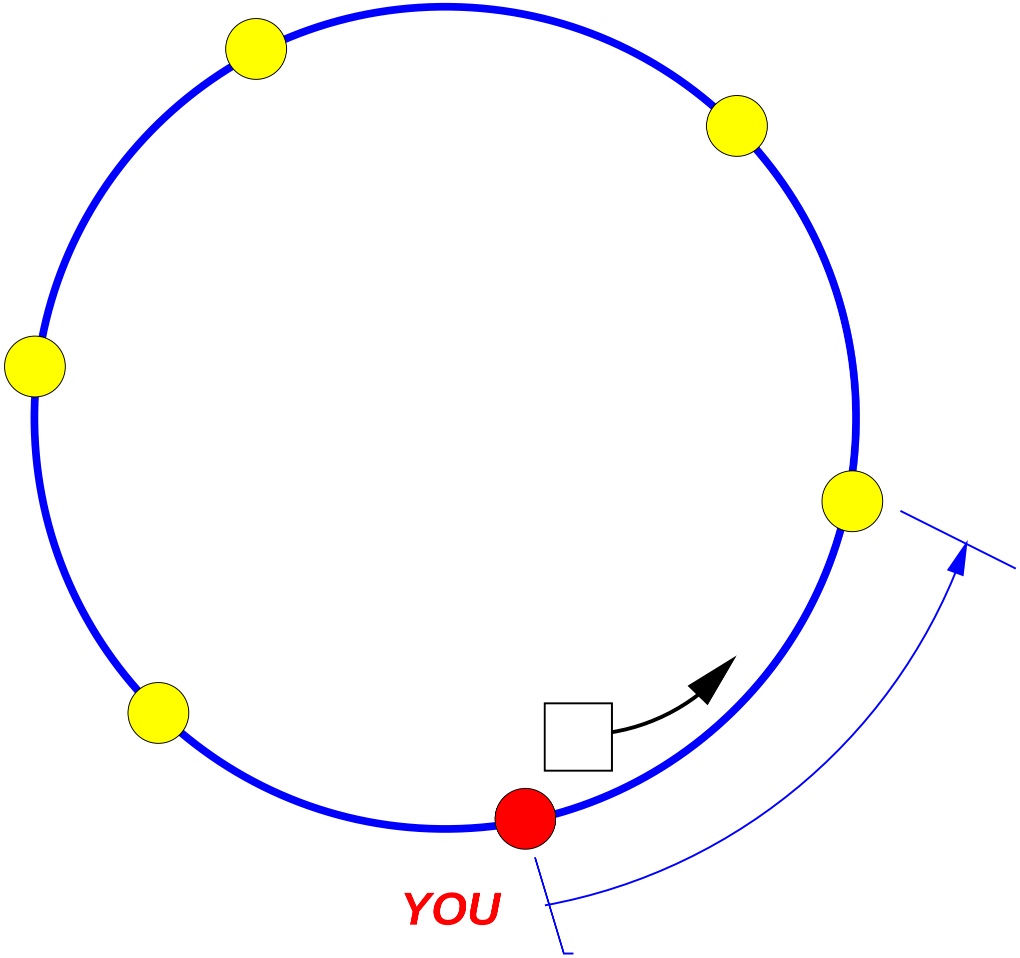


Token Passing Algorithm



- Pick q nodes in total

Token Passing Algorithm



- Pass last token to predecessor

Token Passing Algorithm

- Address space can be decomposed recursively
- Token is destroyed when it reaches its destination
- Nodes do not necessarily have to use a fixed q
- Tokens can be merged to save on propagation overhead

Lookup Algorithm

- Just contact the best known successor. It's probably the right one; if not, it will tell you where to go.
- Correctness of routing is guaranteed by correctness of successor/predecessor pointers
- In worst case, simply follow a chain of successor pointers – slow but correct.
- Stabilization – process that maintains and repairs successor/predecessor pointers

