Structured peer-to-peer overlays: A new platform for distributed systems?

Peter Druschel
Rice University

Group members:

Anwis Das, Andreas Haeberlen, Sitaram Iyer, Alan Mislove, Animesh Nandi, Tsuen Wan “Johnny” Ngan, Ansley Post, Atul Singh, Dan Wallach

Collaborators:

Miguel Castro, Anne-Marie Kermarrec, Antony Rowstron

Microsoft Research, Cambridge

Y. Charlie Hu, *Purdue University*

IRIS: Infrastructure for Resilient Internet Systems

NSF Large ITR project, *http://iris.lcs.mit.edu*

Institutions:

ICIR, MIT, NYU, Rice, UC Berkeley

PIs:

Hari Balakrishnan, Peter Druschel , Joe Hellerstein, Frans Kaashoek, David Karger, Robert Karp, John Kubiatowicz, Barbara Liskov, David Mazières, Robert Morris, Scott Shenker, Ion Stoica

Additional funding from Texas TATP, Intel and Microsoft

Outline

- Background: Peer-to-peer (P2P)
- Structured p2p overlays: Pastry
- Pastry proximity-aware routing
- Sharing state: Distributed hash tables
- Coordination: Cooperative group communication
- Security and Incentives
- Conclusions

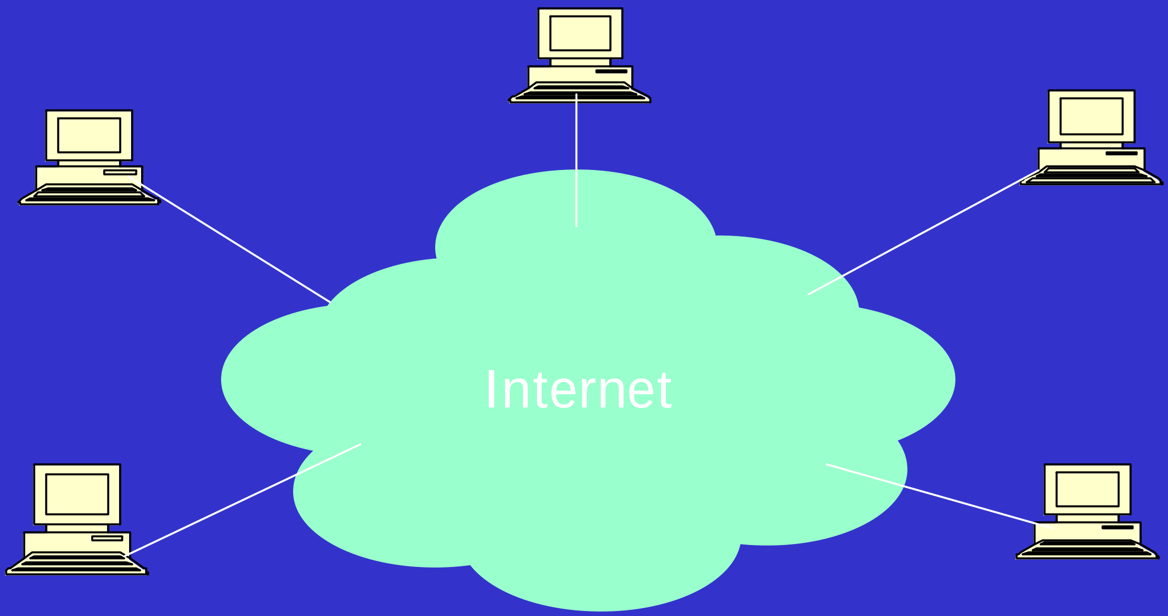
P2P: an exciting social development

- Internet users cooperating to share, for example, music files
  - Napster, Gnutella, Morpheus, KaZaA, etc.
- Lots of attention from the popular press

  "The ultimate form of democracy on the Internet"

  "The ultimate threat to copy-right protection on the Internet"
- Technology has applications far beyond file sharing
- Many vendors have launched P2P efforts

What is P2P technology?



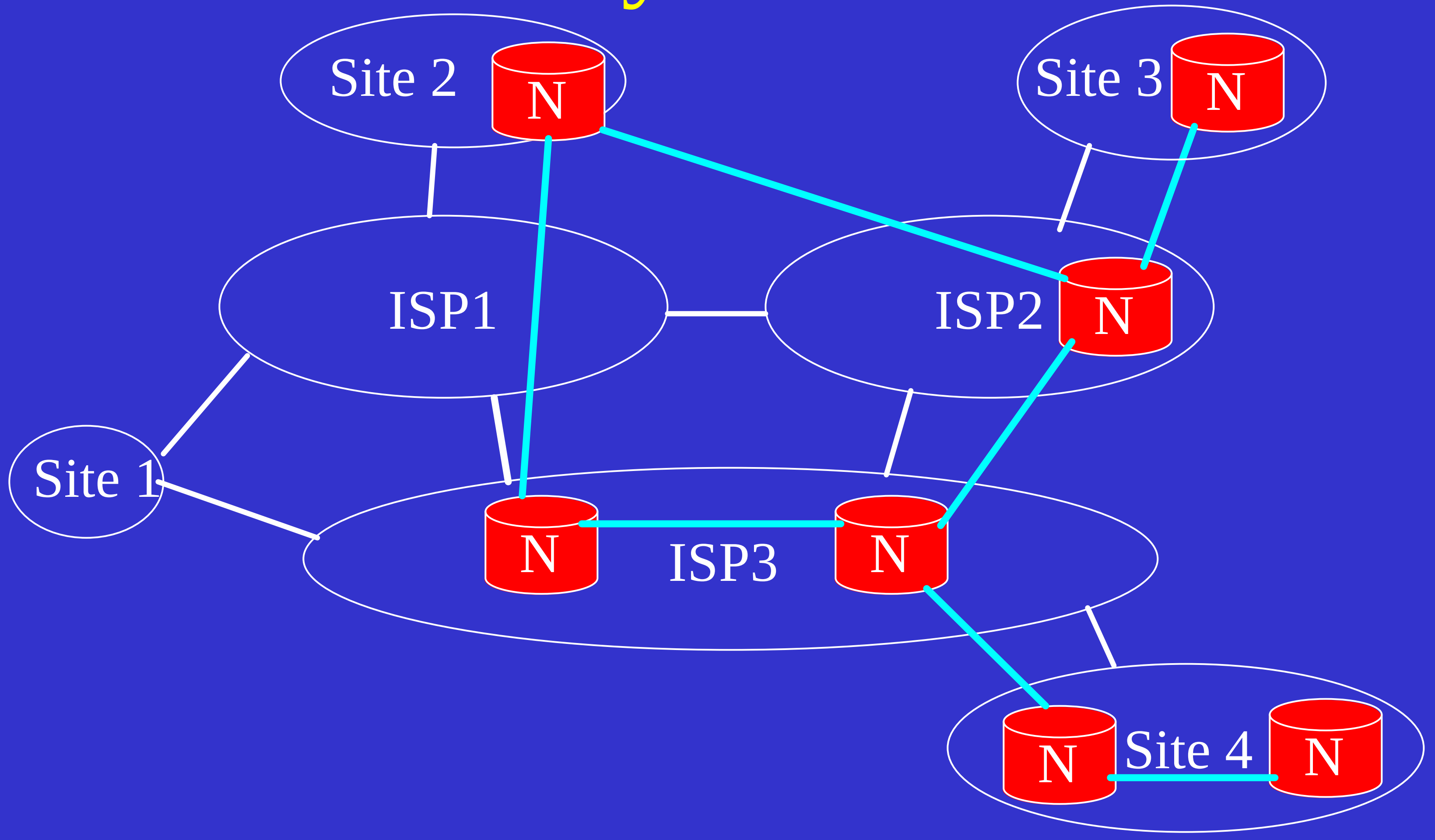
- A distributed system architecture:
  - No centralized control
  - Self-organizing
- Participants share bandwidth, storage, computation
- Typically many nodes, but unreliable, heterogeneous and potentially untrusted

Why p2p?

- Cooperative, shared infrastructure
- Aggregated storage, network and compute resources
- Incremental (“organic”) growth and scaling
- Resource diversity (architecture, location, ownership, rule of law): tolerate faults, attacks

But: realizing this potential presents many technical challenges

Overlay networks



P2P systems are self-organizing overlay networks without central control

P2p overlays

Unstructured overlays (Gnutella,Freenet)

- Random overlay graph construction (cheap)
- Unreliable/inefficient searching

Structured overlays (CAN,Chord,Pastry,Tapestry)

- Overlay conforms to a specific graph structure
- Reliable, efficient searching
- Somewhat higher overlay construction/maintenance overhead

Outline

- Background: Peer-to-peer overlays
- Structured p2p overlays: Pastry
- Pastry proximity-aware routing
- Sharing state: Distributed hash tables
- Coordination: Cooperative group communication
- Security and Incentives
- Conclusions

Structured p2p overlays

Overlay conforms to a specific graph structure

- Reliable, efficient searching

Overlay dynamically maps objects to live nodes, s.th.

