CYCLIC ROUTING IN STRUCTURED PEER-TO-PEER NETWORKS

DMITRY KORZUN AND ANDREI GURTOV

Petrozavodsk State University

Helsinki Institute for Information Technology

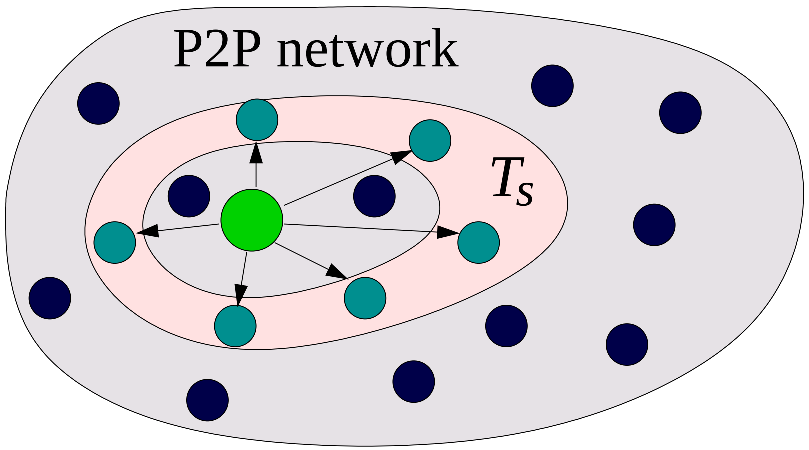
AMICT 2008 seminar

20.05.2008

Typical P2P routing

- P2P network of N nodes
- Node s maintains a routing table $T_s = \{(u, \mathrm{IP}_u)\}$ (all outgoing links of s)
- Node s forwards messages to u via the underlying IP network: $s \xrightarrow{\mathrm{IP}_u} u$
- The choice of appropriate $u \in T_s$ depends on P2P routing protocol (e.g., Chord, Tapestry, Pastry, ...)
- Distance between current and destination nodes becomes progressively closer

A limited (local) view to the network.



Motivation

- Ideally a peer may contact any peer
- Practice, however, it is not so easy

Problems:

- Restricted access to IP addresses
  - non-transitive connectivity
    $u \rightarrow w$, $w \rightarrow u$, but $u \not\rightarrow v$
  - node u and v are NAT-separated
  - node u does not provide IP_u to v
- Malicious nodes
  dropping packets, incorrect data

Goals:

- Extending P2P routing
- More dependability and security
- Preserving efficiency

Related P2P strategies

Look-ahead in $u \to^+ d$

- One level of look-ahead (or neighbor's neighbor)
  $u \to \{v_1, \ldots, v_n\}$ and $v_i \to \{w_{i1}, \ldots, w_{im}\}$
  the best next hop $v = v_k$ is selected depending on $\{w_{11}, \ldots, w_{nm}\}$
- In general, u should select $v = v_k$ depending on the remaining path

Flexible routing table maintenance

- Size $|T_u|$ is not limited by P2P protocol but only by node capacity
- Also $|T_u|$ is independent on other nodes

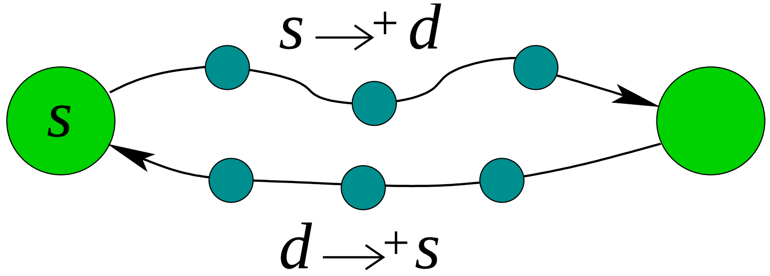
Multipath routing

- Having many neighbors in T_u, u can use some of them in parallel
- Each of these neighbors start an alternate path