Definitions

We say that the network is

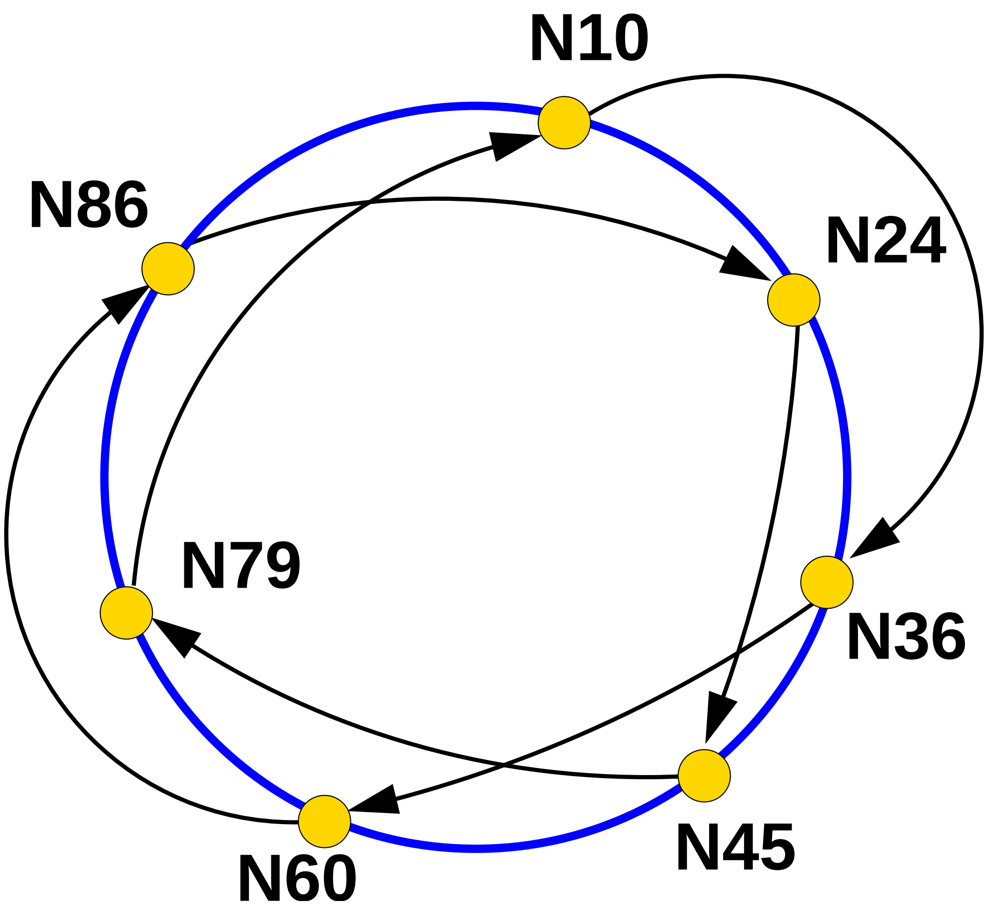
1. *weakly stable* if, for all nodes u, we have $predecessor(successor(u)) = u$;
2. *strongly stable* if, in addition, for each node u, there is no node v such that $u < v < successor(u)$; and
3. *loopy* if it is weakly but not strongly stable (see (Stoica et al., 2002)).

Weak Stabilization

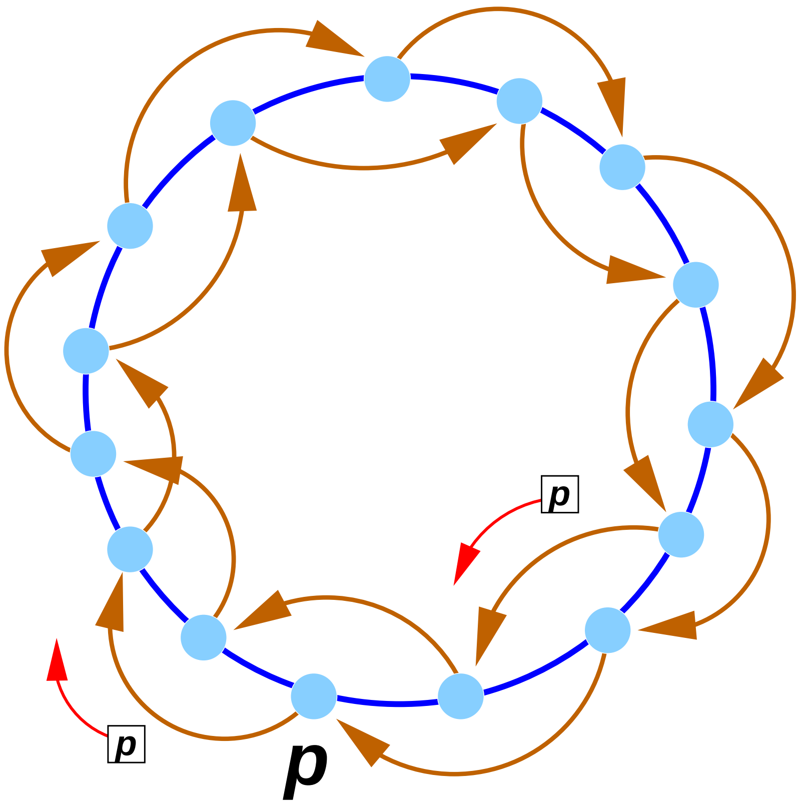
- Nodes periodically probe their immediate neighbors and exchange successor/predecessor lists
- All messages contain IP address, port number and node id

