Tapestry Deployment and Fault-tolerant Routing

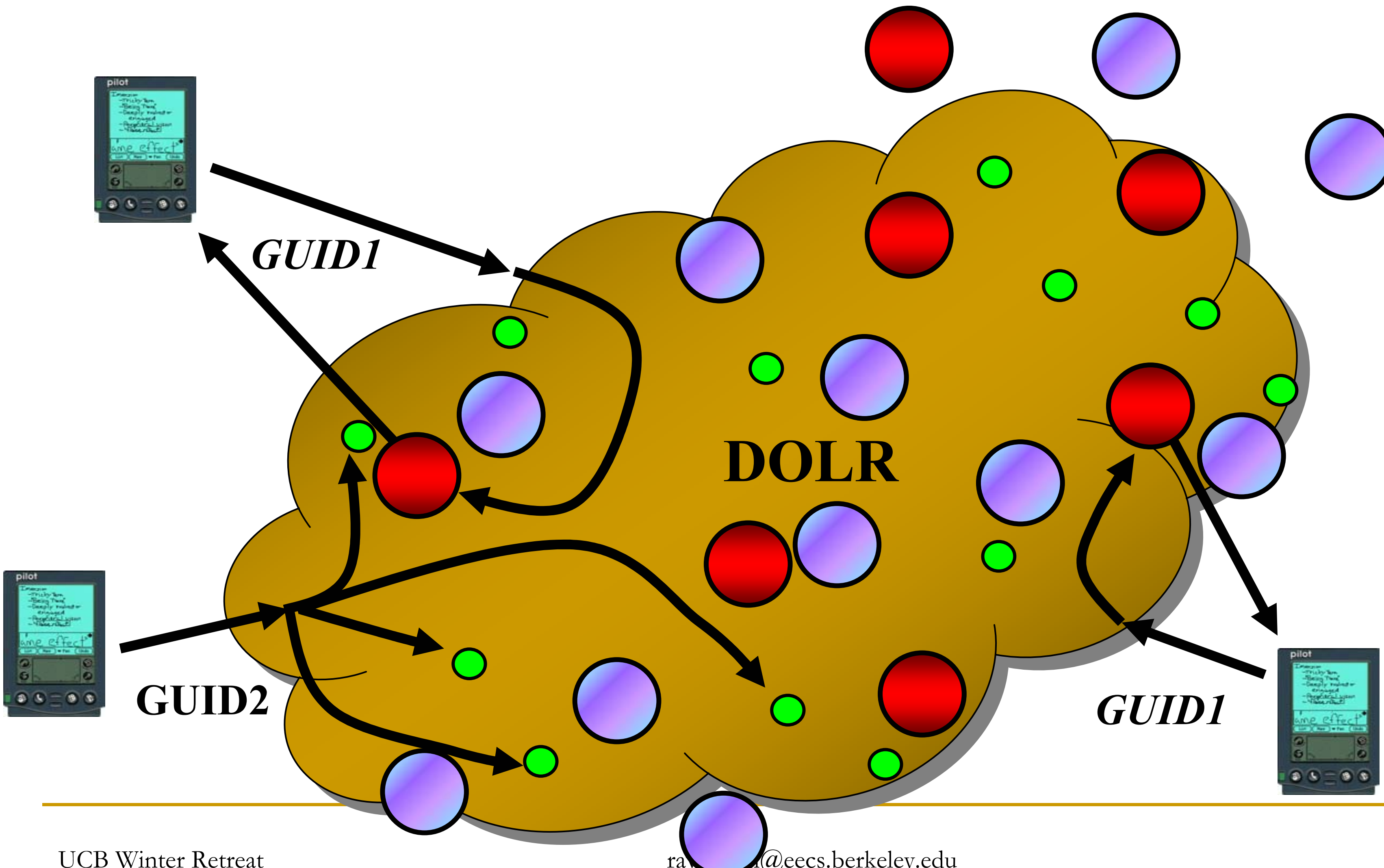
Ben Y. Zhao
L. Huang, S. Rhea, J. Stribling,
A. D. Joseph, J. D. Kubiatowicz

Berkeley Research Retreat
January 2003

Scaling Network Applications

- Complexities of global deployment
  - Network unreliability
    - BGP slow convergence, redundancy unexploited
  - Lack of administrative control over components
    - Constrains protocol deployment: multicast, congestion ctrl.
  - Management of large scale resources / components
    - Locate, utilize resources despite failures

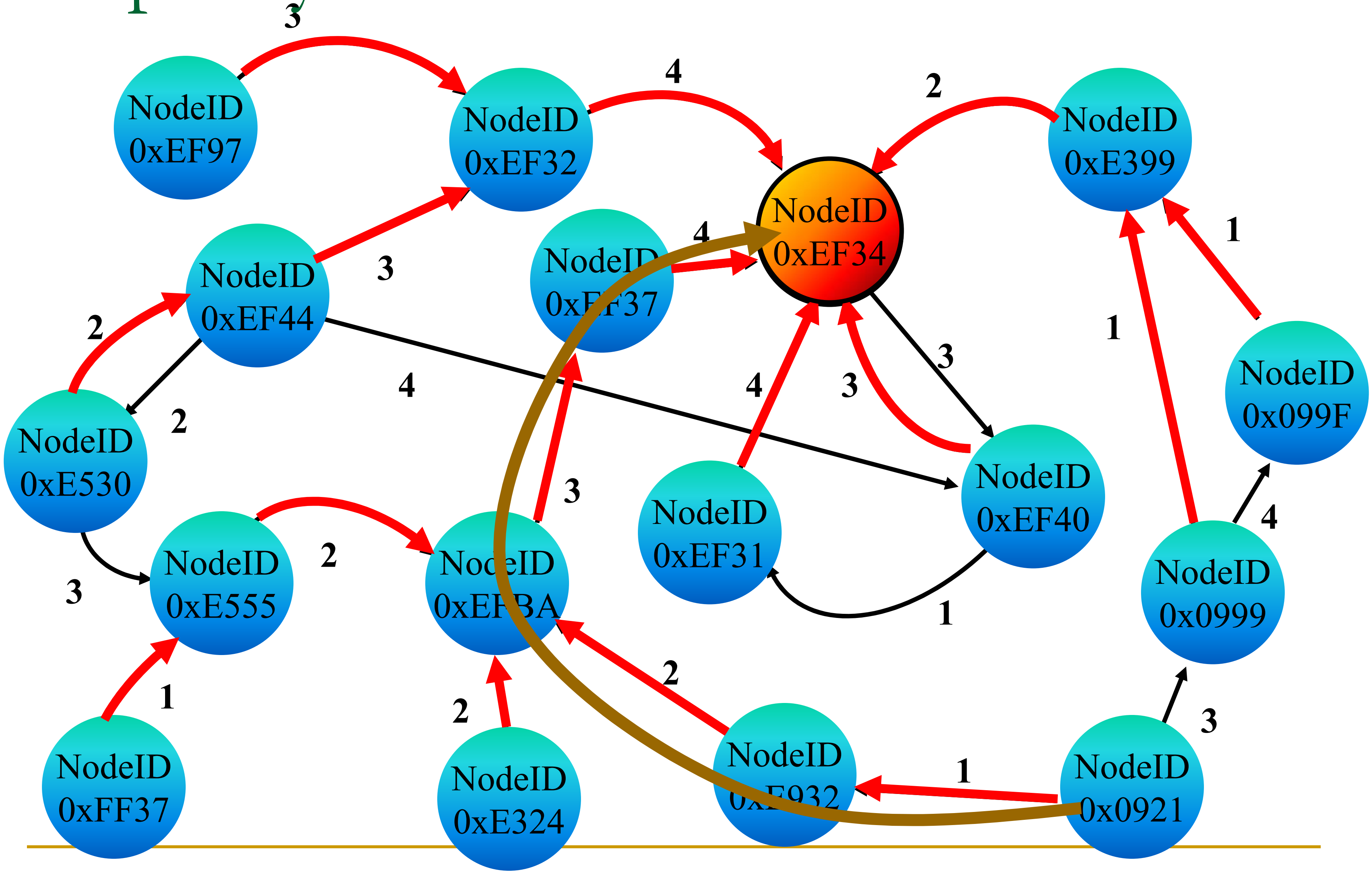
Enabling Technology: DOLR
(Decentralized Object Location and Routing)