Cycles

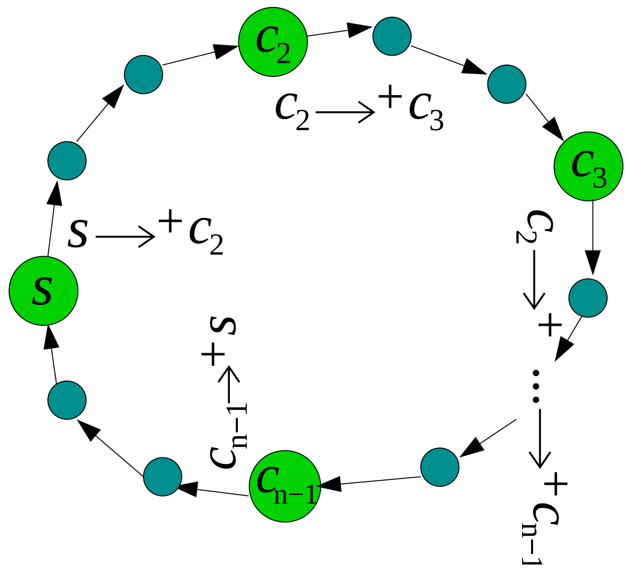
- When s and d communicate they use a cycle

$$s \longrightarrow^{+} d \longrightarrow^{+} s$$



- More information (intermediate nodes)

$$s \longrightarrow^{+} c_2 \longrightarrow^{+} \cdots \longrightarrow^{+} c_{n-1} \longrightarrow^{+} s$$



- In addition to T_s, node s maintains

$$\mathcal{C}_s = \{C_1, \ldots, C_q\},$$

where $C_l = (s; c_{l1}, c_{l2}, \ldots)$

Cyclic routing algorithm

Require: Message p (traveling from s to d) arrives to $u \neq d$.
The node u maintains routing table T_u and cyclic structure $\mathcal{C}_u$.

Find $c \in \mathcal{C}_u$ such that
$c = \big(u \to v_1 \to^+ \cdots \to^+ \widetilde{d} \to^+ \cdots \to^+ v_n \to^+ u\big)$ where $\widetilde{d}$ is close to d;
if c is found **then**
Let v_1 be the next-hop node v;
else
Find the next-hop node $v \in T_u$ according to the underlying DHT;
end if
Forward p to v;