Theorem 1 *The weak stabilization protocol will eventually cause our network to converge to a* weakly stable *state.*

Loopy Example



Strong Stabilization



- Key idea: to detect loops, all we need to do is to traverse the entire ring and make sure that we come back to where we started

Strong Stabilization

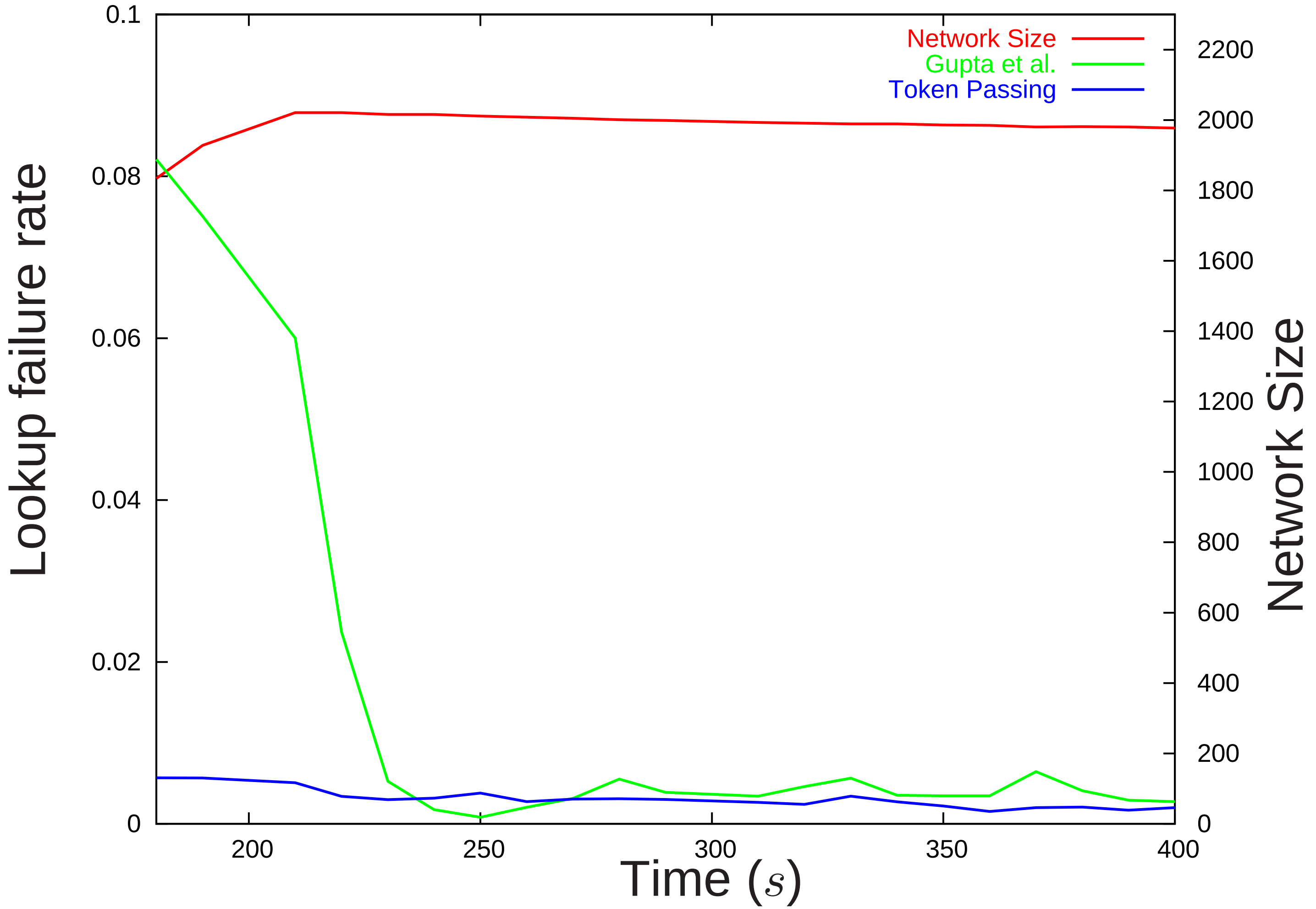
Theorem 2 *The combination of our parallel token-passing algorithm with the weak stabilization protocol will cause our network to converge to a* strongly stable *state within at most $O(n^2)$ rounds of token-passing.*

- Take any set of r nodes and have them send a message to the consecutive node.
- If a loop exists, at least one pair will detect it.
- Key Insight: this property does not change if you choose the r nodes recursively.