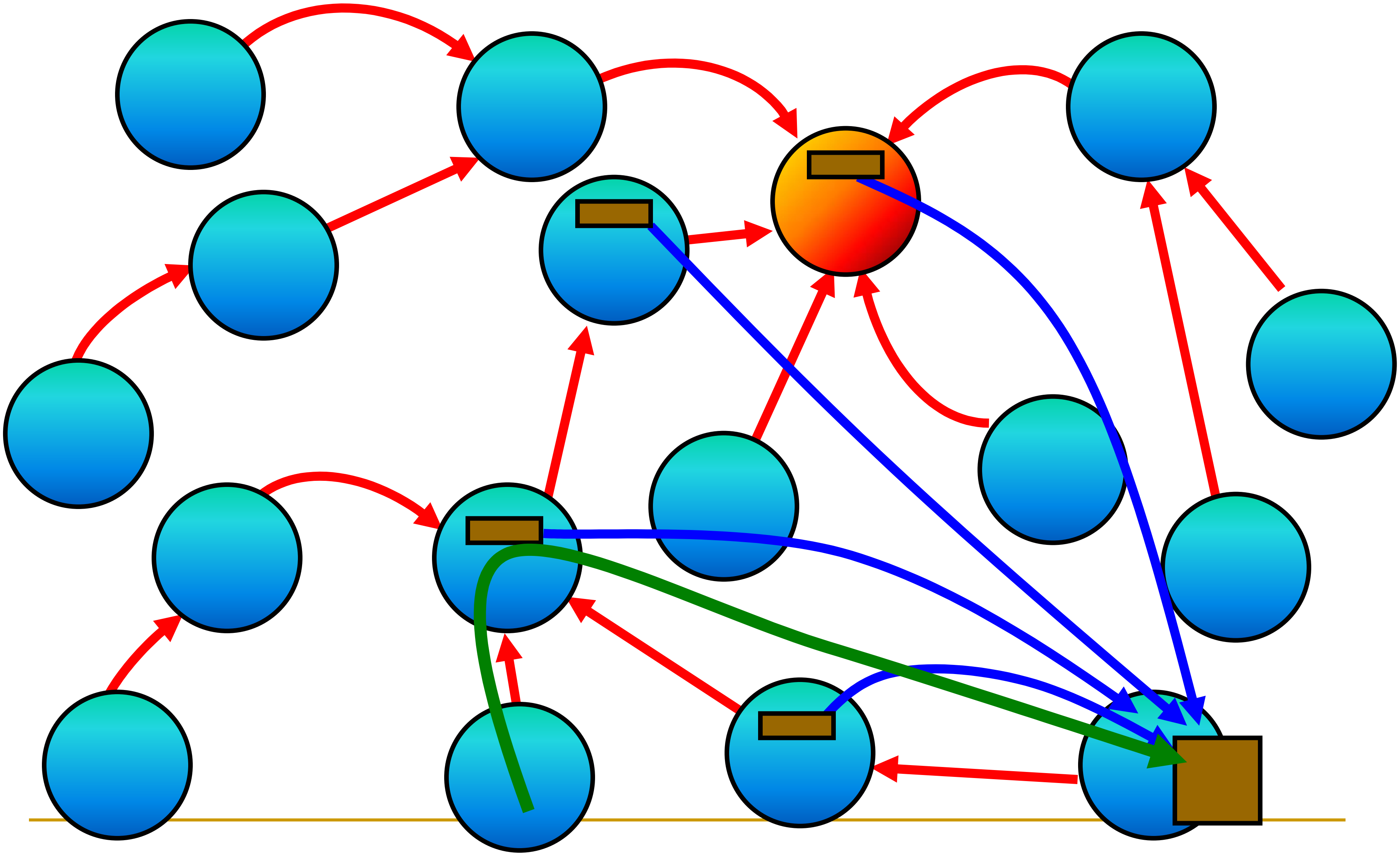
What is Tapestry?

- DOLR driving OceanStore global storage
  (Zhao, Kubiatowicz, Joseph et al. 2000)
- Network structure
  - Nodes assigned bit sequence **nodeIds** from namespace: $0\text{-}2^{160}$, based on some radix (e.g. 16)
  - **keys** from same namespace
    Keys dynamically map to 1 unique live node: *root*
- Base API
  - Publish / Unpublish (Object ID)
  - RouteToNode (NodeId)
  - RouteToObject (Object ID)