- Each object is assigned a unique live node
- The number of objects per node is balanced

One primitive:

route(M, X): route message *M* to the live node currently associated with object key *X*

Structured overlays support many applications

Enhanced Internet services:

- Multicast/Anycast/Mobility [i3, Scribe]
- Overlay QoS
- Naming systems [INS, SFR, NLS]

Co-operative services:

- Shared storage [CFS, OceanStore, PAST, Ivy]
- Content distribution [Squirrel, SplitStream]
- Query and indexing [PIER]
- Messaging [POST]
- Backup store [HiveNet, Pastiche, PAST]
- Web archiver [Herodotus]

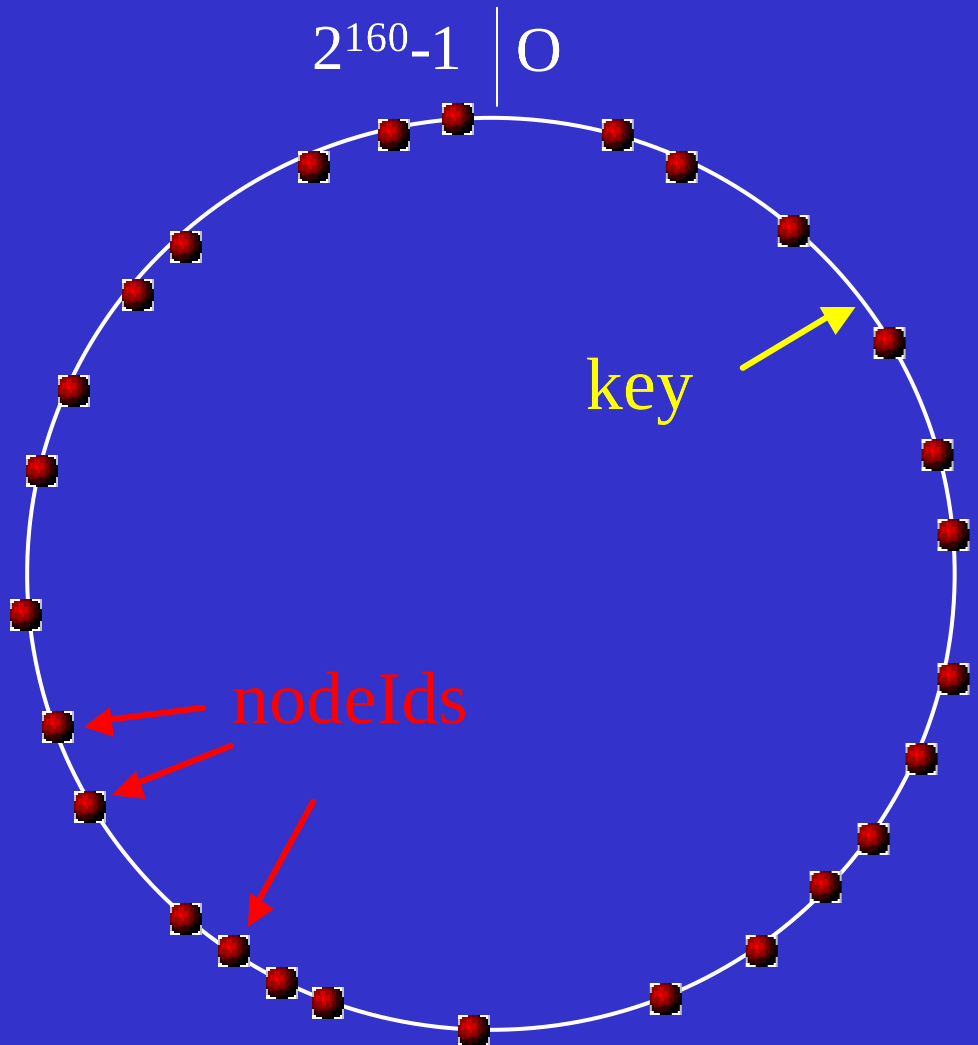
Research challenges

1. Scalable lookup
2. Balancing load
3. Handling failures
4. Network-awareness for performance
5. Robustness with untrusted participants
6. Programming abstraction
7. Heterogeneity
8. Coping with systems in flux
9. Anonymity/Anti-censorship

} this talk

Goal: simple, provably-good algorithms

Pastry: Identifier space



Consistent hashing [*Karger et al. '97*]

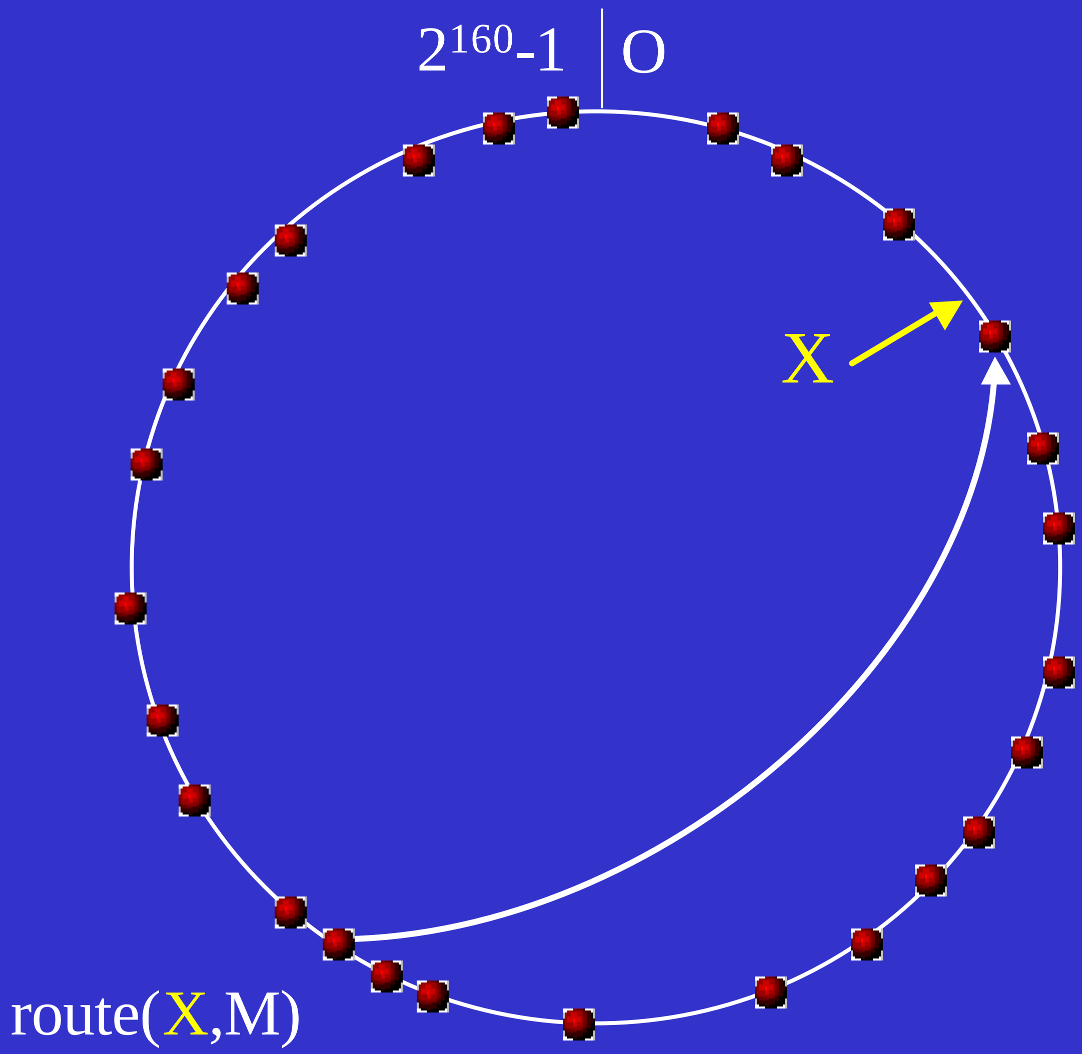
160 bit circular id space

nodeIds (uniform random)

keys (uniform random)

Each key is mapped to the live node with numerically closest nodeId

Pastry: Routing



Msg with key *X* is routed to live node with nodeId closest to *X*

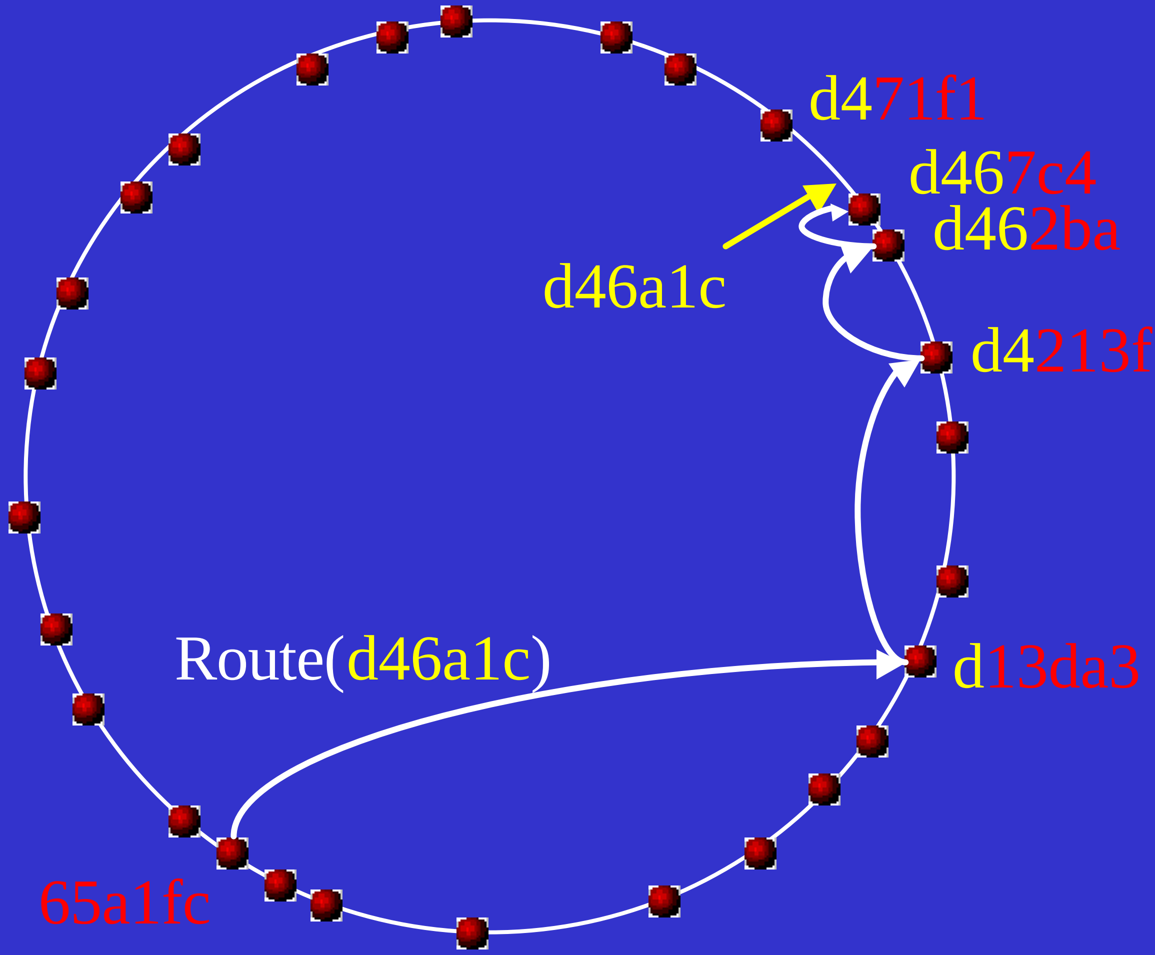
Problem:
complete routing table not feasible

Overlay routing

Idea: Trade forwarding hops for routing table size

- **CAN:** $N^{1/d}$ hops, d neighbors
- **Chord:** $\frac{1}{2} \log_2 N$ hops, $O(\log N)$ neighbors
- **Pastry, Tapestry, Kademlia:** $\log_b N$ hops, $O(\log N)$ neighbors (b is normally 16).
- **Viceroy:** $O(\log N)$ hops, k neighbors

Pastry: Prefix-based routing



Properties

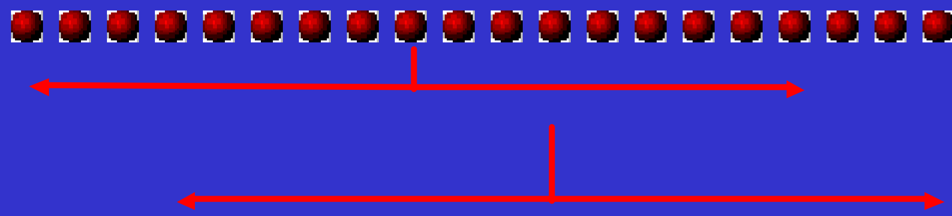
- $\log_{16} N$ steps
- $O(\log N)$ state

Pastry: Routing table (# 65a1fc*x*)

Row 0	*0x*	*1x*	*2x*	*3x*	*4x*	*5x*		*7x*	*8x*	*9x*	*ax*	*bx*	*cx*	*dx*	*ex*	*fx*
Row 1	*60x*	*61x*	*62x*	*63x*	*64x*		*66x*	*67x*	*68x*	*69x*	*6ax*	*6bx*	*6cx*	*6dx*	*6ex*	*6fx*
Row 2	*650x*	*651x*	*652x*	*653x*	*654x*	*655x*	*656x*	*657x*	*658x*	*659x*		*65bx*	*65cx*	*65dx*	*65ex*	*65fx*
Row 3	*65a0x*		*65a2x*	*65a3x*	*65a4x*	*65a5x*	*65a6x*	*65a7x*	*65a8x*	*65a9x*	*65aax*	*65abx*	*65acx*	*65adx*	*65aex*	*65afx*

log_{16} *N* rows

Pastry: Leaf sets



Each node maintains IP addresses of the nodes with the L/2 numerically closest larger and smaller nodeIds, respectively.

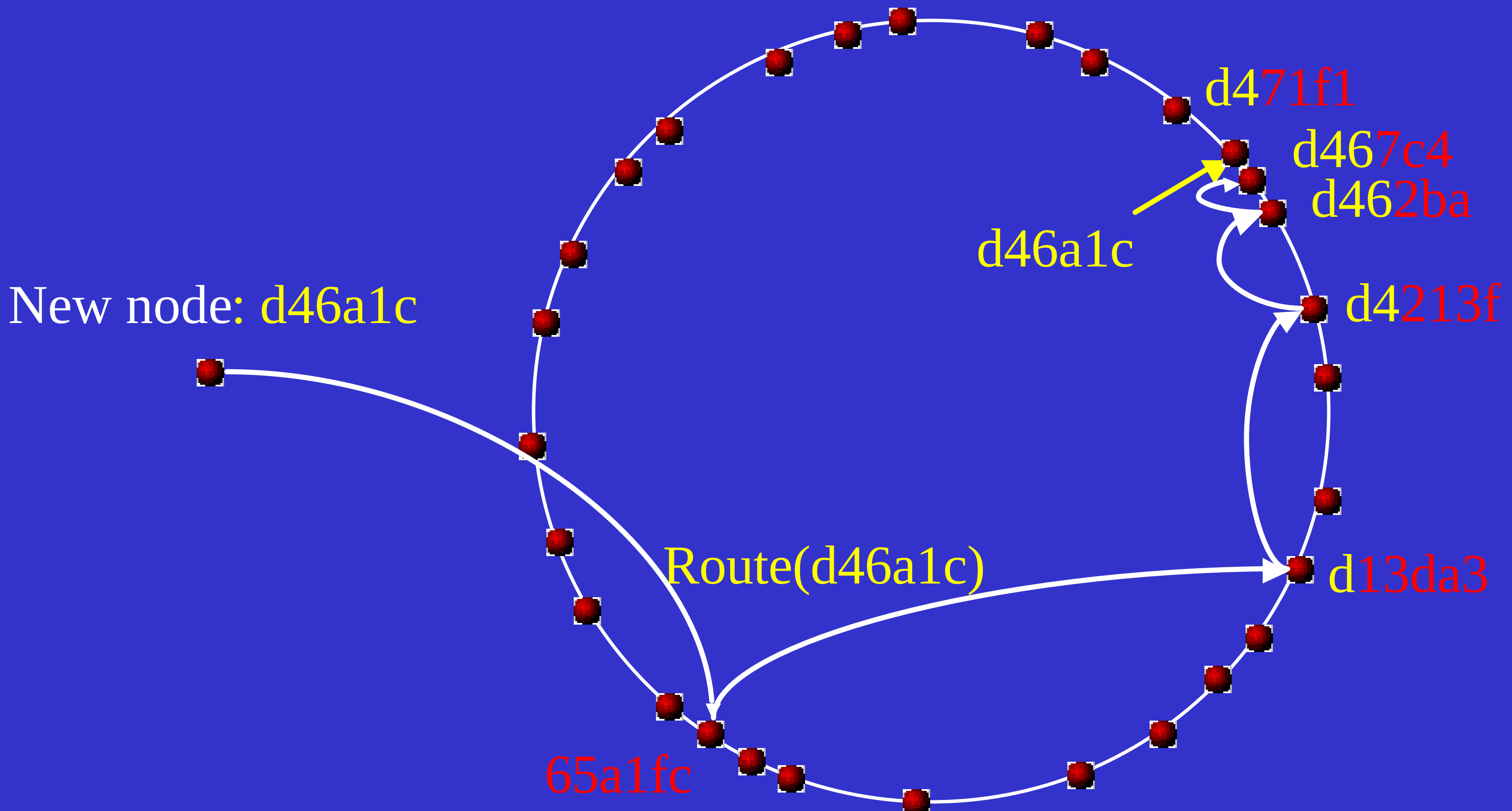
- routing efficiency/robustness
- fault detection (keep-alive)
- application-specific local coordination (e.g., replication)

Pastry: Self-organization

Initializing and maintaining node state
(overlay construction and maintenance)

- Node addition
- Node departure (failure)

Pastry: Node addition



Node departure (failure)

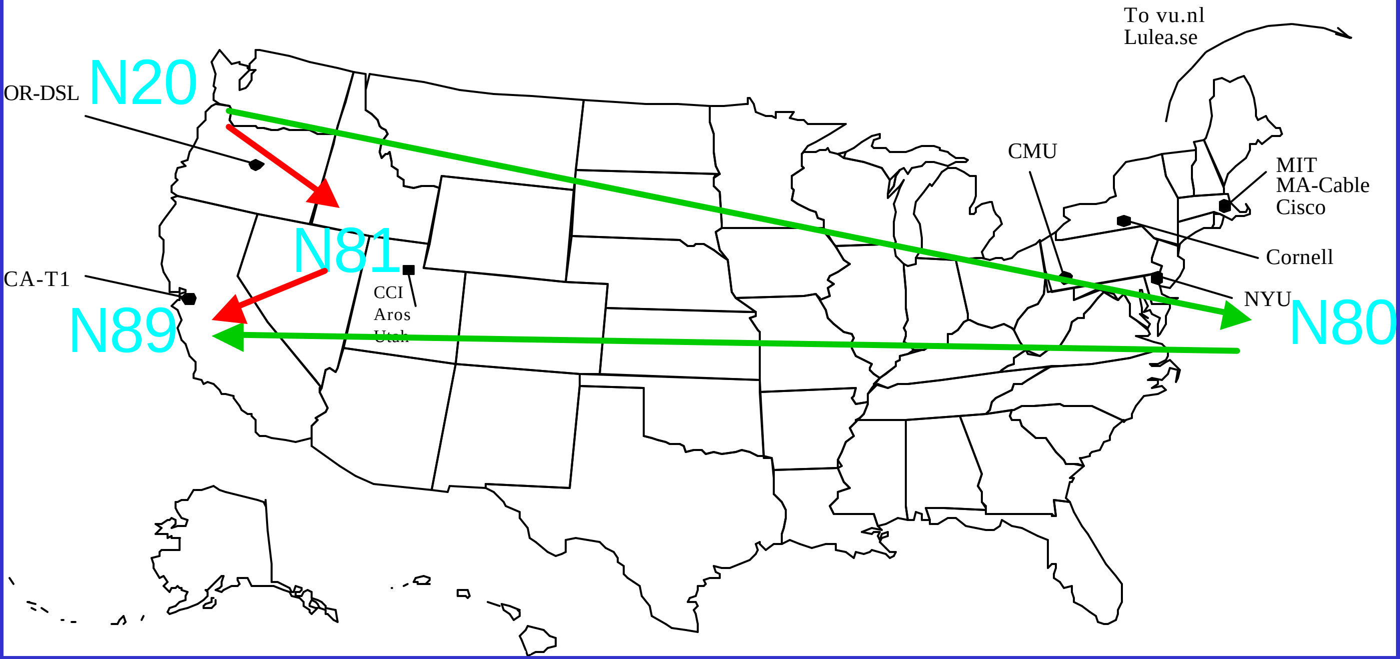
Leaf set members exchange keep-alive messages

- **Leaf set repair (eager):** request set from farthest live node in set
- **Routing table repair (lazy):** get table from peers in the same row, then higher rows

Outline

- Background: Peer-to-peer overlays
- Structured p2p overlays: Pastry
- Pastry proximity-aware routing
- Sharing state: Distributed hash tables
- Coordination: Cooperative group communication
- Security and Incentives
- Conclusions

Optimize routing to reduce latency



- Nodes close in id space, but far away in Internet
- Goal: put nodes in routing table that result in few hops and low latency

Pastry: Proximity routing

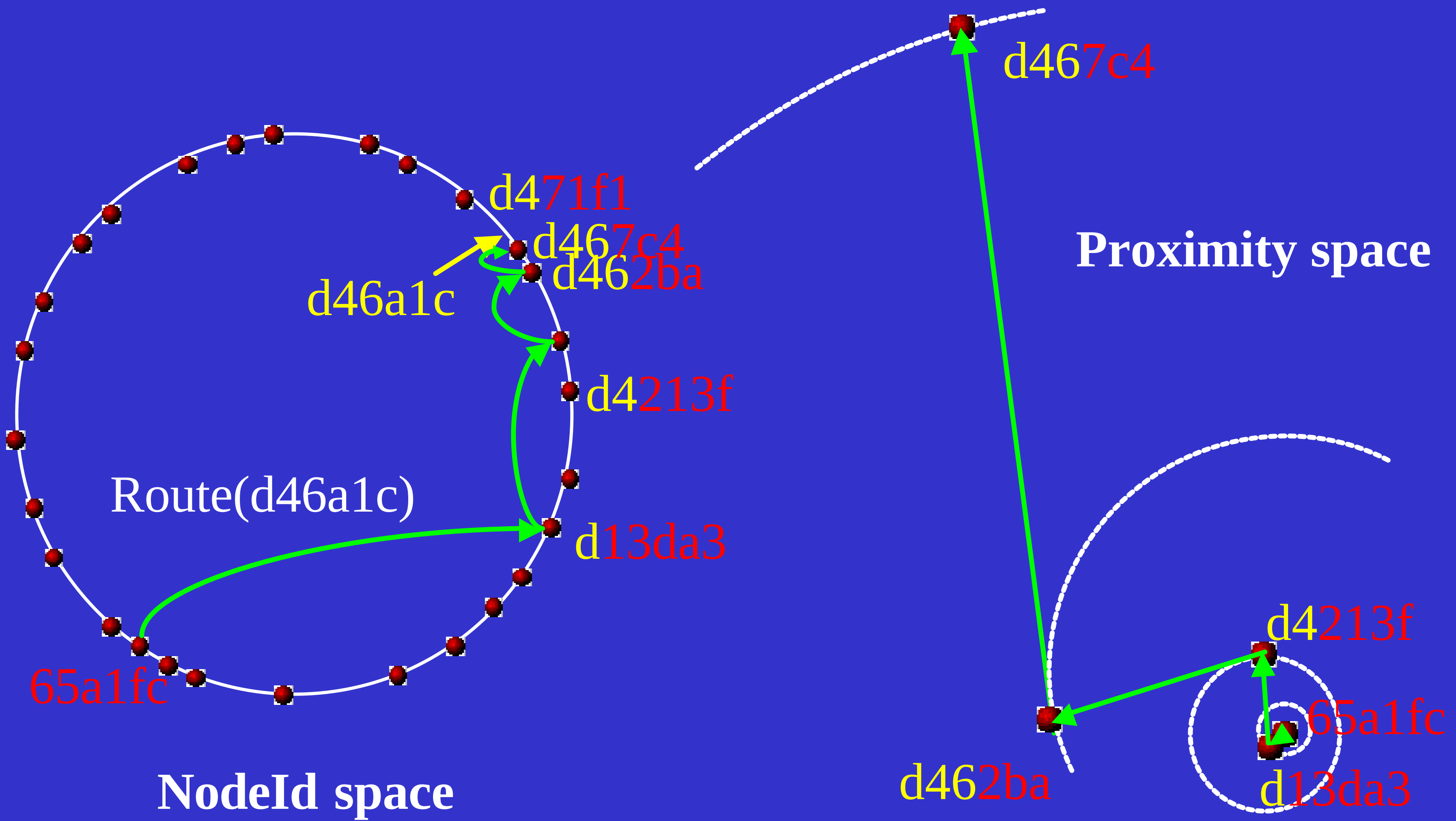
Assumptions:

- scalar proximity metric (e.g., RTT)
- a node can probe distance to any other node

Proximity invariant:

Each routing table entry refers to a node close to the local node (in the network), among all nodes with the appropriate nodeId prefix.

Pastry: Routes in proximity space

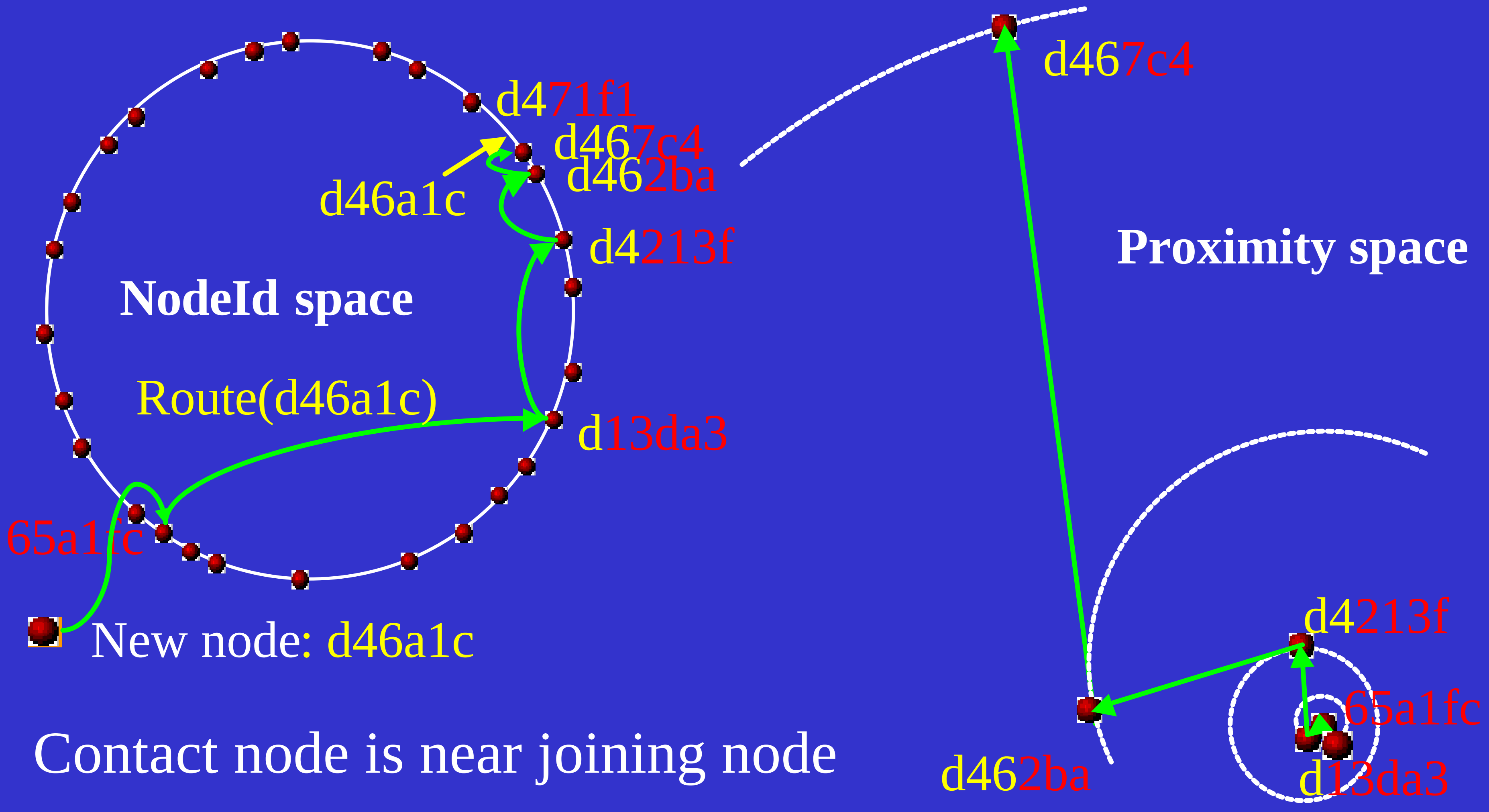


Pastry: Locality properties

1) Low-delay routes: *Average delay penalty, relative to IP, is low (1.3 - 2.2)*

2) Route convergence: *Routes of messages sent by nearby nodes with same keys converge at a node near the source nodes*

Pastry: Node addition

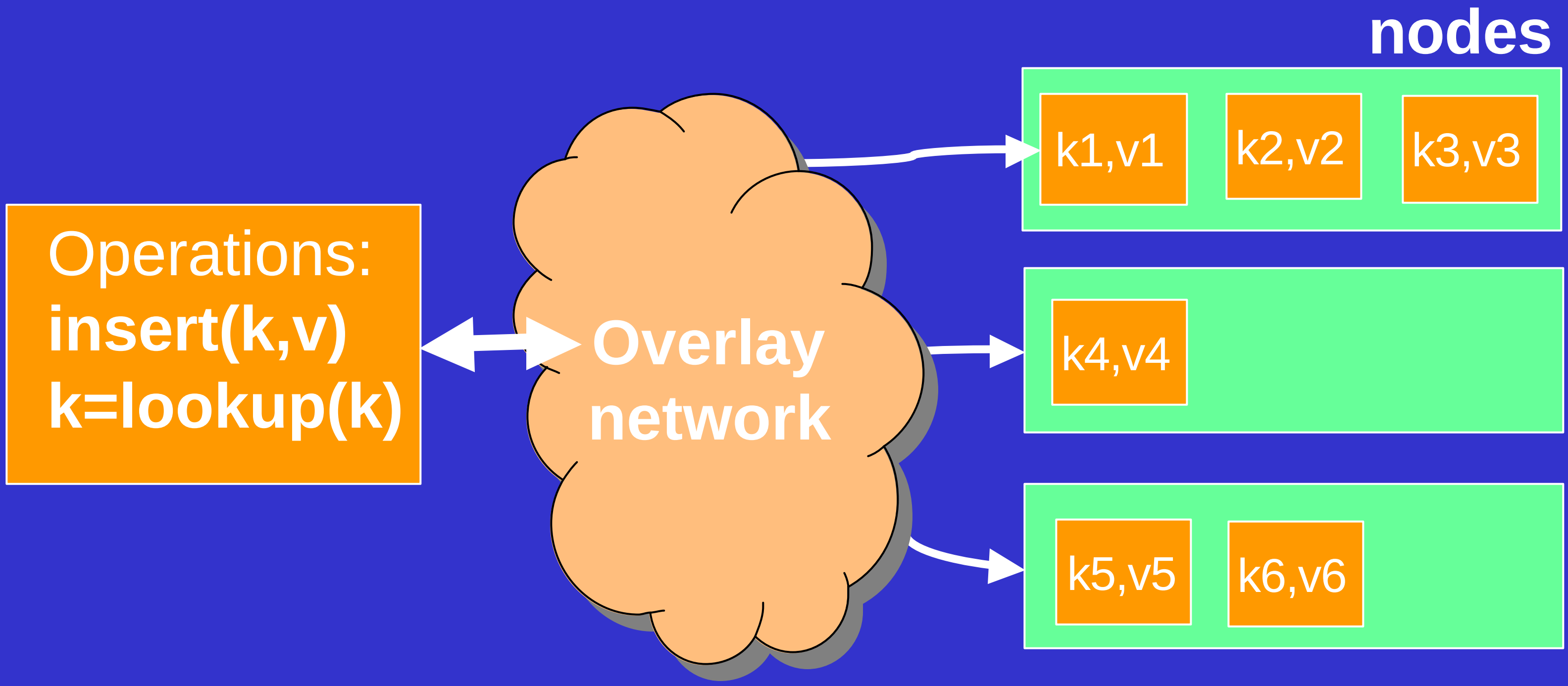


Contact node is near joining node

Outline

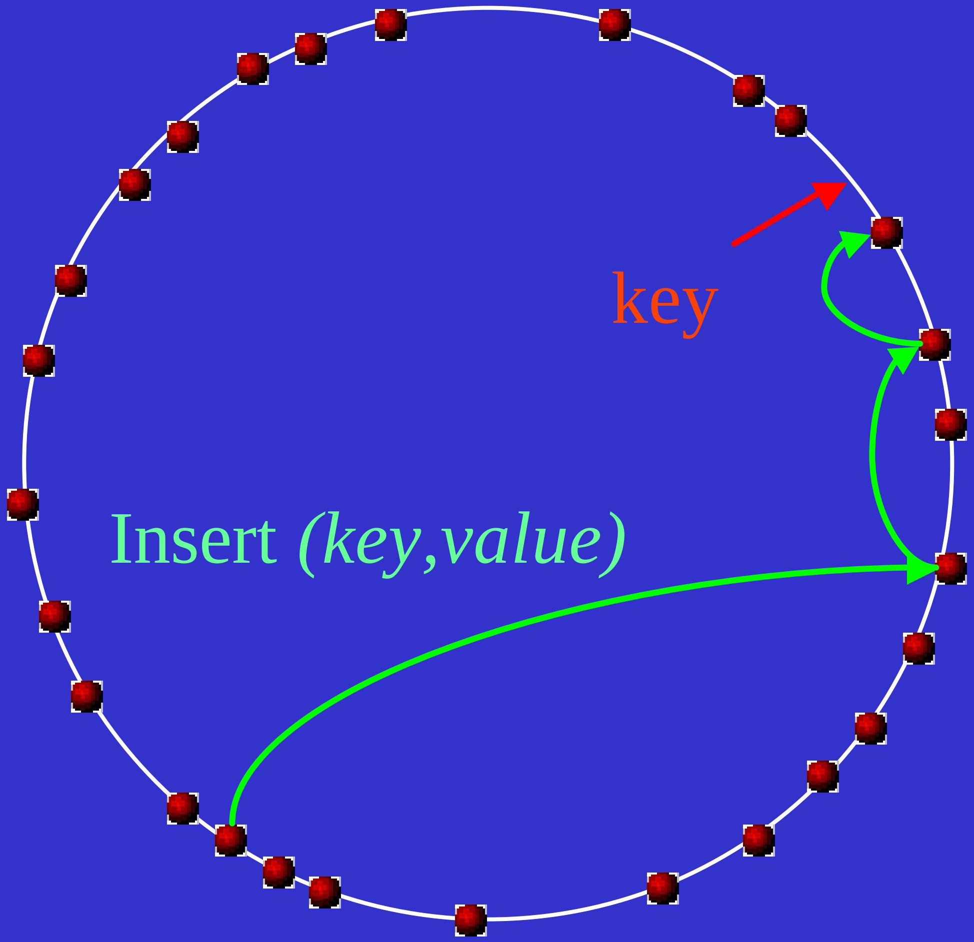
- Background: Peer-to-peer overlays
- Structured p2p overlays: Pastry
- Pastry proximity-aware routing
- Sharing state: Distributed hash tables
- Coordination: Cooperative group communication
- Applications
- Conclusions

Distributed Hash Table (DHT)

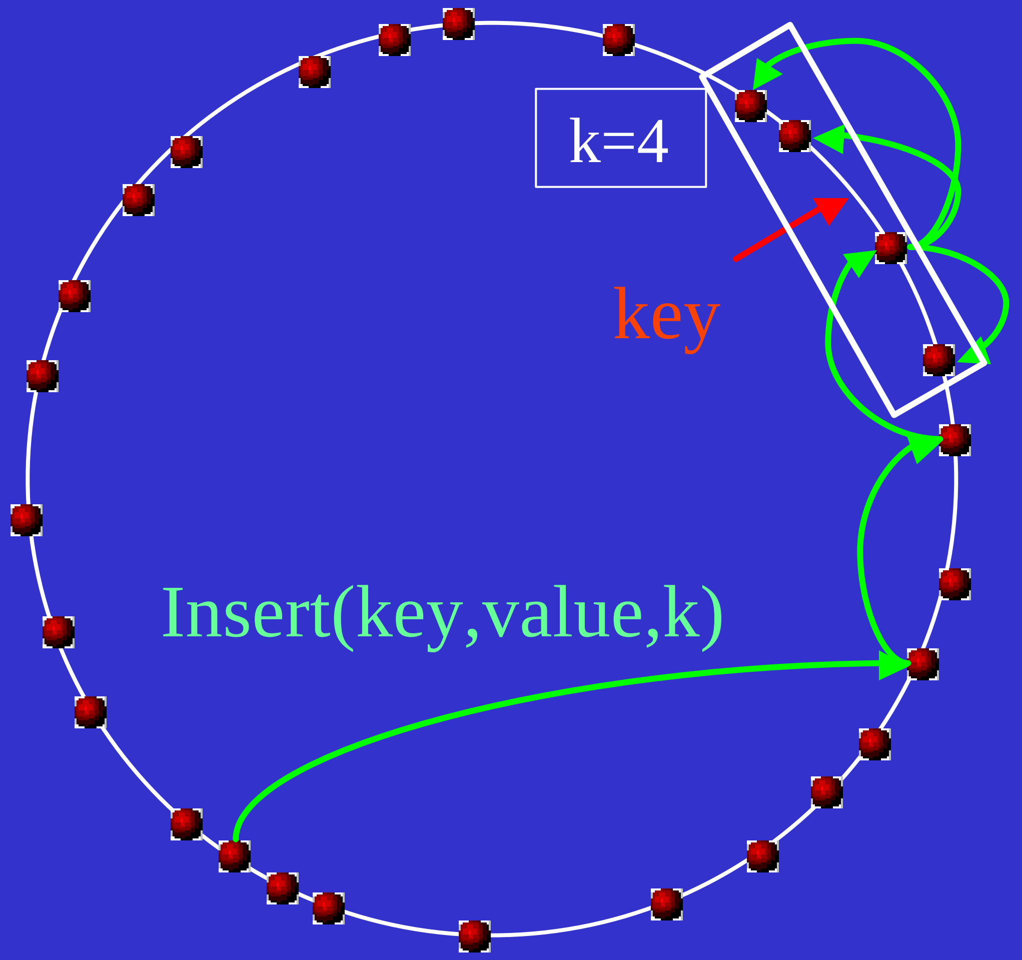


- Structured overlay maps keys to nodes
- Decentralized and self-organizing
- Scalable, robust

DHT: insertion

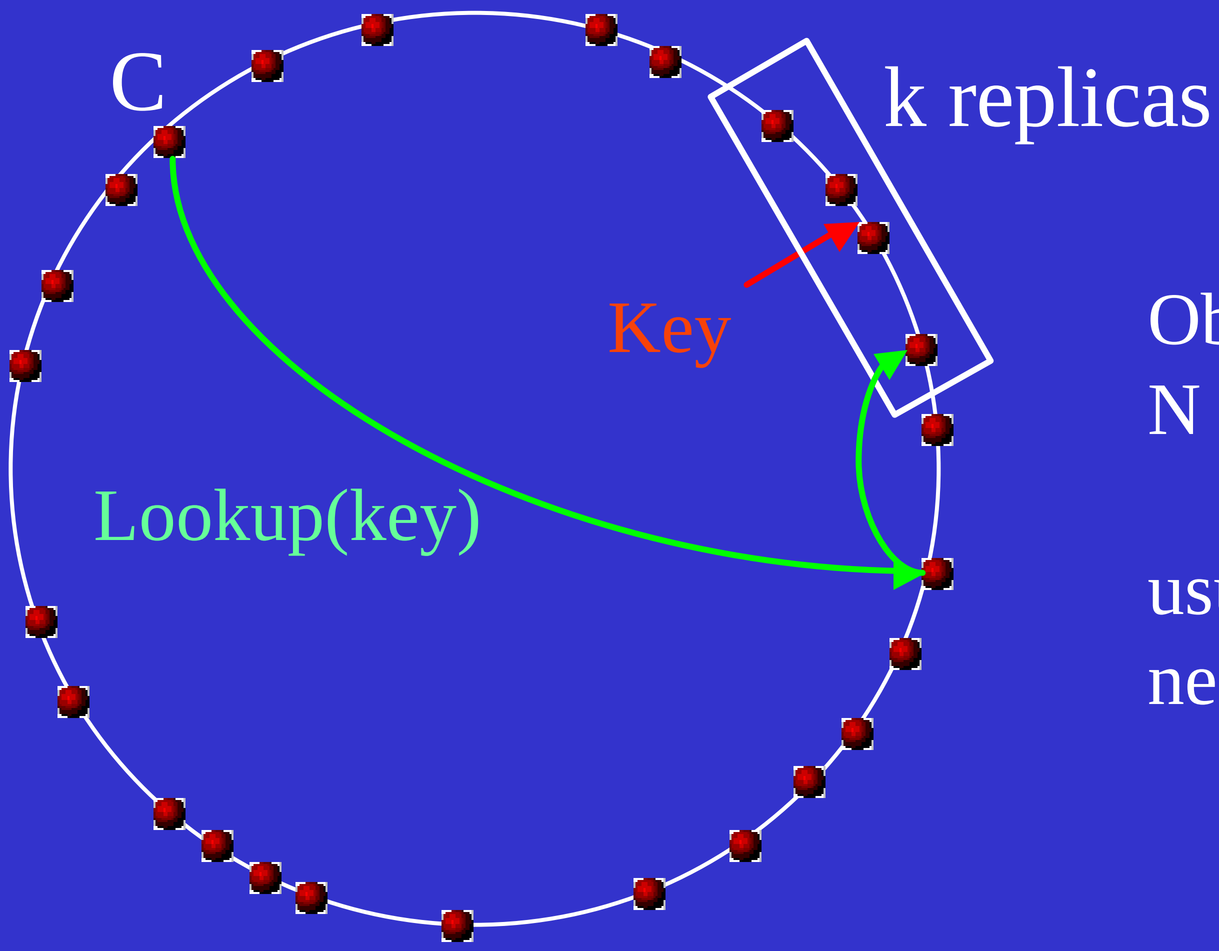


DHT: Replication



Storage Invariant:
Tuple replicas are stored on *k* nodes with *nodeIds* closest to *key*

DHT: Lookup



Object located in $\log_{16} N$ steps (expected)

usually locates replica nearest client C

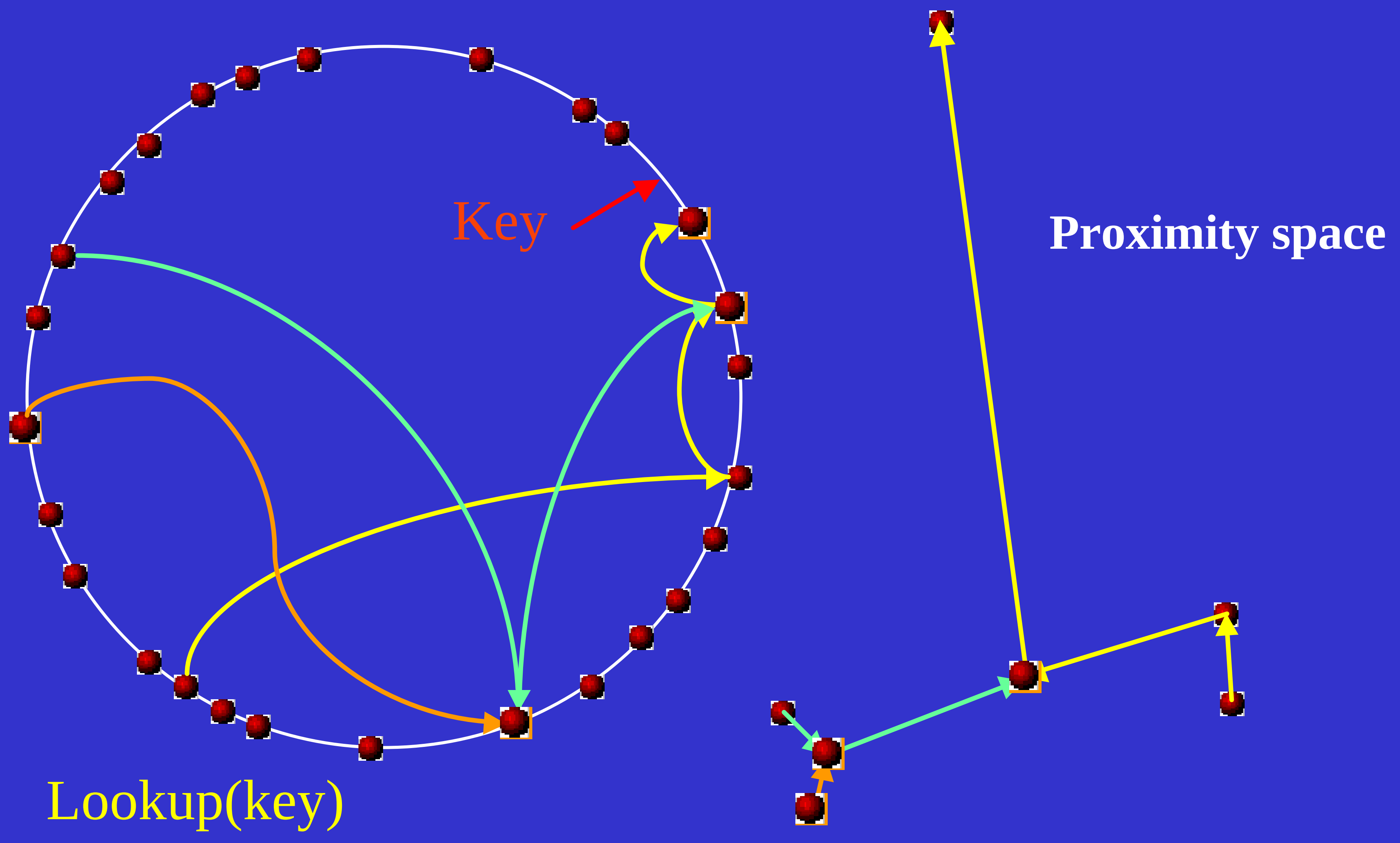
DHT: Dynamic caching

- Nodes cache tuples in the unused portion of their allocated disk space
- Files cached on nodes along the route of lookup and insert messages

Goals:

- maximize query xput for popular tuples
- balance query load
- improve client latency

DHT: Dynamic caching



Outline

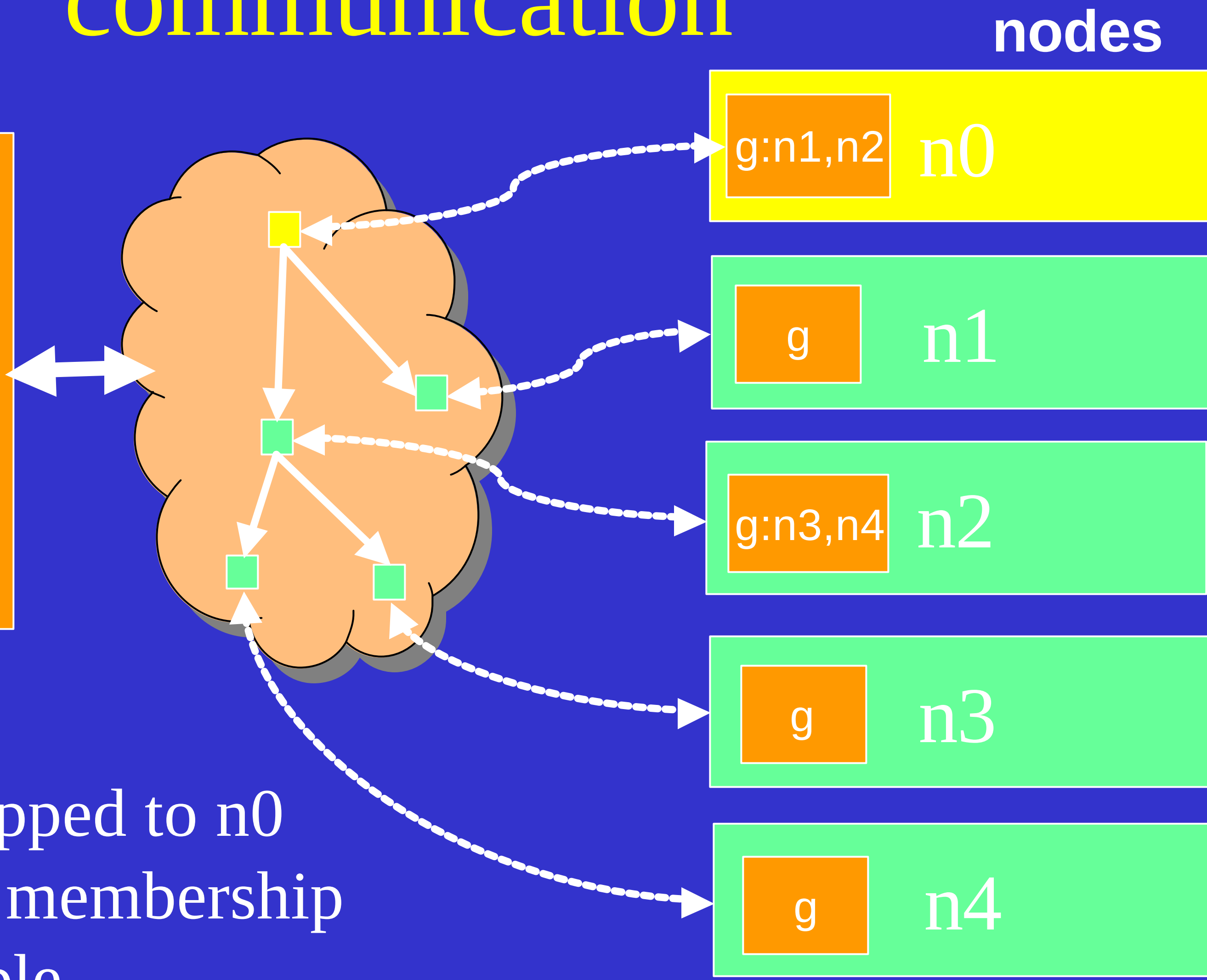
- Background: Peer-to-peer overlays
- Structured p2p overlays: Pastry
- Pastry proximity-aware routing
- Storing state: Distributed hash tables
- Coordination: Cooperative group communication
- Security and incentives
- Conclusions

Coordination: Cooperative group communication

- Scribe: Tree-based group management
- Multicast, anycast, manycast primitives
- Scalable: large numbers of groups, members, wide range of members/group, dynamic membership

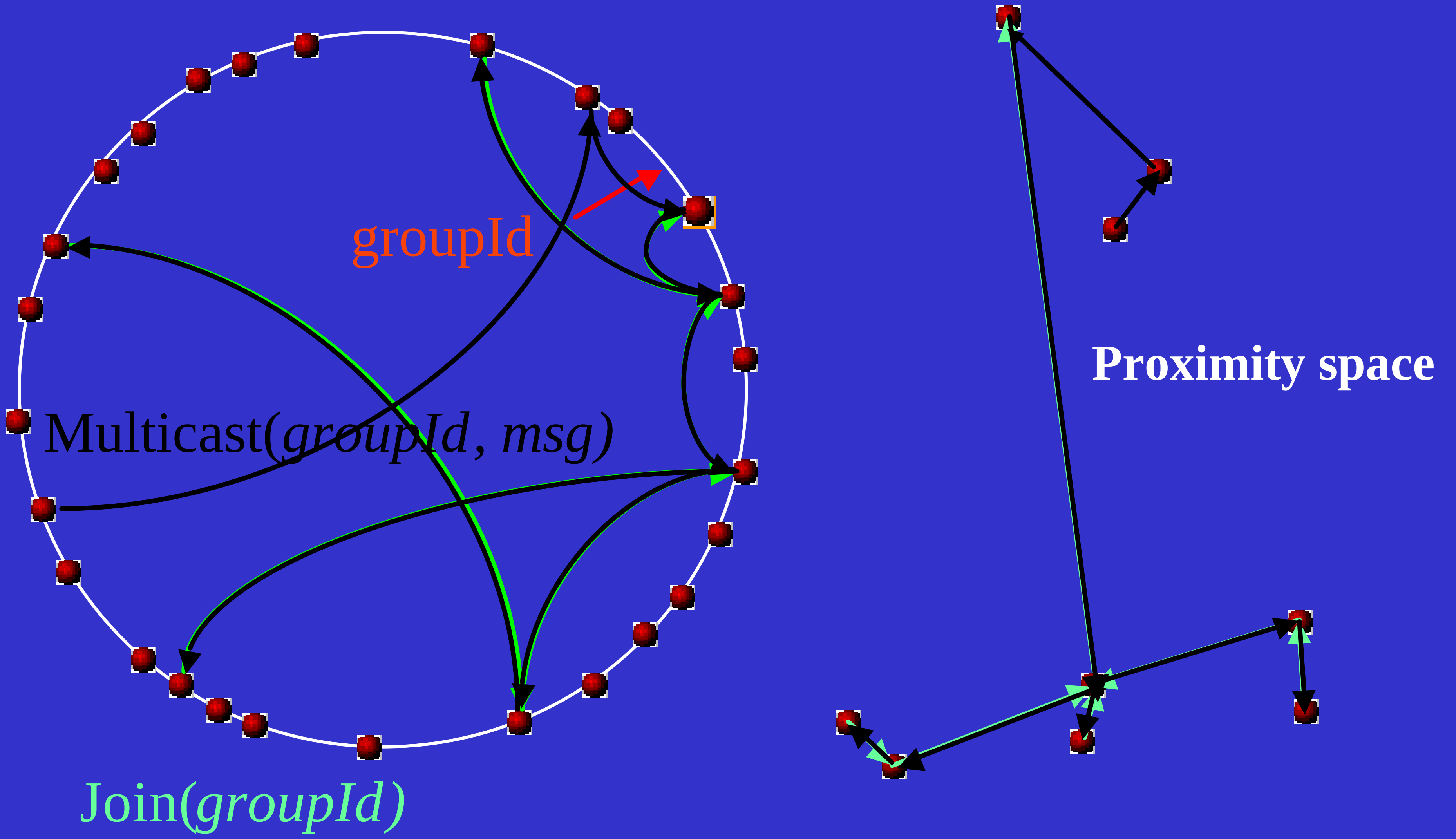
Cooperative group communication

Operations:
create(g)
join(g)
leave(g)
multicast(g,m)
anycast(g,m)



- groupId *g* mapped to n0
- decentralized membership
- robust, scalable

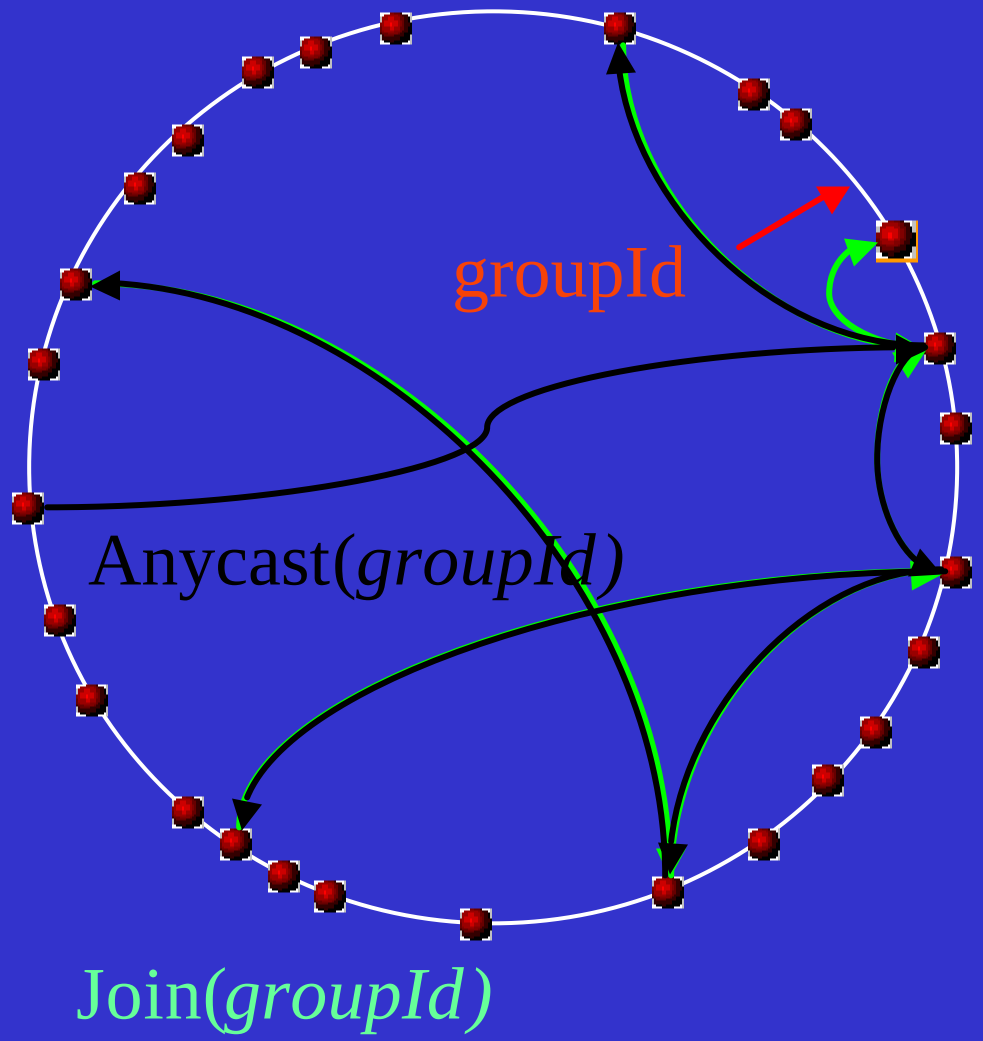
Scribe

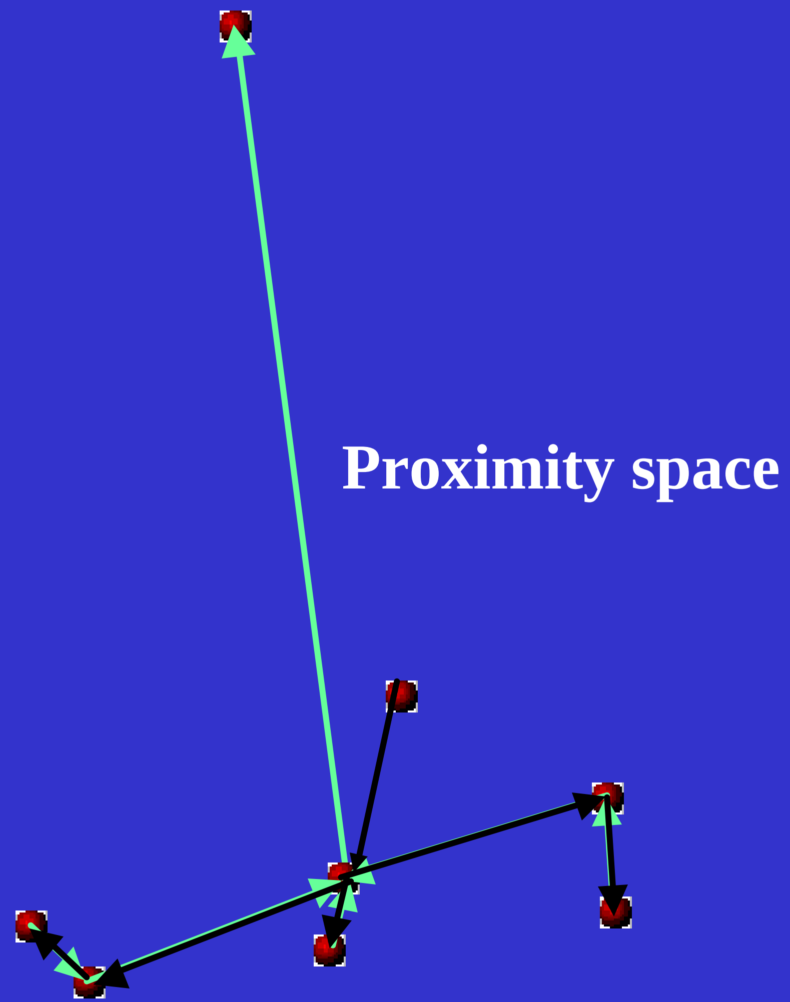

groupId
Multicast(*groupId*, *msg*)
Join(*groupId*)
Proximity space

Scribe multicast: Results

- Experimental setup
  - Georgia Tech Transit-Stub model
  - 100,000 nodes randomly selected out of .5M
  - Zipf-like subscription distribution, 1500 topics
- Delay penalty: ~1.7 (relative to IP multicast)
- Link stress: Mean 2.4 versus .7 with IP multicast

Scribe: Anycast





Scribe: Anycast

- Supports highly dynamic groups
- Suitable for decentralized resource discovery (can add predicate during DFS)
- Results (100k nodes/.5M network):
  - Join: 4.1 msgs (empty group); avg 3.5 msgs (2,500 members)
  - 1,000 anycasts: 4.1 msg (empty group); avg 2.3 msgs (2,500 members)
  - Locality: For >90% of anycasts, <7% of members were closer than the receiver

Key-based routing (KBR) API

[IPTPS'03]

- *route(M, X):* route message *M* to node with nodeId numerically closest to *X*
- *deliver(M):* deliver message *M* to application (upcall)
- *forwarding(M, X):* message *M* is being forwarded towards key X (upcall)

Key-based routing (KBR) API

- *getNeighborSet():* obtain the current sent of neighbors in the id space.
- *getReplicaSet(X):* obtain a replicaSet suitable for an object with key *X*
- *range(r, N):* obtain ranges of keys for which node *N* is a *r*-root.
- *local-lookup(X, num):* obtain up to *num* possible next-hop nodes appropriate for a message with key *X*.

Outline

- Background: Peer-to-peer overlays
- Structured p2p overlays: Pastry
- Pastry proximity-aware routing
- Storing state: Distributed hash tables
- Notification: Cooperative group communication
- Security and Incentives
- Conclusions

Securing the overlay [*OSDI'02*]

Participating nodes can suffer byzantine faults

- Malicious participants
- Compromised nodes

Solution:

- Secure nodeId assignment
- Secure node join protocol
- Secure routing primitive
- Can tolerate up to 25% faulty nodes

Security model

Participating nodes can suffer byzantine faults

- fraction f, $0 \le f < 1$, of participating nodes may be faulty; fraction c, $1/N \le c \le f$, may collude

Assumption:

- Applications authenticate data and services in the overlay
- => attacks are limited to denial-of-service

Securing Data

- Self-authenticating data
  - Content-hash data (key = SHA-1(contents))
  - Public-key data (key=SHA-1(pub-key), content and timestamp signed with priv-key)
- Application may encrypt content for privacy
- Pastry secure routing primitive ensures
  - *k* replicas are stored on a random sample of nodes
  - a non-faulty replica can be reached eventually

Attacks

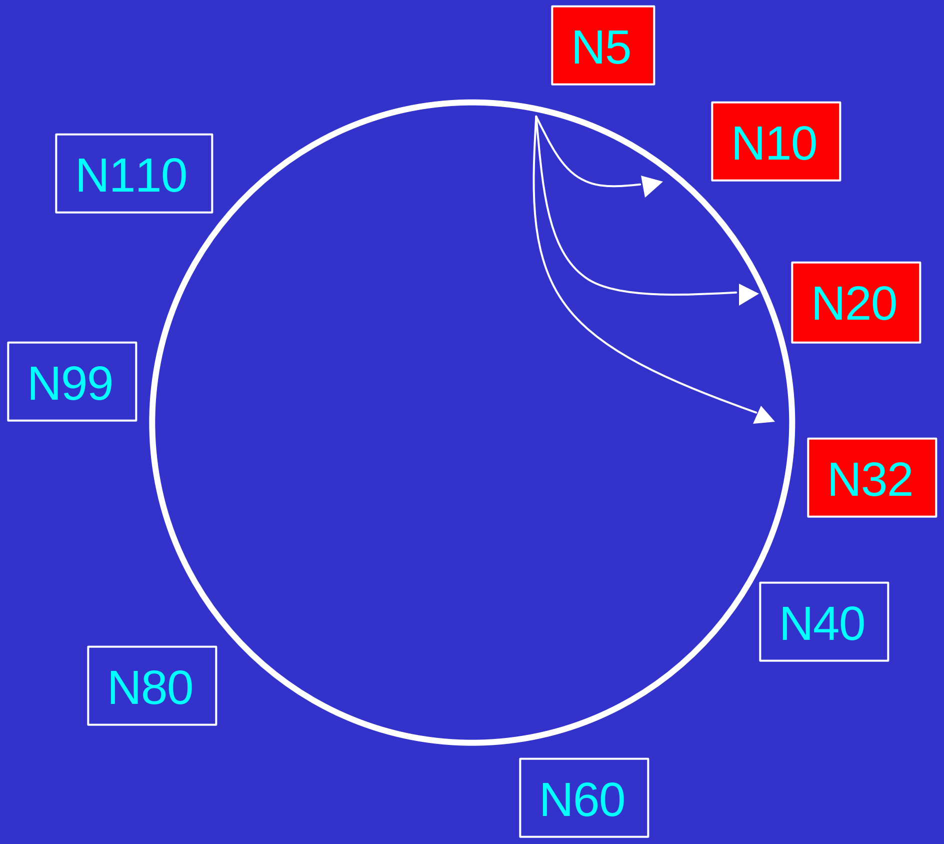
Prevent messages from reaching replica roots

- drop, corrupt, mis-route messages
- bias routing tables to refer to faulty nodes

Cause objects to be placed on faulty nodes

- choose nodeId values
- otherwise impersonate replica roots

Sybil attack [Douceur 02]



- Attacker creates multiple identities
- Attacker controls enough nodes to foil the redundancy

➢ *Need a way to control creation of node IDs*

One solution: certified node IDs

- Certificate authority generates, signs node Ids and public keys of nodes
- Nominal $ charge or real-world identity checks discourage multiple ids

Secure routing primitive

sec-route(key, msg, r): ensures that msg is delivered to each non-faulty node in the set of the *r* closest replica roots for the key, with high probability.

Requires:

- secure nodeId assignment
- secure routine table maintenance
- secure forwarding

Enforcing fair sharing of resources

Two approaches:

- Use byzantine consensus protocol [Castro'99]
  - Each resource use requires approval by a majority among a set of "manager nodes"
- Economic approach [IPTPS'02]
  - Provide incentives for nodes to act honestly

Economic approach

Idea: double-entry bookkeeping plus auditing [IPTPS'03]

- Each node publishes credits (resources it provides) and debits (resources it consumes)
- Incentive to keep "books" accurate:
- Imbalance exposed during audit
- Missing debit allows granting node to withdraw the resource

PAST: Storage quotas

Balance storage supply and demand

- user holds *smartcard* issued by *brokers*
  - hides user private key, usage quota
  - debits quota upon issuing file certificate
- storage nodes hold smartcards
  - advertise supply quota
  - storage nodes subject to random audits within leaf sets

Status

Functional prototypes

- Pastry [*Middleware 2001*]
- PAST [*HotOS-VIII*, *SOSP’01*]
- Scribe [*NGC‘01*, *IEEE JSAC’02*, *NGC’03*]
- SplitStream [SOSP’03]
- Squirrel [*PODC’02*]

http://freepastry.cs.rice.edu

Outline

- Background: Peer-to-peer overlays
- Structured p2p overlays: Pastry
- Pastry proximity-aware routing
- Storing state: Distributed hash tables
- Notification: Cooperative group communication
- Applications
- Conclusions

Applications

- Archival/backup storage: PAST [*SOSP'01*], Pastiche [OSDI'02]
- Filesystems: Ivy [*OSDI'02*], OceanStore [*ASPLOS'00*]
- Cooperative Web caching: Squirrel [*PODC'02*]
- Streaming content distribution: SplitStream [*submitted*]
- Cooperative messaging/communication: Scribe [JSAC'02], POST [*submitted*], i3 [*Sigcomm'02*]
- Distributed database: PIER [*unpub*]

Applications

Augmenting Internet infrastructure:

- group communication (multicast, anycast)
- overlay QoS

Co-operative services

- archival/backup storage
- cooperative Web caching/ flash crowds
- bulk content distribution
- messaging/communication

New applications?

Current Work

- Security
- Resource management, Incentives
- Keyword search capabilities
- Network filesystems
- Streaming content distribution
- Cooperative communication/messaging
- Databases
- Anonymity/Anti-censorship

Conclusion

- Structured p2p are a powerful platform for construction of scalable, resilient, cooperative services
- Much more work to be done to realize the potential
- Looking for novel applications enabled by this technology

For more information

- Pastry: http://freepastry.rice.edu
- IRIS: http://iris.lcs.mit.edu

Peer-to-peer systems

Music sharing: Napster, Gnutella, FreeNet, KaZaA

File storage: CFS [*SOSP'01*], FarSite [*OSDI'02*], Ivy [OSDI'02], Oceanstore [*ASPLOS'00*], Pangea [OSDI'02], PAST [*SOSP'01*], Pastiche [OSDI'02]

Event notification/multicast: Herald [*HotOS'01*], Bayeux [*NOSDAV'01*], CAN-multicast [*NGC'01*], Scribe [*JSAC'02*]

Content distribution: SplitStream [*submitted*], Squirrel [*PODC'02*]

Messaging: i3, POST

Anonymity/Anti-censorship: Crowds [*CACM'99*], Onion routing [*JSAC'98*], Tangler [*CCS'02*], Dagster [*submitted*]

Historical web archiver

- Goal: make and archive a daily check point of the Web
- Estimates:
  - Web is about 57 Tbyte, compressed HTML+img
  - New data per day: 580 Gbyte
  - ➢ 128 Tbyte per year with 5 replicas
- Design:
  - 12,810 nodes: 100 Gbyte disk each and 61 Kbit/s per node