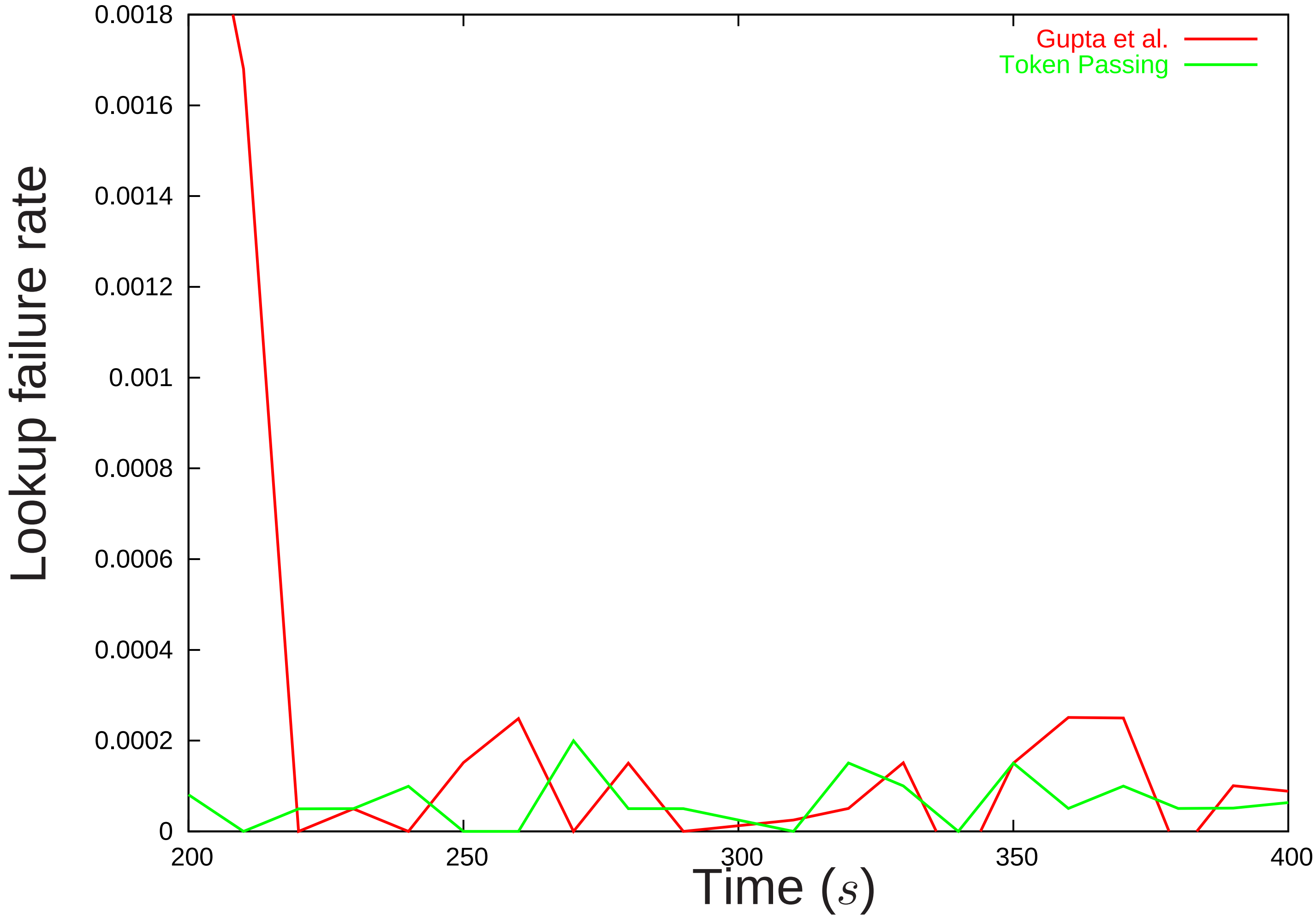
Simulation Results

- Analysis shows that our scheme is feasible in terms of bandwidth consumption under realistic assumptions
- Implemented algorithm in *p2psim* and compared it directly to Anjali's scheme
- Parameters:
  - Node lifetime: 60 mins
  - Avg 10 node joins per second for 200 s.
  - Nodes rejoin after mean interval of 6 mins
  - Query rate — 1 per sec per node