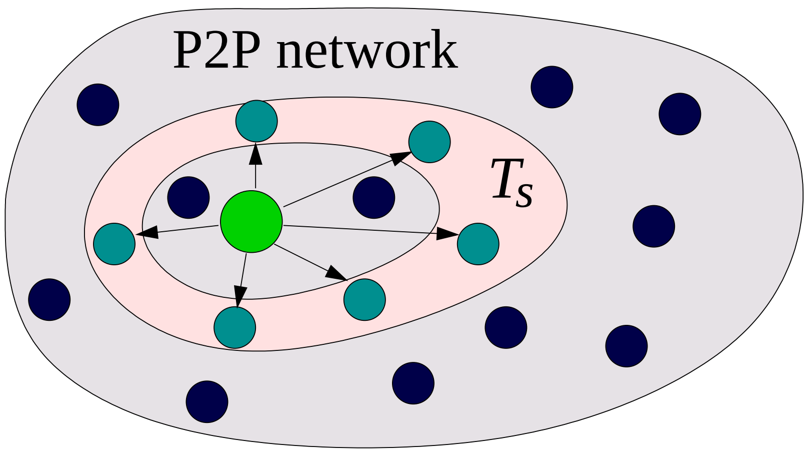
Global and local routing

- **Global**
  Routing to an area where responsible nodes lie

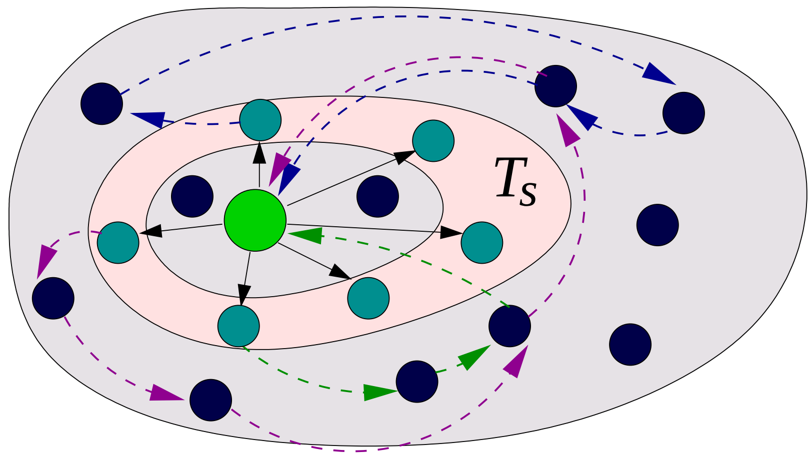
- **Local**
  Being in neighborhood, routing to a destination

- Cyclic routing is global while underlying P2P routing is local

Typical P2P routing



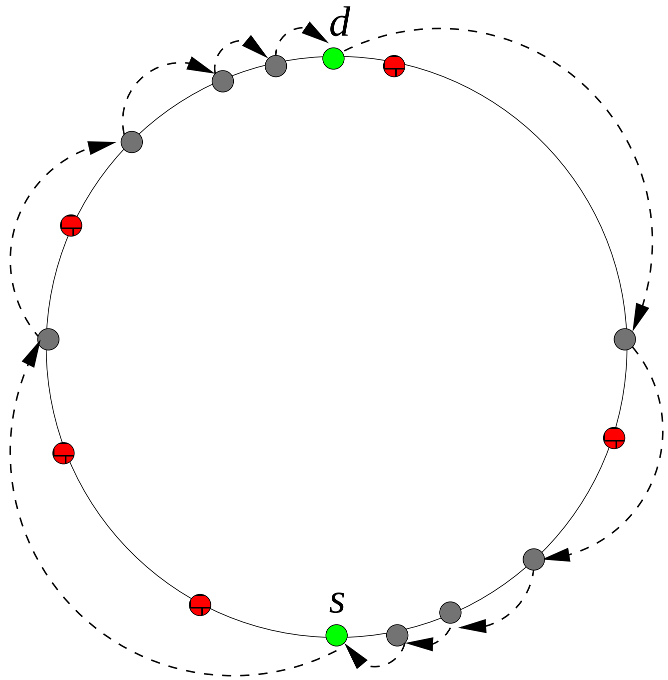
Cyclic routing



Routes around malicious nodes

Lookup-based cycle construction (passive)

1. lookup for `key` leads to path $s \rightarrow^{+} d$
2. acknowledgment, $d \rightarrow^{+} s$
3. cycle is stored in $\mathcal{C}_s$

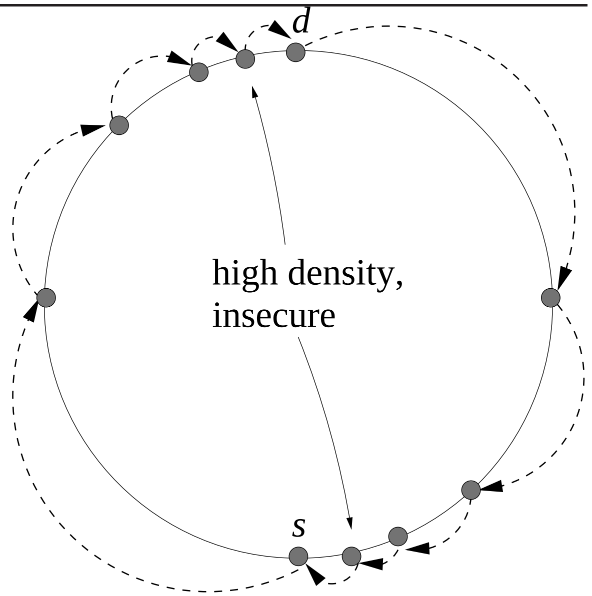


- Successful routes are stored to be reused
- No need for intermediate nodes to provide their IPs
- More security can be added, e.g., cryptography
- Trustworthy paths