Tapestry Mesh



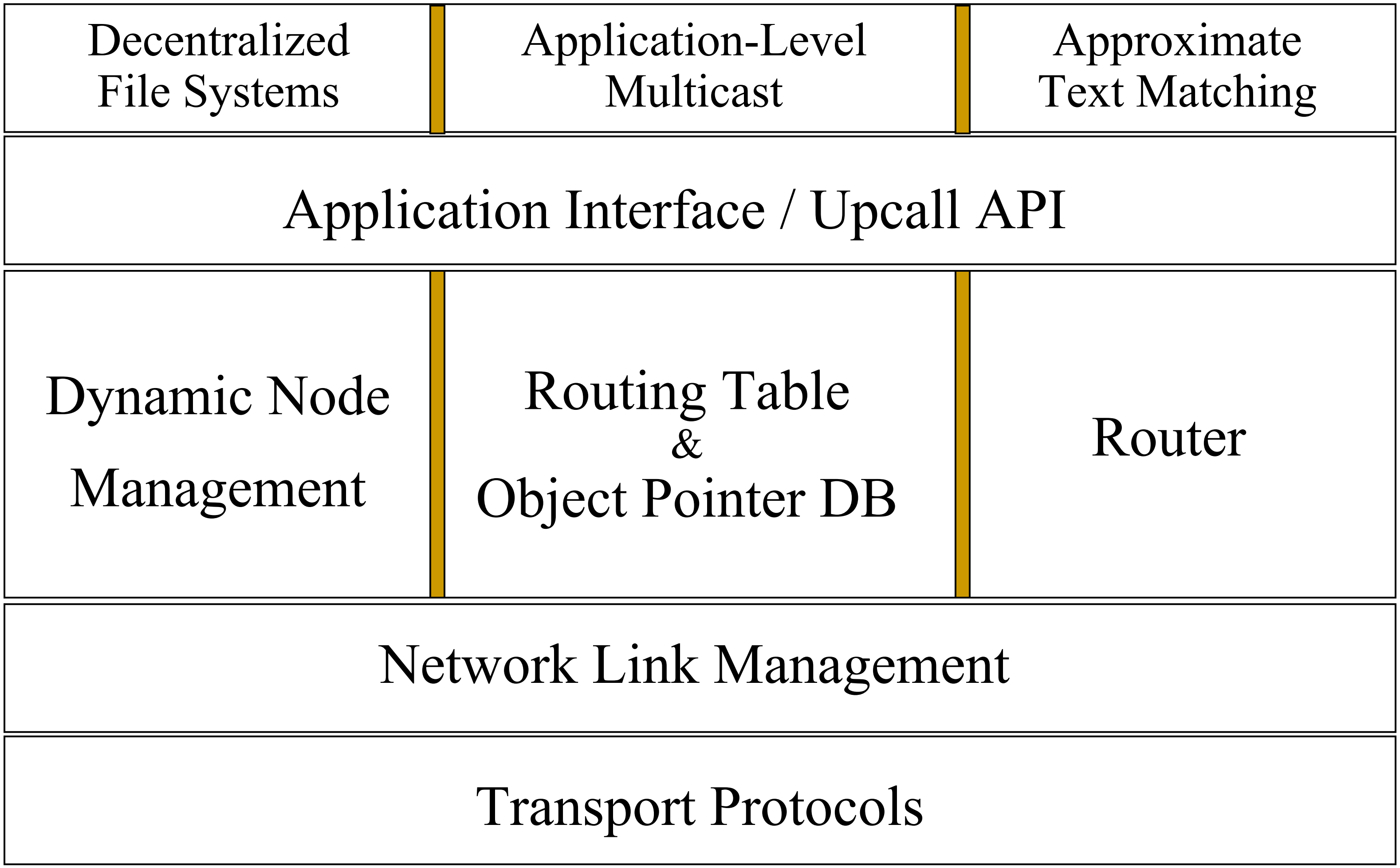
Object Location



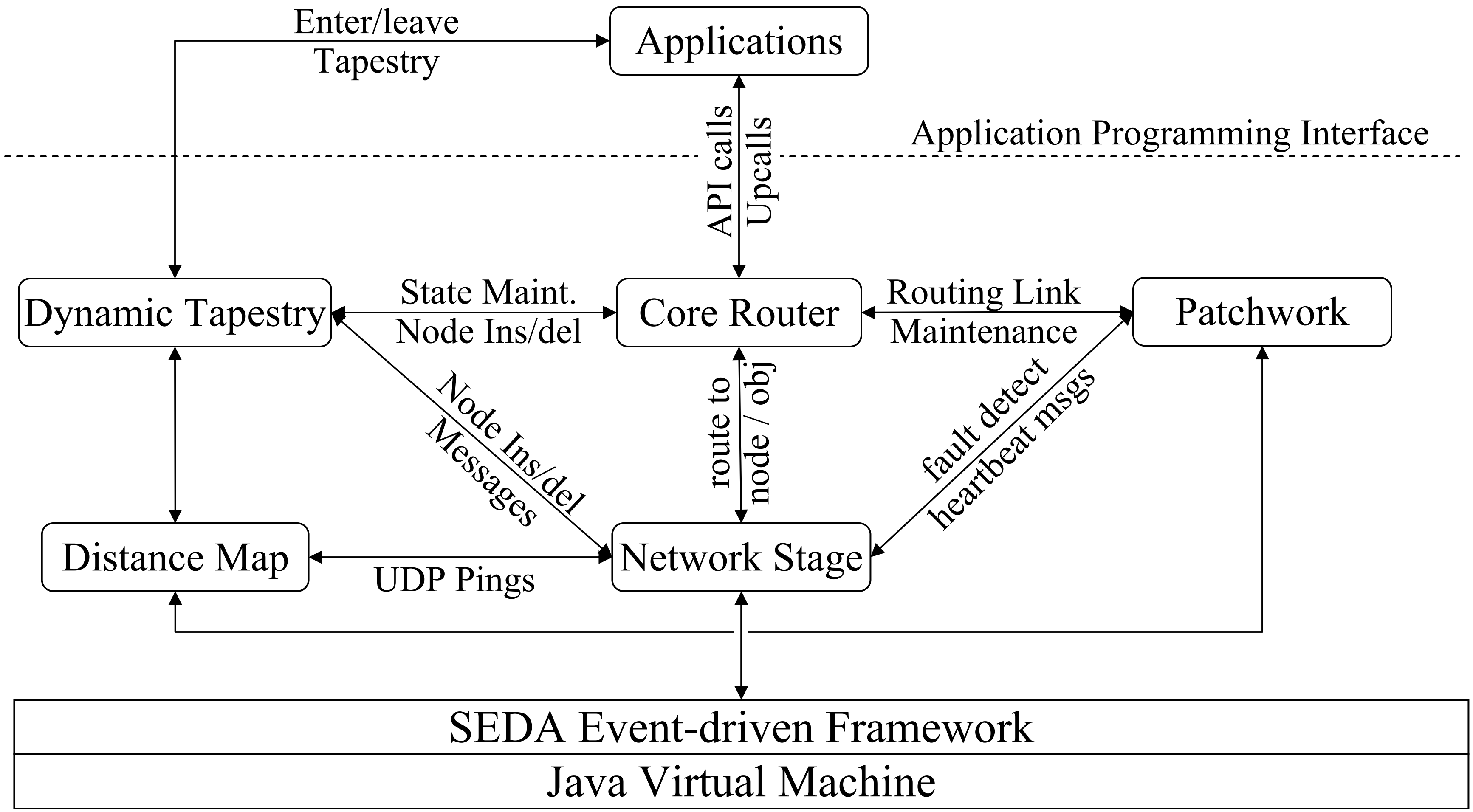
Talk Outline

- Introduction
- Architecture
  - Node architecture
  - Node implementation
- Deployment Evaluation
- Fault-tolerant Routing

Single Node Architecture



Single Node Implementation



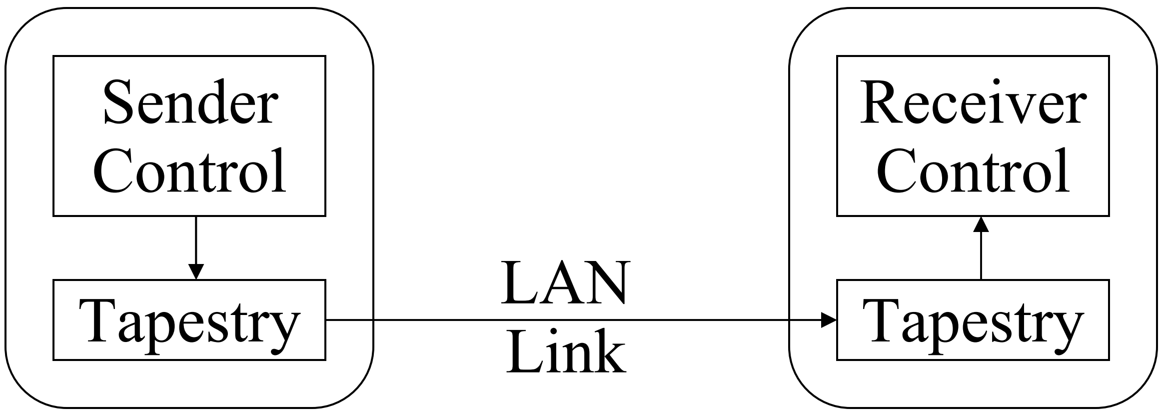
Deployment Status

- C simulator
  - Packet level simulation
  - Scales up to 10,000 nodes
- Java implementation
  - 50000 semicolons of Java, 270 class files
  - Deployed on local area cluster (40 nodes)
  - Deployed on Planet Lab global network (~100 distributed nodes)

Talk Outline

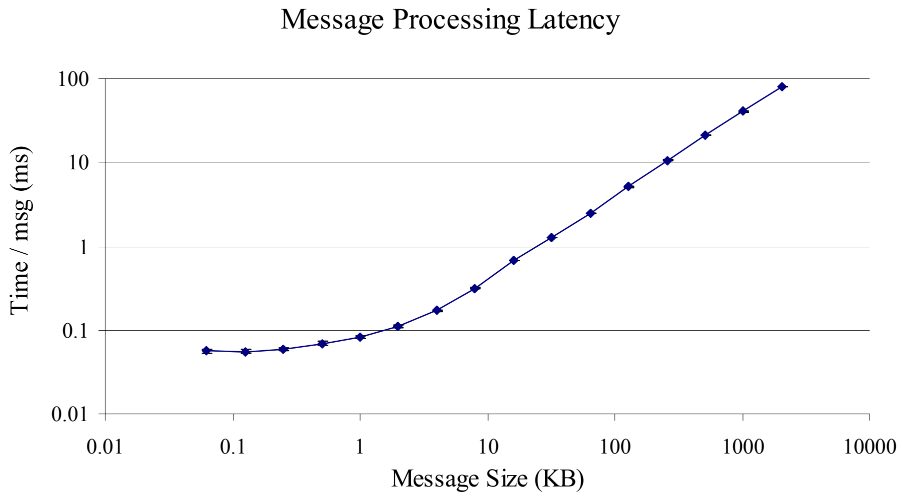
- Introduction
- Architecture
- Deployment Evaluation
  - Micro-benchmarks
  - Stable network performance
  - Single and parallel node insertion
- Fault-tolerant Routing