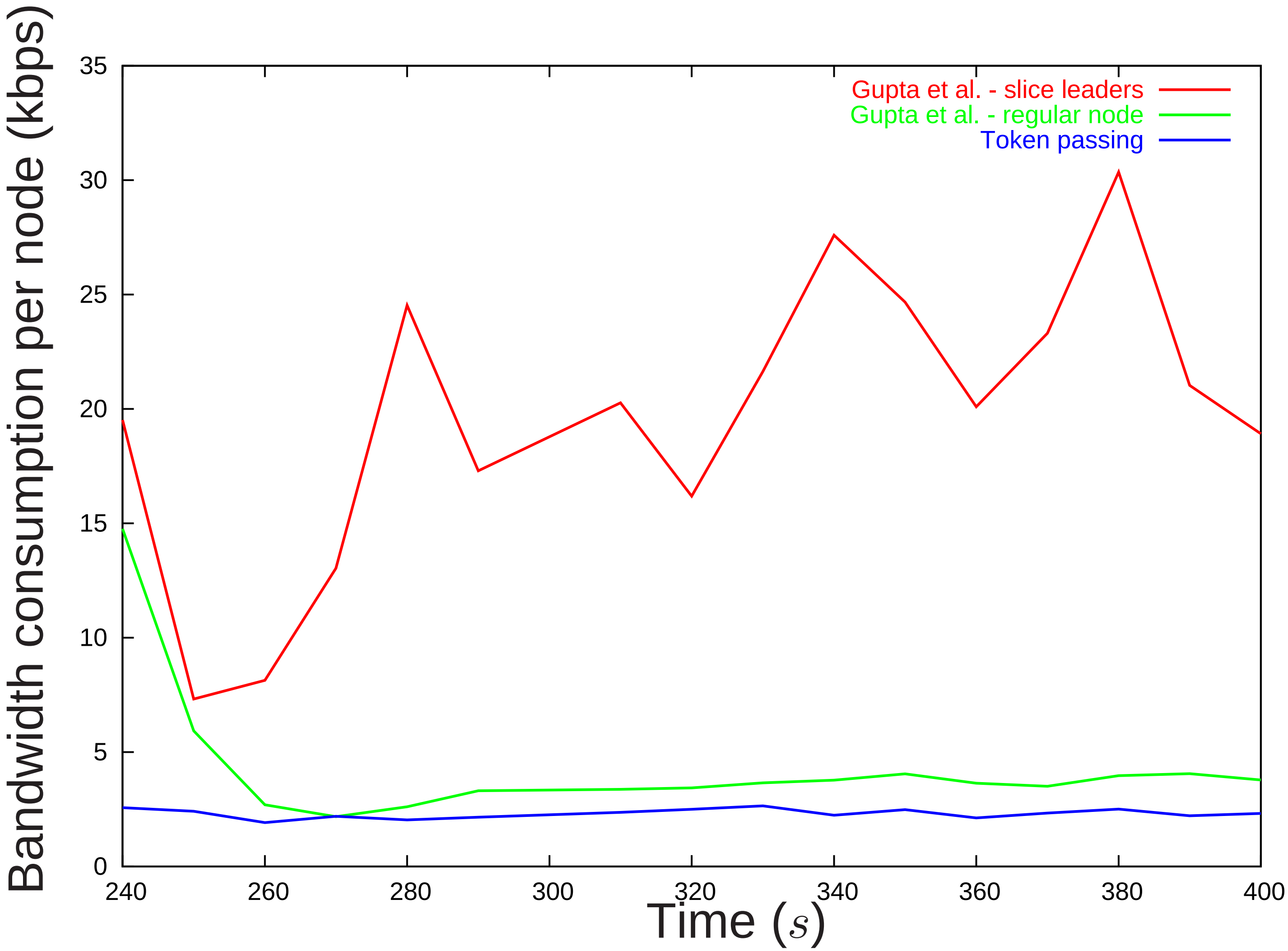
One-Hop Failure Rates



Two-Hop Failure Rates



Bandwidth Consumption



Related Work

- One-Hop (Gupta et al., 2004)
- Superpeers (Mizrak et al., 2003)
- Kelips (Gupta et al., 2003)

Conclusion

- Performs as well as Anjali's scheme
- Simplicity
- Imposes a slightly higher average overhead, but imposes uniform load
- Can vary q to adapt to network heterogeneity
- Have strong stabilization as a side-effect