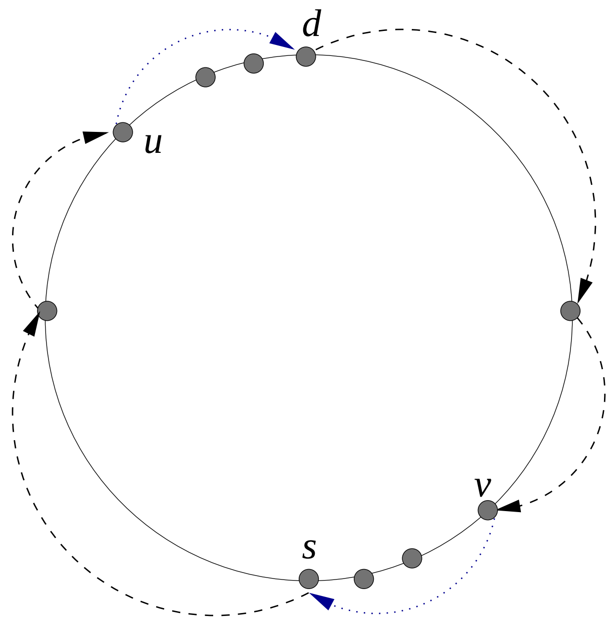
Skipping dense areas

Chord DHT produces insecure routes

(many nodes in small area)

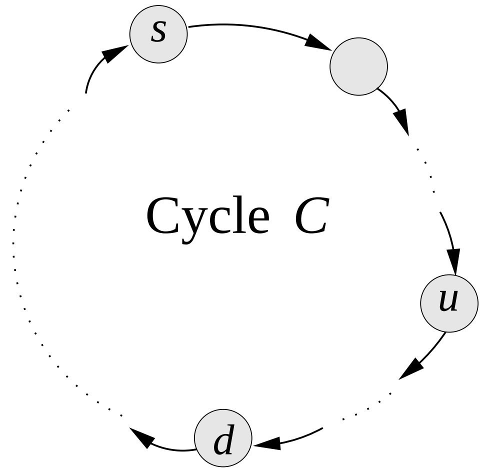


- Constructing cycle $s \to^{+} d \to^{+} s$
- Remove closely related nodes (dense area)
- Nodes u and v are allowed to find a new path to d and s, respectively
- Changing a cycle

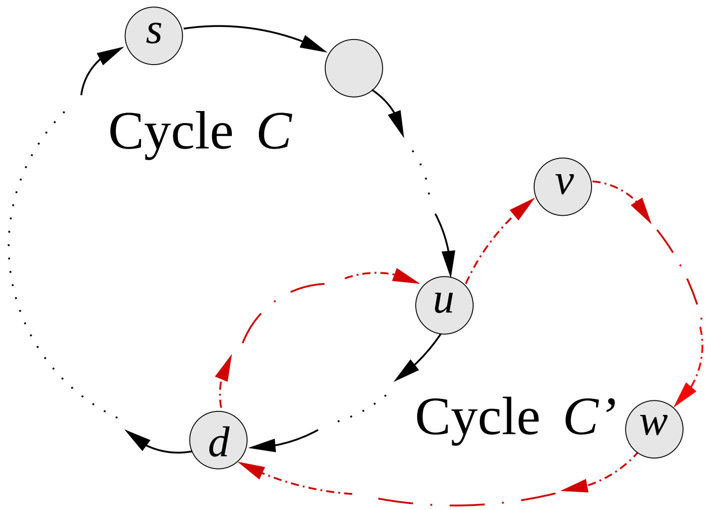


Changing a cycle

- A cycle provides a path to transfer a packet
  - Fig.(a): nodes do not change the path selecting the same (or close) cycle C
  - Fig.(b): node u changes the path selecting cycle $C' = \big(u \rightarrow v \rightarrow^{+} w \rightarrow^{+} d \rightarrow^{+} u\big)$
- It can lead to loops, Fig.(b)
- Chord allows loop-free routing
- A way to modify/repair a cycle initially set by a lookup source