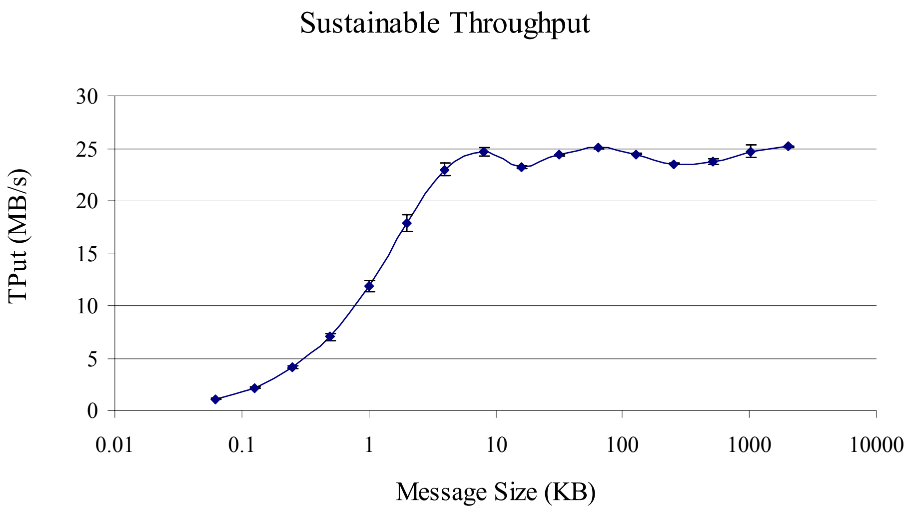
Micro-benchmark Methodology



- Experiment run in LAN, GBit Ethernet
- Sender sends 60001 messages at full speed
- Measure inter-arrival time for last 50000 msgs
  - 10000 msgs: remove cold-start effects
  - 50000 msgs: remove network jitter effects

Micro-benchmark Results



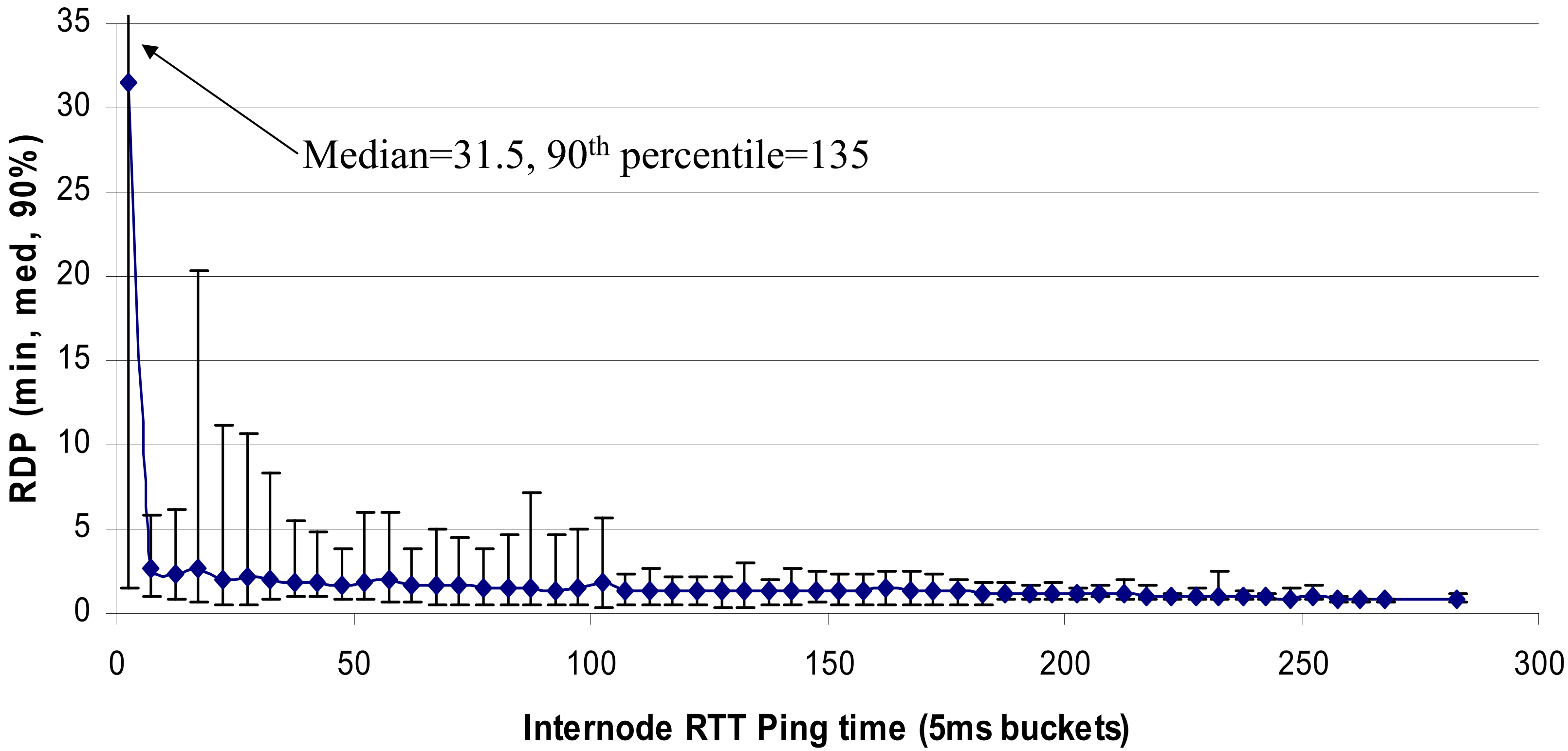


- Constant processing overhead ~ 50μs
- Latency dominated by byte copying
- For 5K messages, throughput = ~10,000 msgs/sec