Achieving One-Hop DHT Lookup and Strong Stabilization by Passing Tokens

Ben Leong and Ji Li

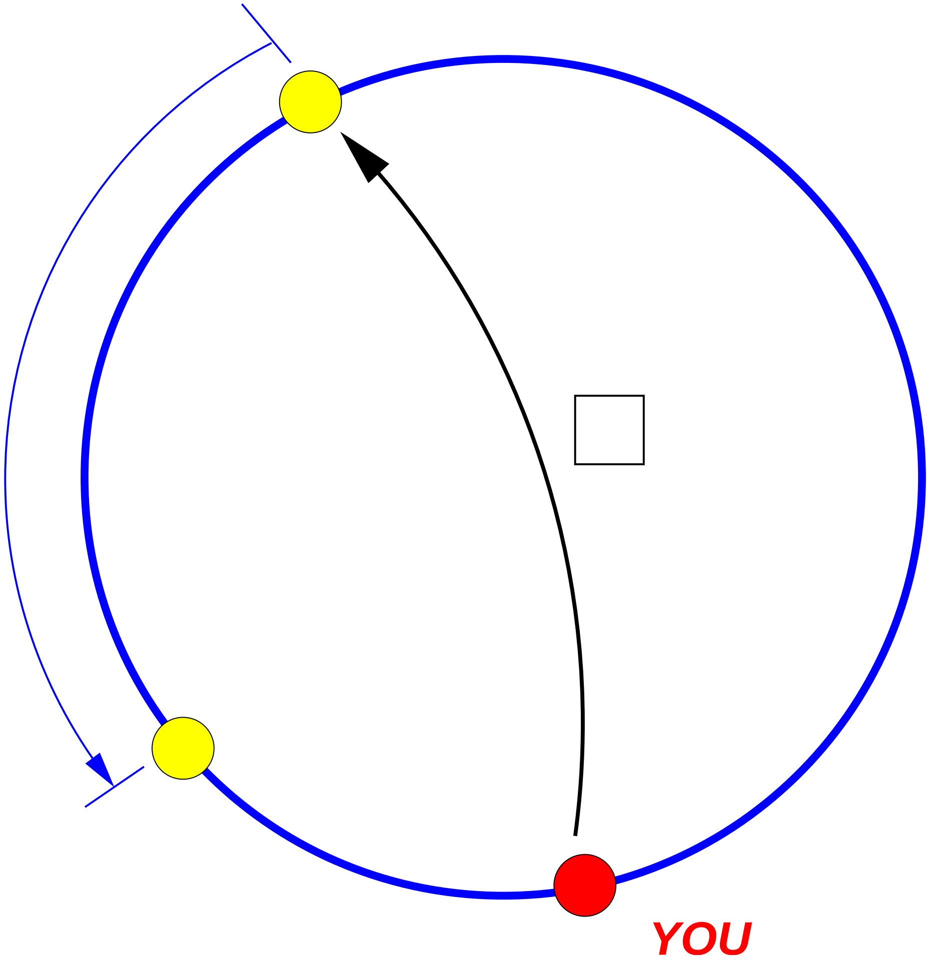
MIT Computer Science and Artificial Intelligence Laboratory