(a)



(b)

Opportunistic routing

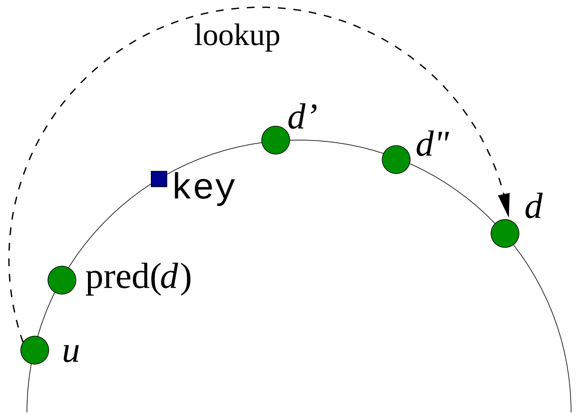
- In pure Chord, the predecessor of a destination node is a point of failure, Fig.(a)
- Let a lookup jump over the primary destination, when replication is in use (DHash by Dabek et al.)
- Stop whenever $\mathrm{nodeID} \leq \mathtt{key}$
- Hopefully we are still in replication area, Fig.(b)
- Estimate in advance:

  $$[\mathtt{key}, \mathtt{key} + r \times D_{\mathrm{avg}}],$$

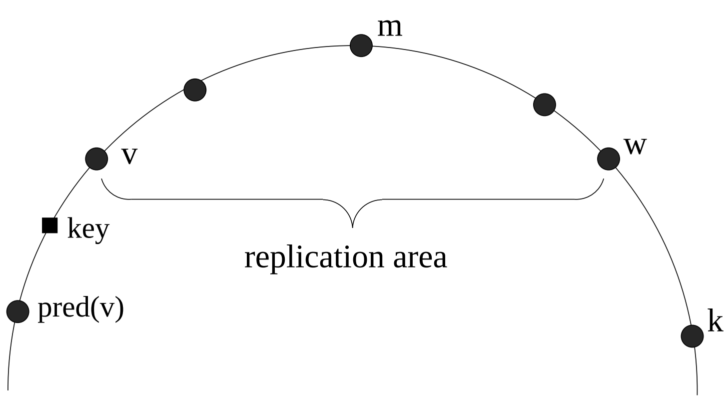
  where r is #replicas, D_{avg} is the average distance between sequential nodes

  More conservatively

  $$[\mathtt{key}, \mathtt{key} + 1/2 \times r \times D_{\mathrm{avg}}]$$



(a)



(b)

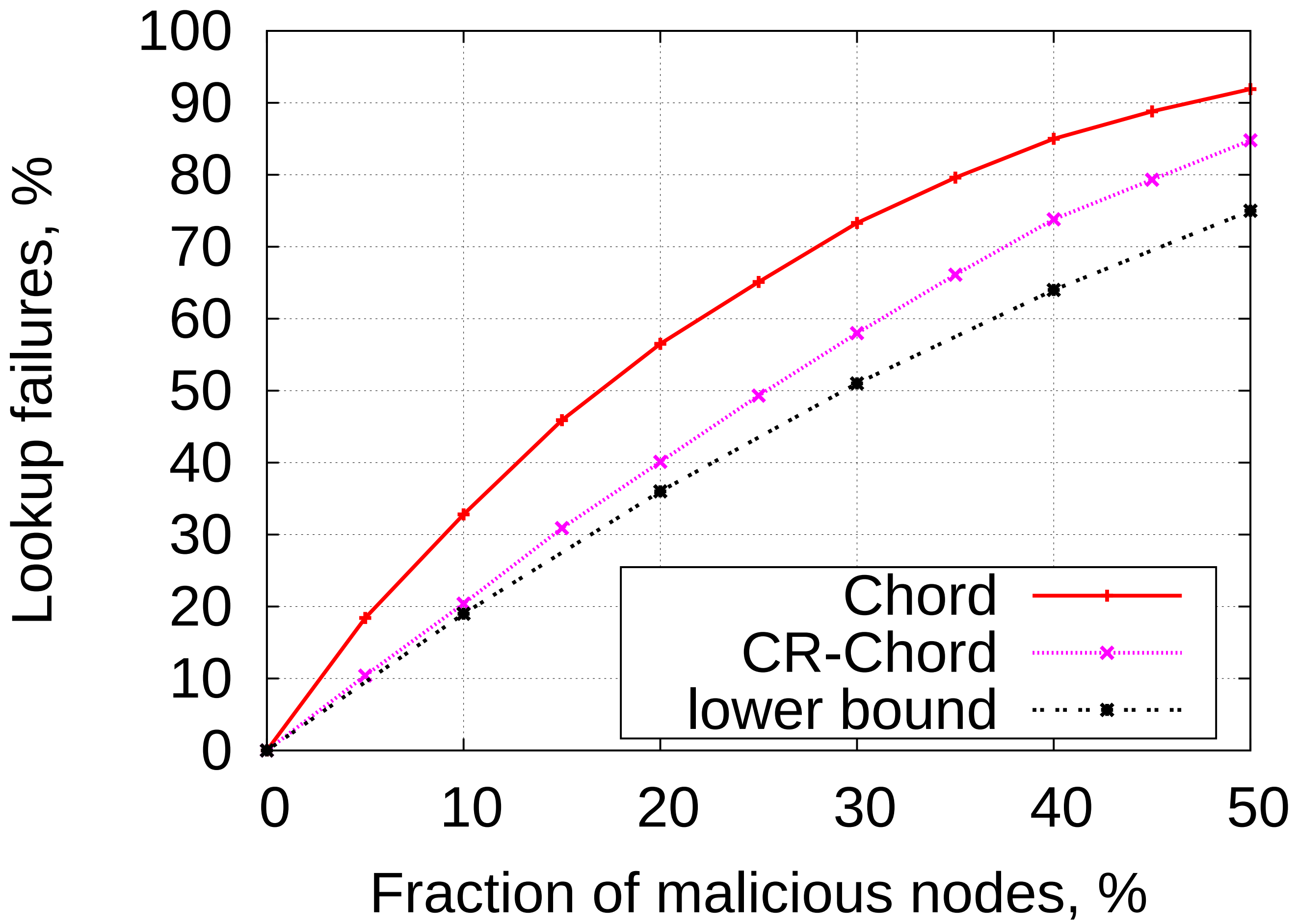
Simulation (together with Boris Nechaev)

CR-Chord = Chord + CyclicRouting

- Goal:
  Find out how cyclic routing helps to mitigate malicious attacks
  (better lookup availability)
- Assumptions:
  - Malicious nodes drop lookup packets, but reply to ping
  - Only good nodes generate lookups and are responsible for documents
  - Currently static environment
  - Instant attack (G good nodes, M malicious nodes, $N = G + M$)

Results

- Chord is not well resistant to presence of malicious nodes
- CR-Chord increases lookup availability
- Note that no IP restrictions were in the simulation



Future work

Cyclic routing:

- Enhancing cycles construction/transformation
- Opportunistic routing
- $\mathcal{C}_s$ evolution (cycles insertion, transformation, removal)
- Finger tables maintenance using cycles

Simulation:

- More intelligent malious nodes and attack scenarios
- IP providing policy (trust)
- Cycles in dynamic environment

Thank you!

Questions?