Large Scale Methodology

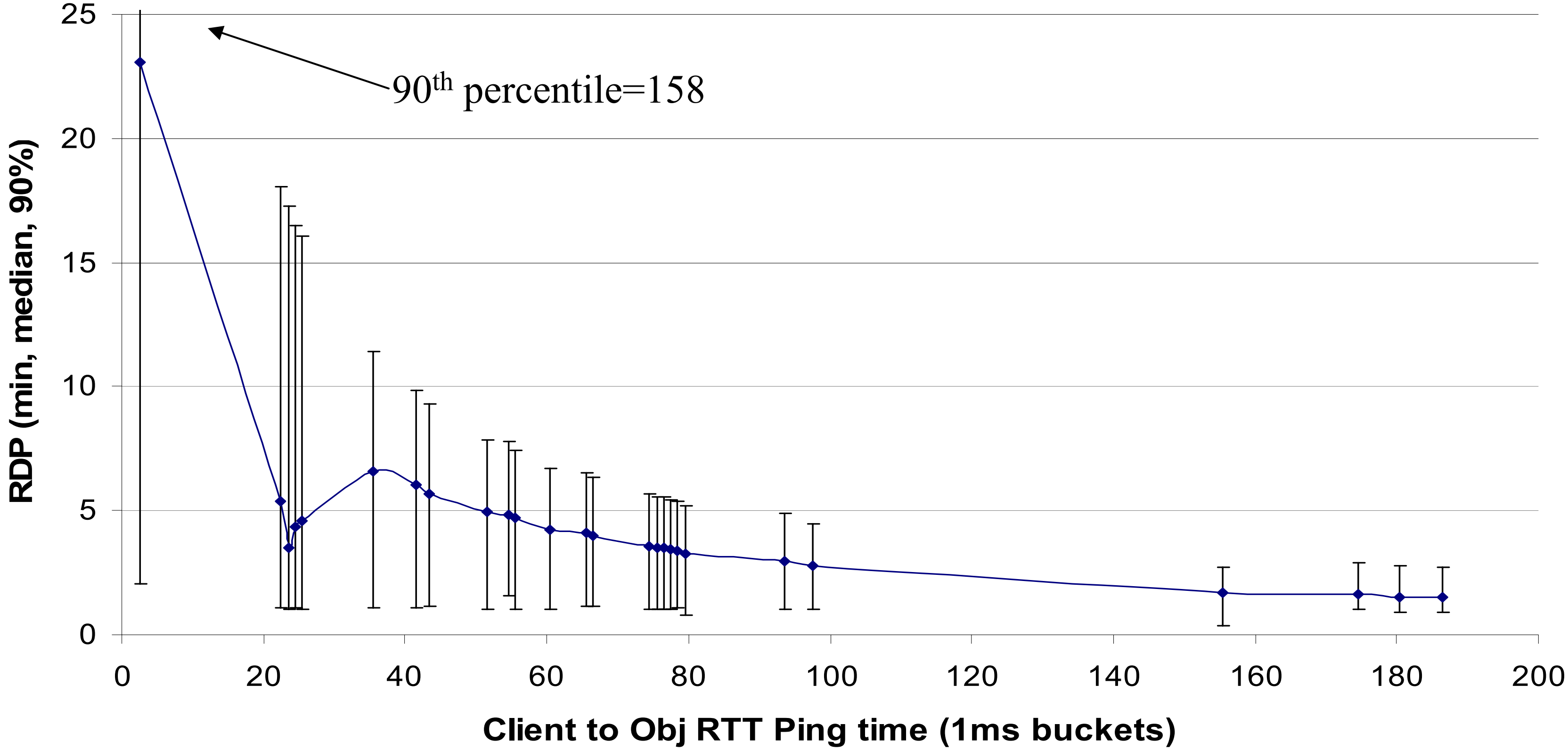
- PlanetLab global network
  - 101 machines at 42 institutions, in North America, Europe, Australia (~ 60 machines utilized)
  - 1.26Ghz PIII (1GB RAM), 1.8Ghz P4 (2GB RAM)
  - North American machines (2/3) on Internet2
- Tapestry Java deployment
  - 6-7 nodes on each physical machine
  - IBM Java JDK 1.30
  - Node virtualization inside JVM and SEDA
  - Scheduling between virtual nodes increases latency

Node to Node Routing



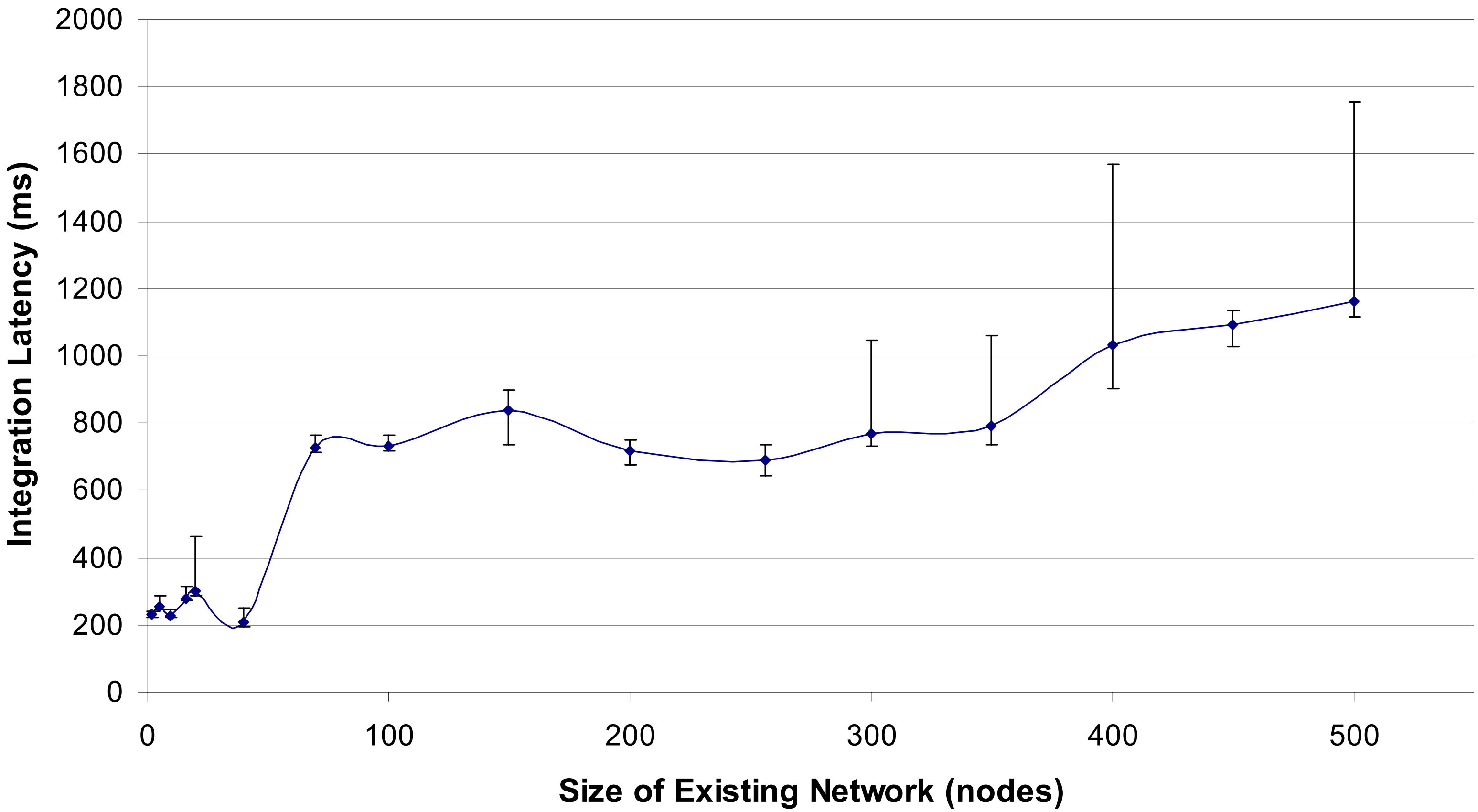
- Ratio of end-to-end routing latency to shortest ping distance between nodes
- All node pairs measured, placed into buckets

Object Location



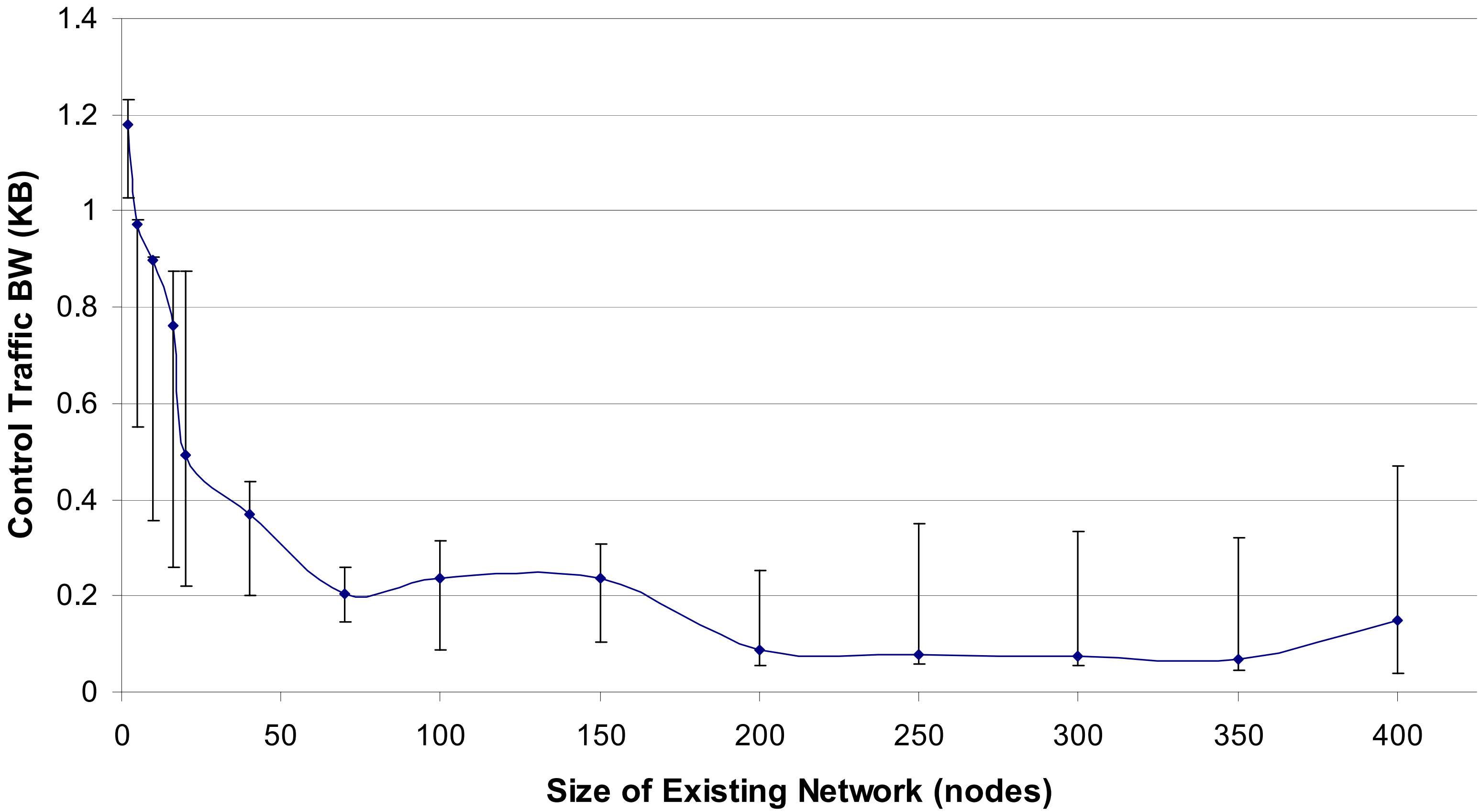
- Ratio of end-to-end latency for object location, to shortest ping distance between client and object location
- Each node publishes 10,000 objects, lookup on all objects

Latency to Insert Node



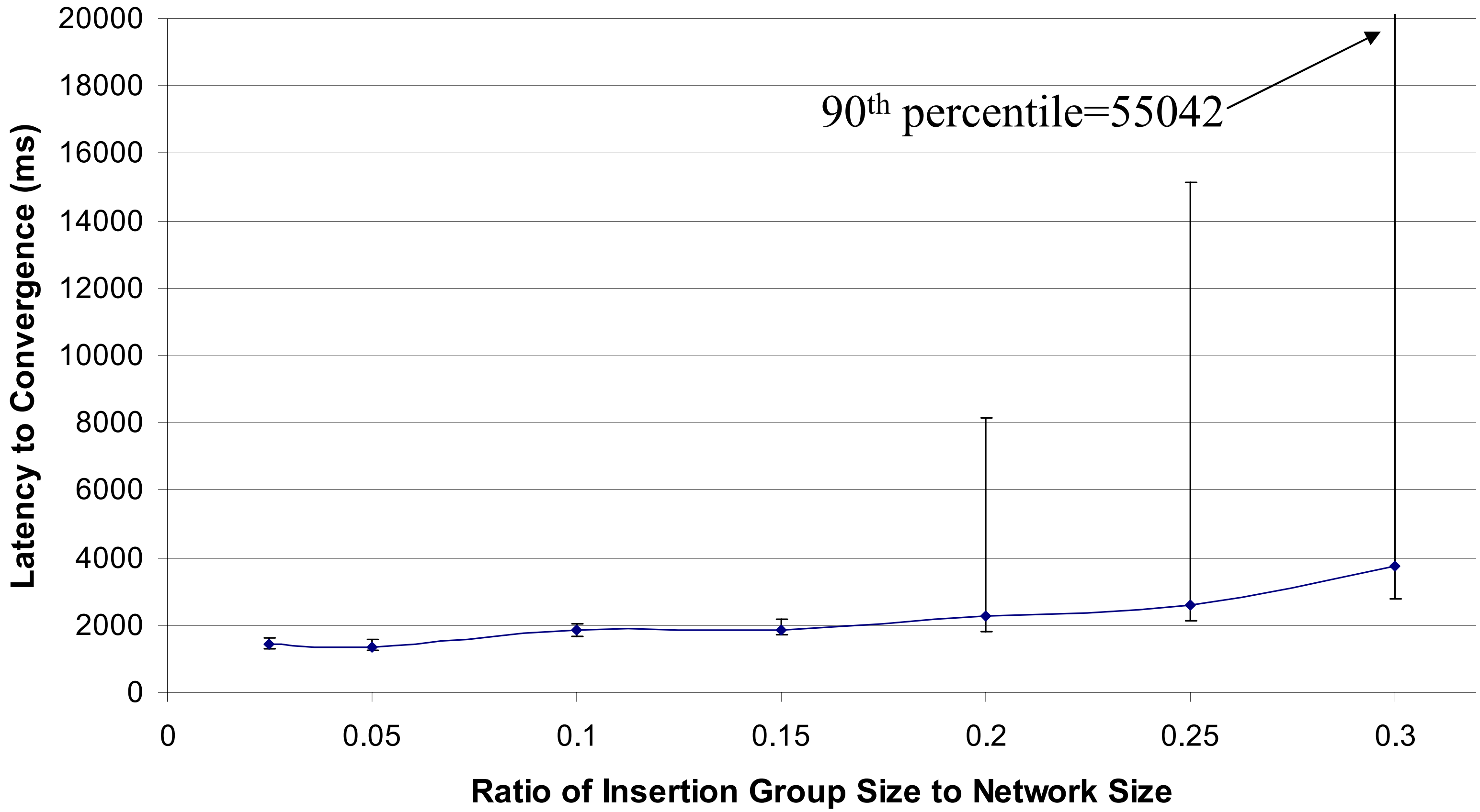
- Latency to dynamically insert a node into an existing Tapestry, as function of size of existing Tapestry
- Humps due to expected filling of each routing level

Bandwidth to Insert Node



- Cost in bandwidth of dynamically inserting a node into the Tapestry, amortized for each node in network
- Per node bandwidth decreases with size of network

Parallel Insertion Latency



- Latency to dynamically insert nodes in unison into an existing Tapestry of 200
- Shown as function of insertion group size / network size

Talk Outline

- Introduction
- Architecture
- Deployment Evaluation
- Fault-tolerant Routing
  - Tunneling through scalable overlays
  - Example using Tapestry

Adaptive and Resilient Routing

- Goals
  - Reachability as a service
  - Agility / adaptability in routing
  - Scalable deployment
  - Useful for all client endpoints

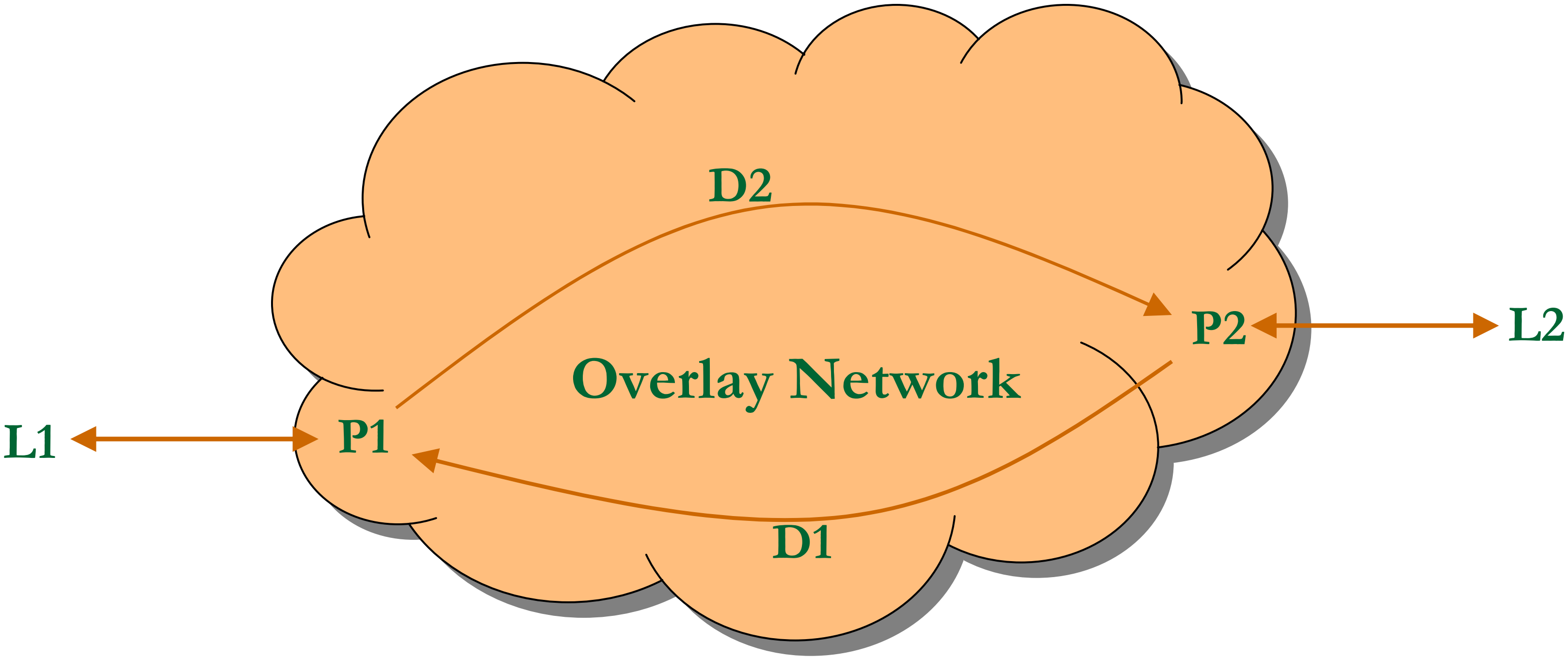
Existing Redundancy in DOLR/DHTs

- Fault-detection via soft-state beacons
  - Periodically sent to each node in routing table
    - Scales logarithmically with size of network
  - Worst case overhead: 2^{40} nodes, 160b ID → 20 hex
    1 beacon/sec, 100B each = 240 kbps
    can minimize B/W w/ better techniques (Hakim, Shelley)
- Precomputed backup routes
  - Intermediate hops in overlay path are flexible
    Keep list of backups for outgoing hops
    (e.g. 3 node pointers for each route entry in Tapestry)
  - Maintain backups using node membership algorithms
    (no additional overhead)

Bootstrapping Non-overlay Endpoints

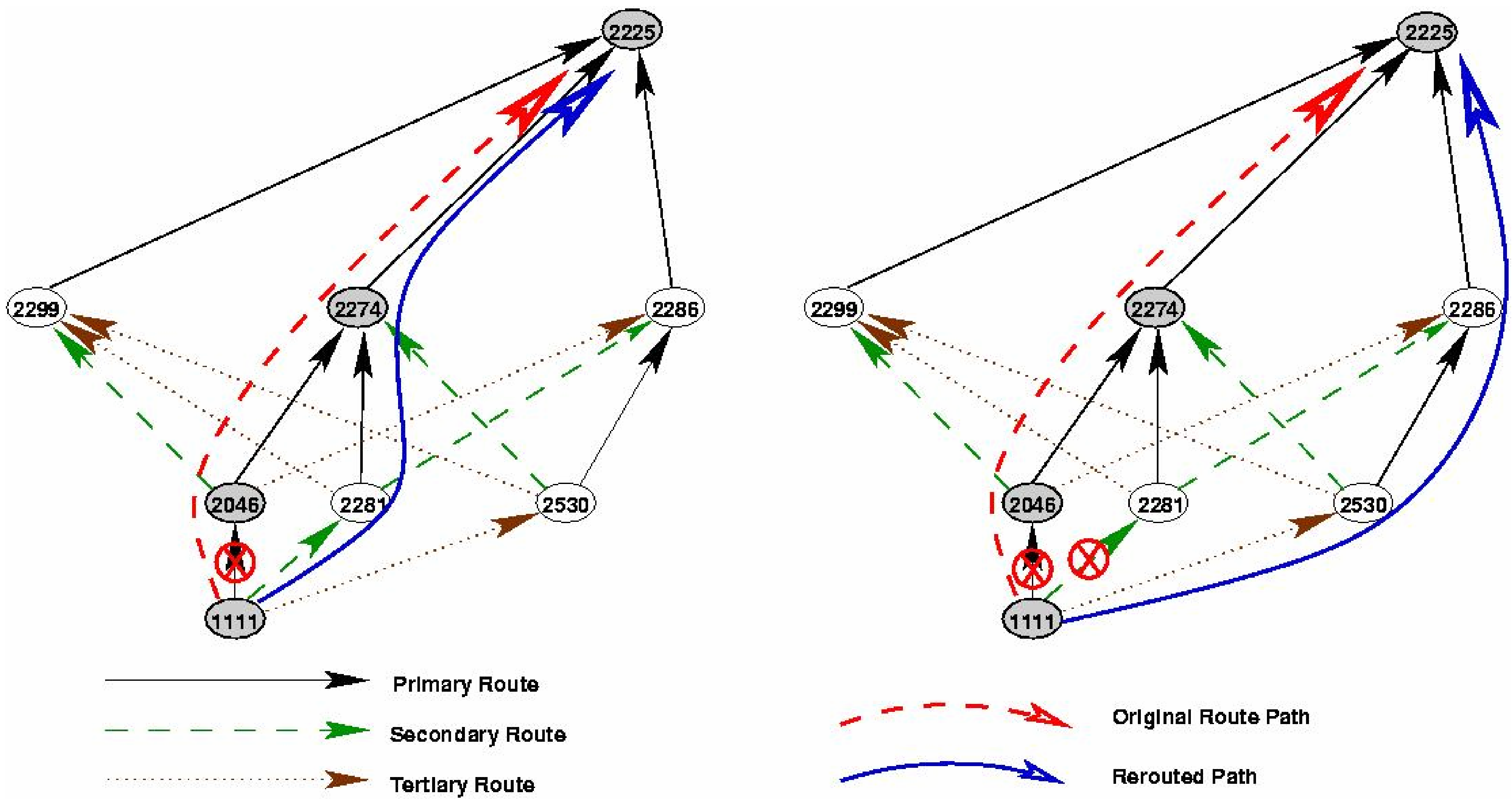
- Goal
  - Allow non-overlay nodes to benefit
  - Endpoints communicate via overlay proxies
- Example: legacy nodes L_1, L_2
  - L_i registers w/ nearby overlay proxy P_i
  - P_i assigns L_i a proxy name D_i *s.t.* D_i is the closest possible unique name to P_i (e.g. start w/ P_i, increment for each node)
  - L_i and L_2 exchange new proxy names
  - messages route to nodes using proxy names

Tunneling through an Overlay

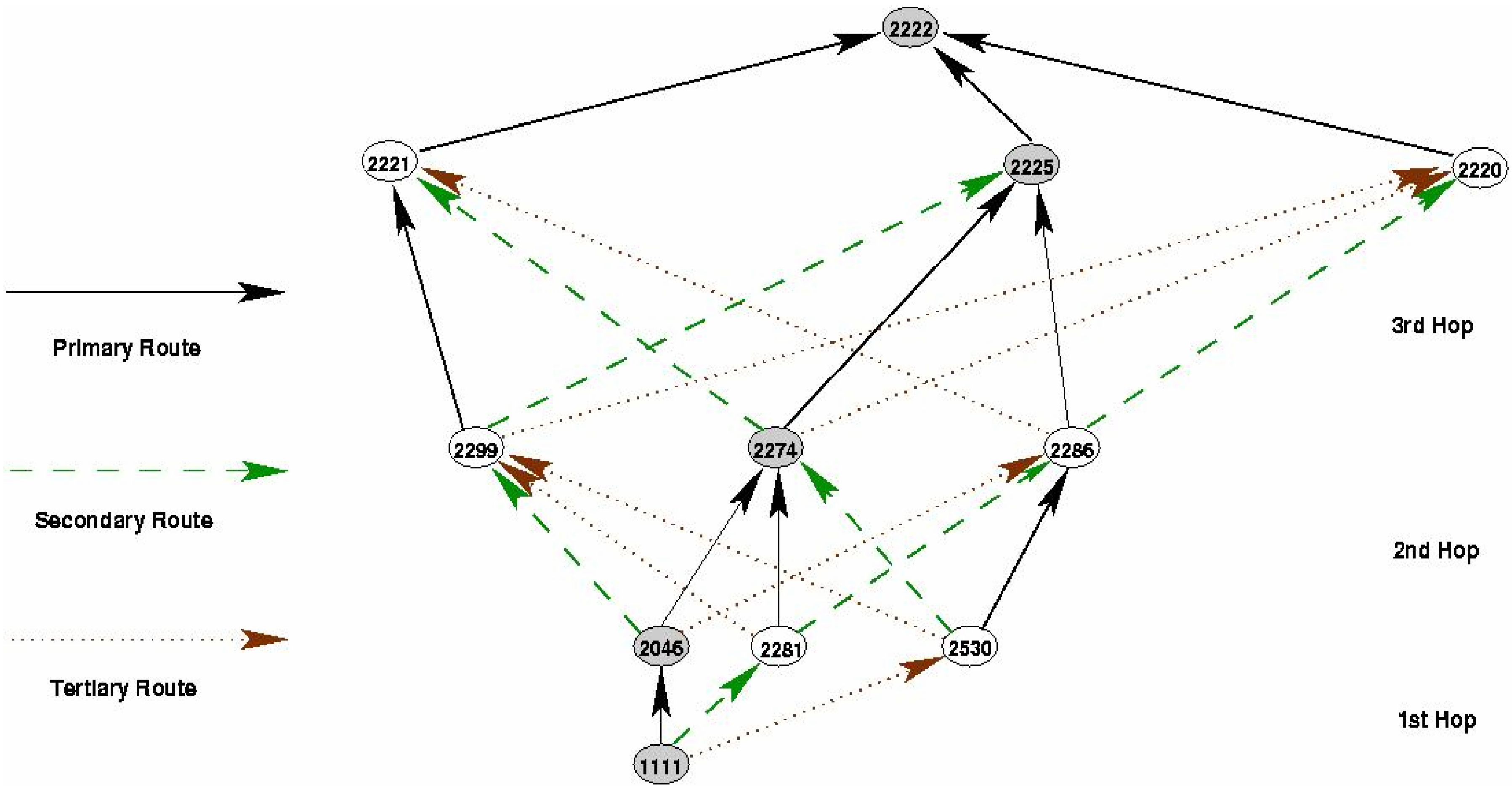


- L1 registers with P1 as document D1
- L2 registers with P2 as document D2
- Traffic tunnels through overlay via proxies

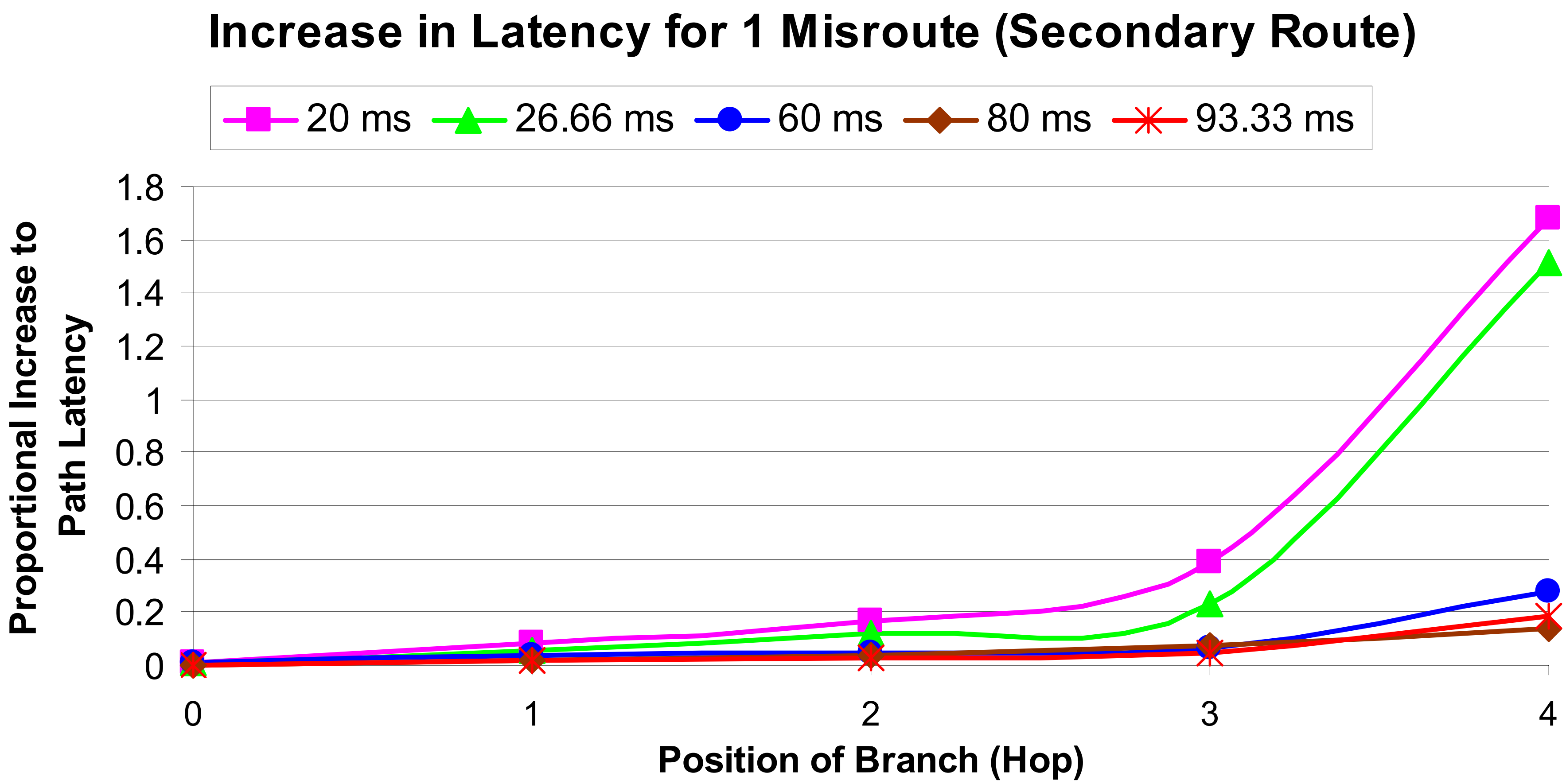
Failure Avoidance in Tapestry



Routing Convergence



Bandwidth Overhead for Misroute



- Status: under deployment on PlanetLab

For more information …

Tapestry and related projects (and these slides):

http://www.cs.berkeley.edu/~ravenben/tapestry

OceanStore:

http://oceanstore.cs.berkeley.edu

Related papers:

http://oceanstore.cs.berkeley.edu/publications

http://www.cs.berkeley.edu/~ravenben/publications

ravenben@eecs.berkeley.edu