{benleong, jli}@mit.edu

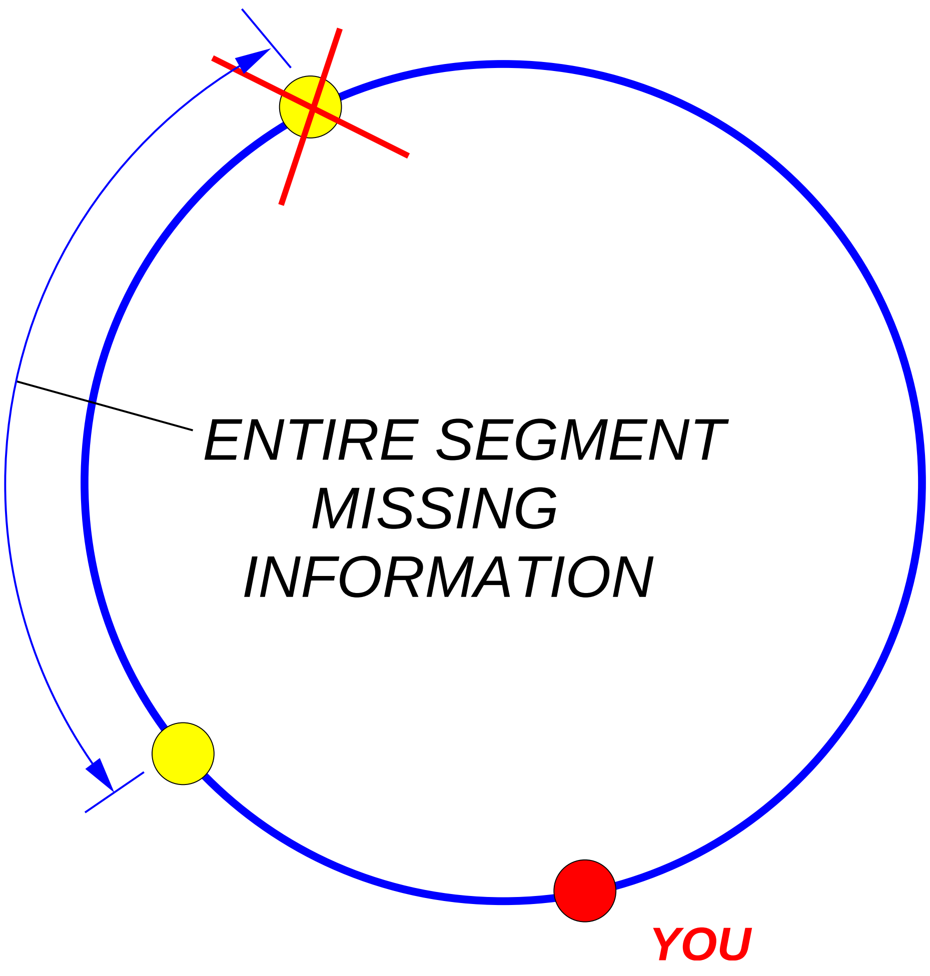
Fault Tolerance

- Sometimes bad things happen and a token is lost.
- Missing information is discovered during lookup process
- Observation: a lost token results in a consecutive segment of the address space missing some piece of information
- Solution: propagate *repair tokens*
  - Passed in both directions
  - Cannot be split

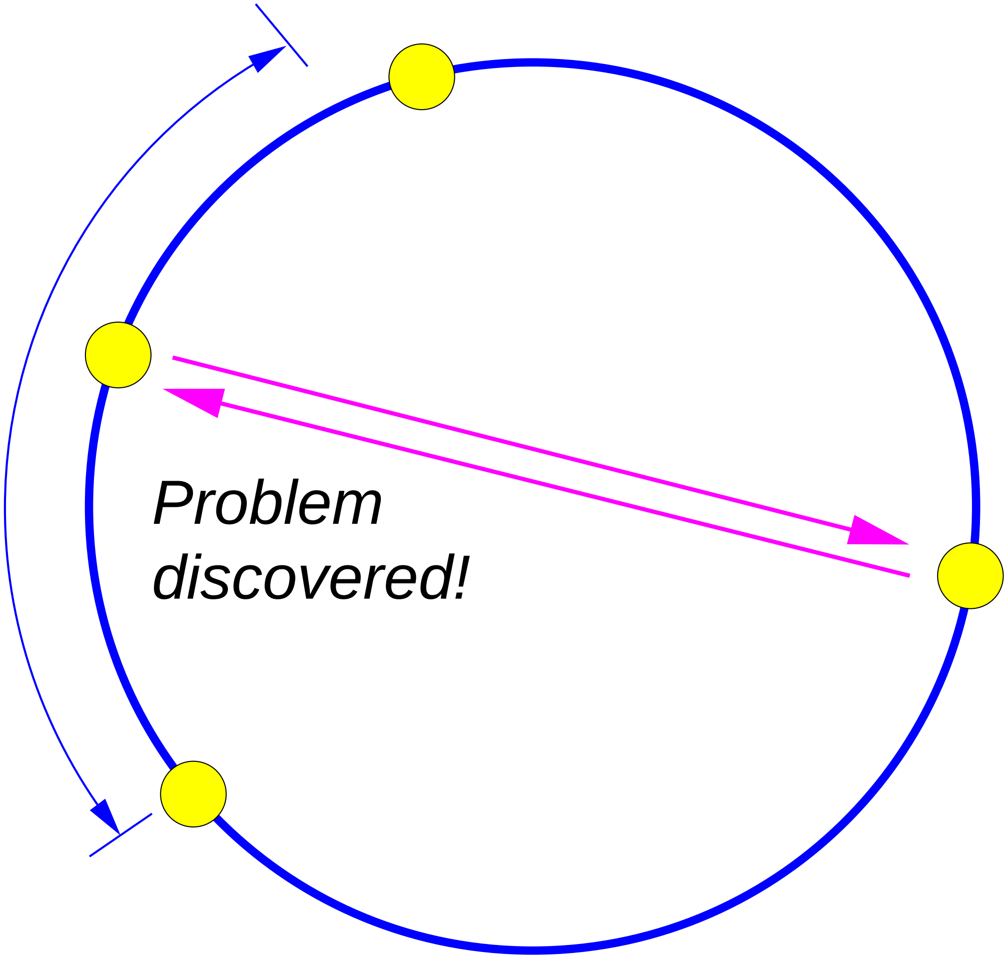
Repair



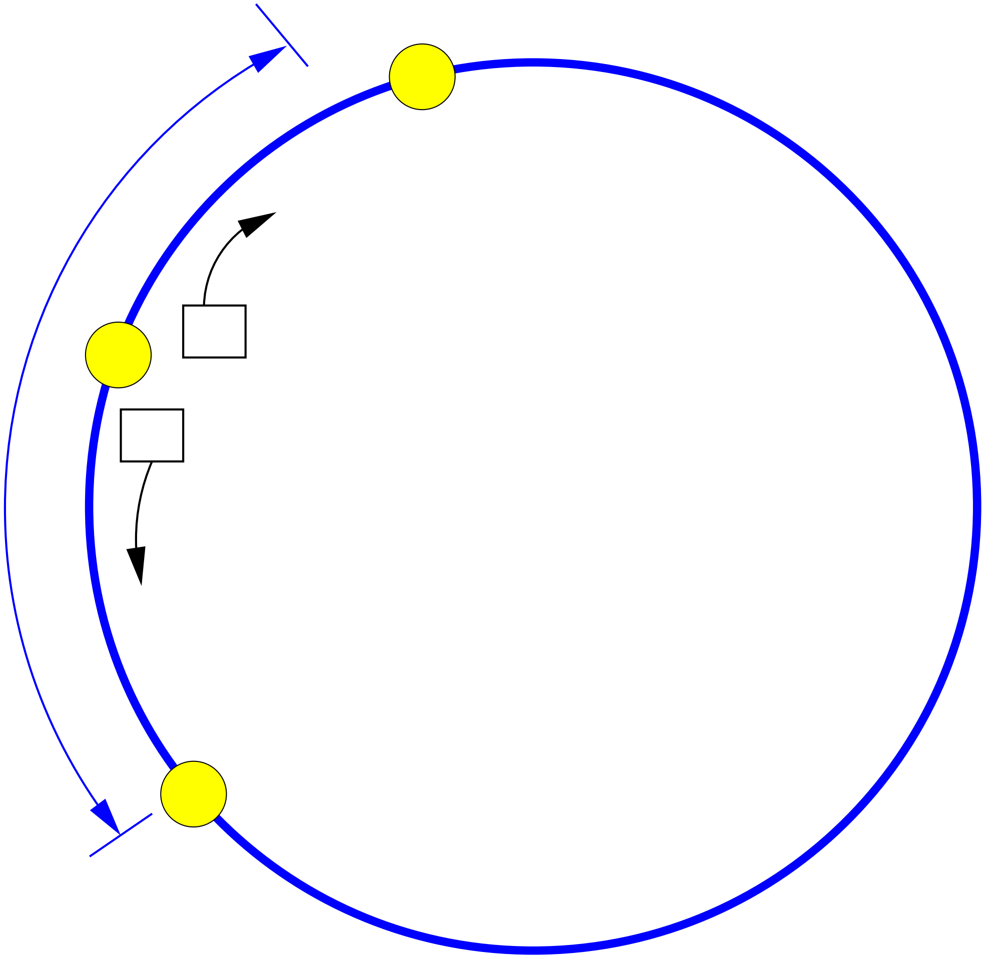
Repair



Repair



Repair



References

Gupta, A., Liskov, B., and Rodrigues, R. (2004). Efficient routing for peer-to-peer overlays. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI 2004)*, pages 113–126.

Gupta, I., Birman, K., Linga, P., Demers, A., and van Renesse, R. (2003). Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*.

Mizrak, A., Cheng, Y., Kumar, V., and Savage, S. (2003). Structured superpeers: Leveraging heterogeneity to provide constant-time lookup. In *Proceedings of the 4th IEEE Workshop on Internet Applications*.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. (2001). A scalable content-addressable network. In *Proceedings of the 2001 ACM SIGCOMM Conference*.

Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*.

Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H. (2001). Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001*

ACM SIGCOMM Conference, pages 149–160.

Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. (2002). Chord: A scalable peer-to-peer lookup service for internet applications. Technical report, MIT LCS.

Zhao, B. Y., Kubiatowicz, J. D., and Joseph, A. D. (2001). Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley.