Analysis of the Evolution of Peer-to-Peer Systems

Proseminar
"Peer – to – Peer Information Systems"
WS 04/05
Prof. Gerhard Weikum

Speaker : Emil Zankov
Tutor : Sebastian Michel

Talking Points

- Motivation
- Related Work
- Problems
- Analysis
- Summary

What is an Ideal P2P Network?

- Running continuously forever
- Efficient lookups
- Allow node to join and leave
- Properly rearranging the ideal overlay

Motivation

- A P2P network works well when the nodes join sequential, but what is if this happen concurrently?
- The overlay is no more ideal if fault occur
- What happens if faults accumulate
- A real P2P system is almost never in ideal state

Goals

By existence of concurrent join und unexpected departure to guarantee:

- Efficient lookups
- Connected Network
- Small number of idealization rounds

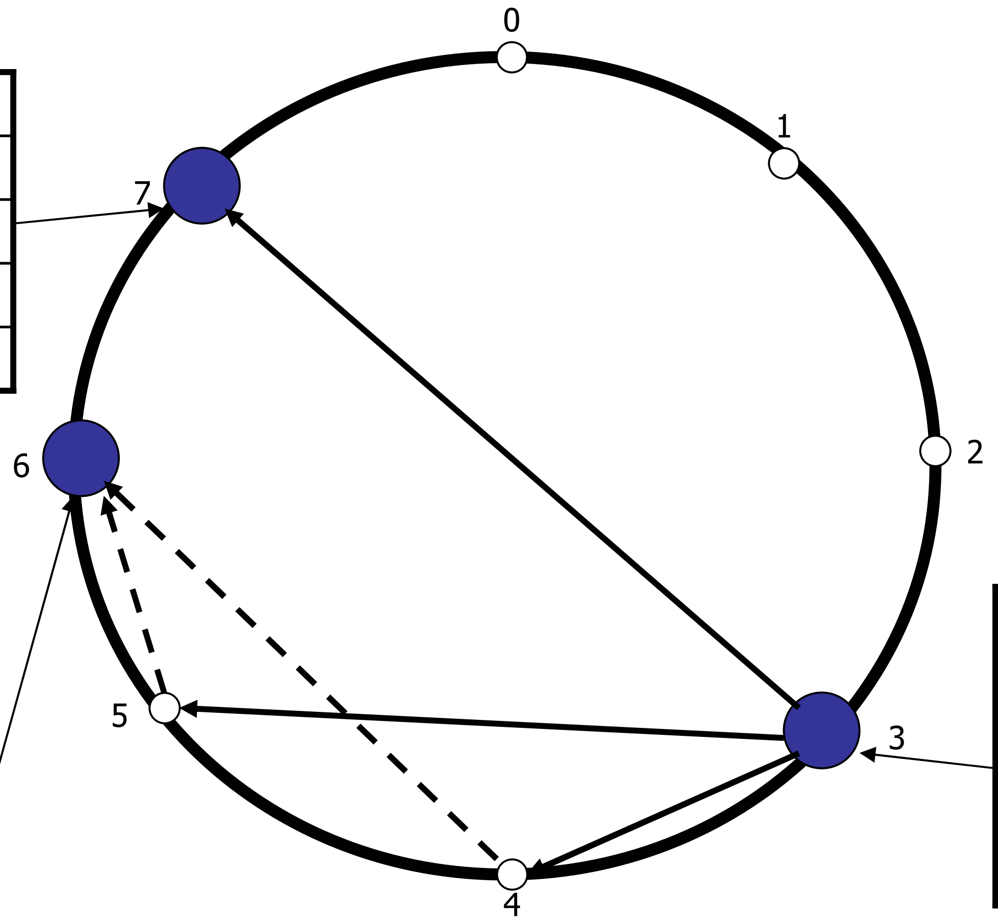
Related Work

- Node join and leave only at well-behaved network (Plaxton et al. 1997)
- Fault tolerance only if |joining nodes| > |departing nodes|(Saia at al. 2002)
- Maintenance using an central server (Pandurangan et al. 2001)

The Chord P2P System

Finger Table 7		
start	int	succ
0	[0,1)	3
1	[1,3)	3
3	[3,7)	3

Finger Table 6		
start	int	succ
7	[7,0)	7
0	[0,2)	3
2	[2,6)	3



Finger Table 3		
start	int	succ
4	[4,5)	6
5	[5,7)	6
7	[7,3)	7

Half-Life Definition

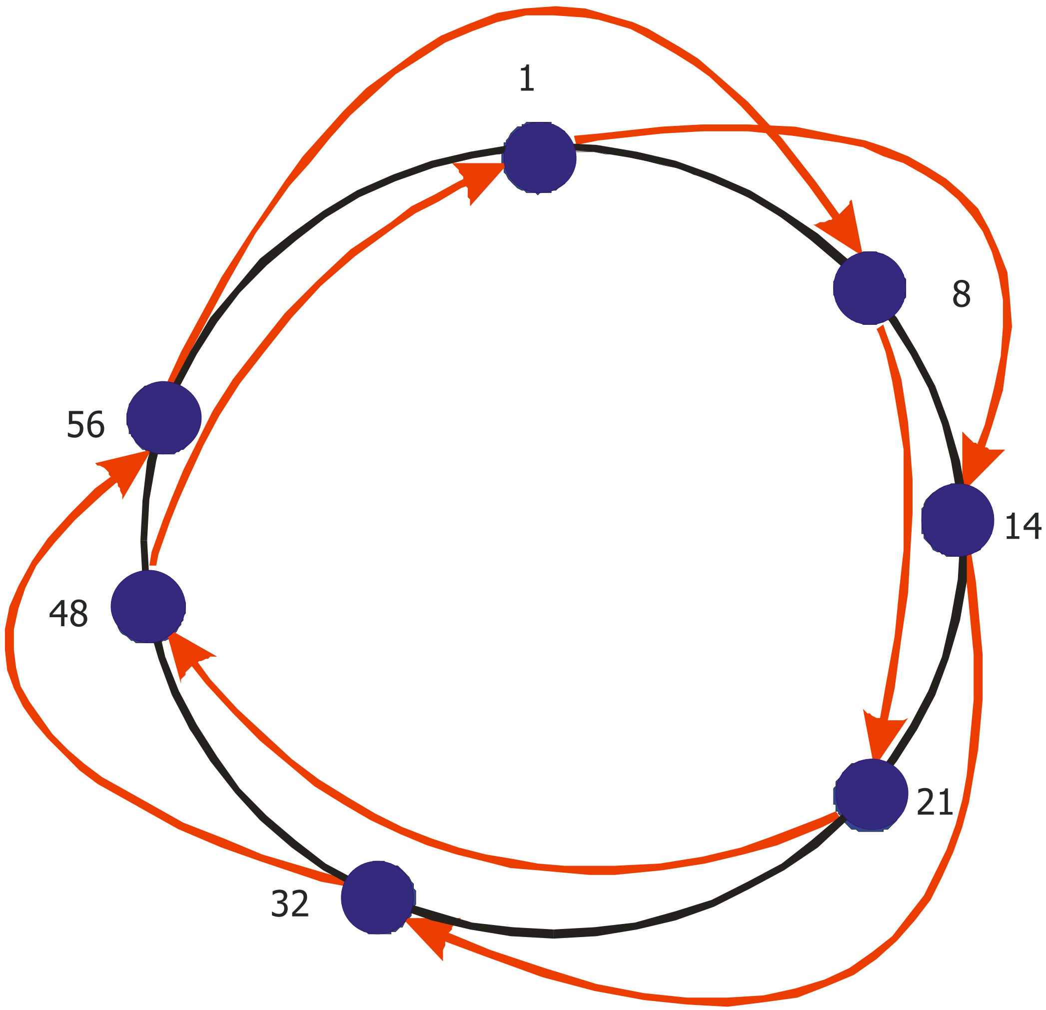
Given a *N* node system at time t

- Doubling time(DT) – a time from t required for *N* additional nodes to join
- Halving time(HT) – a time from t required for *N*/2 live nodes to depart
- Half-Life is Min(DT,HT)

Half-Life is a coarse factor of the rate of change of a system

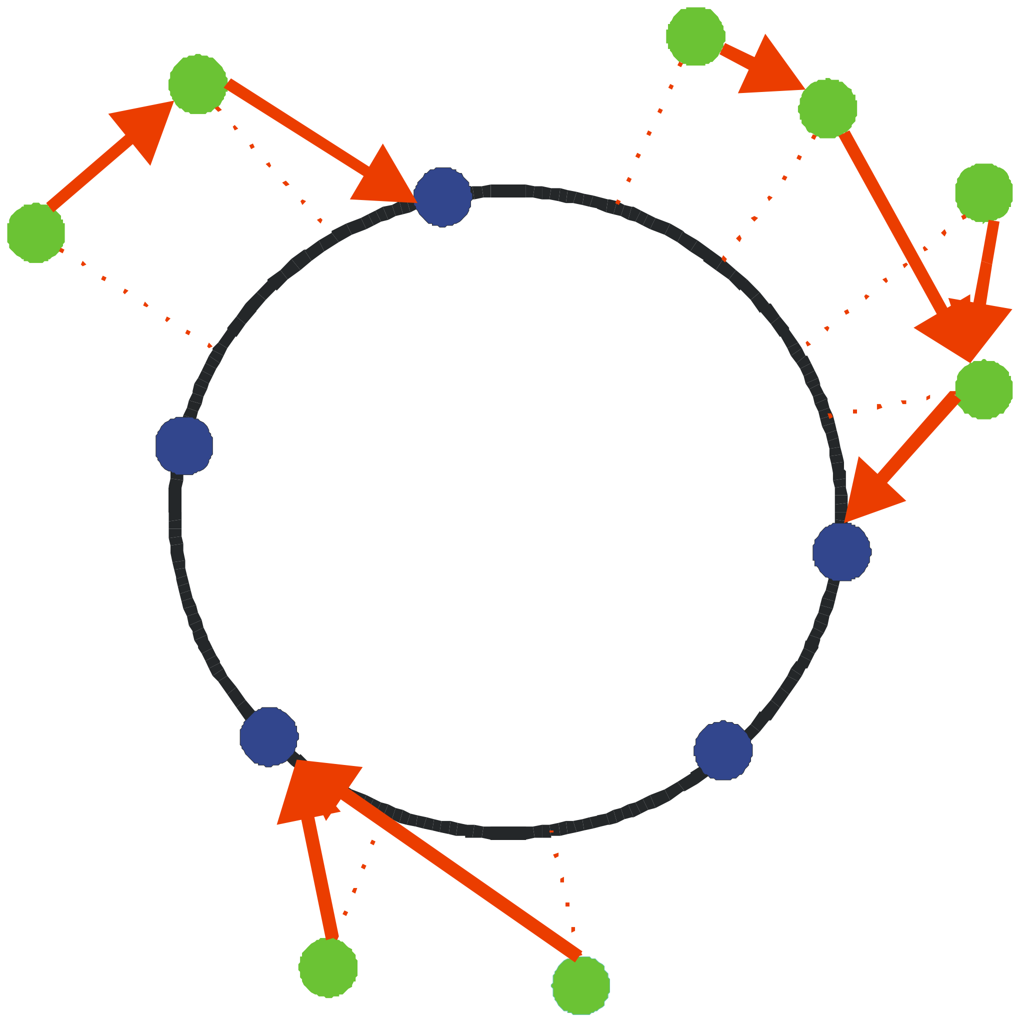
Loopy Problem

- Loopy states
  - Weakly ideal
  - Strong ideal
- Reasons
  - Impl. Bugs
  - Breakdown of join/depart Model
  - Low probability events



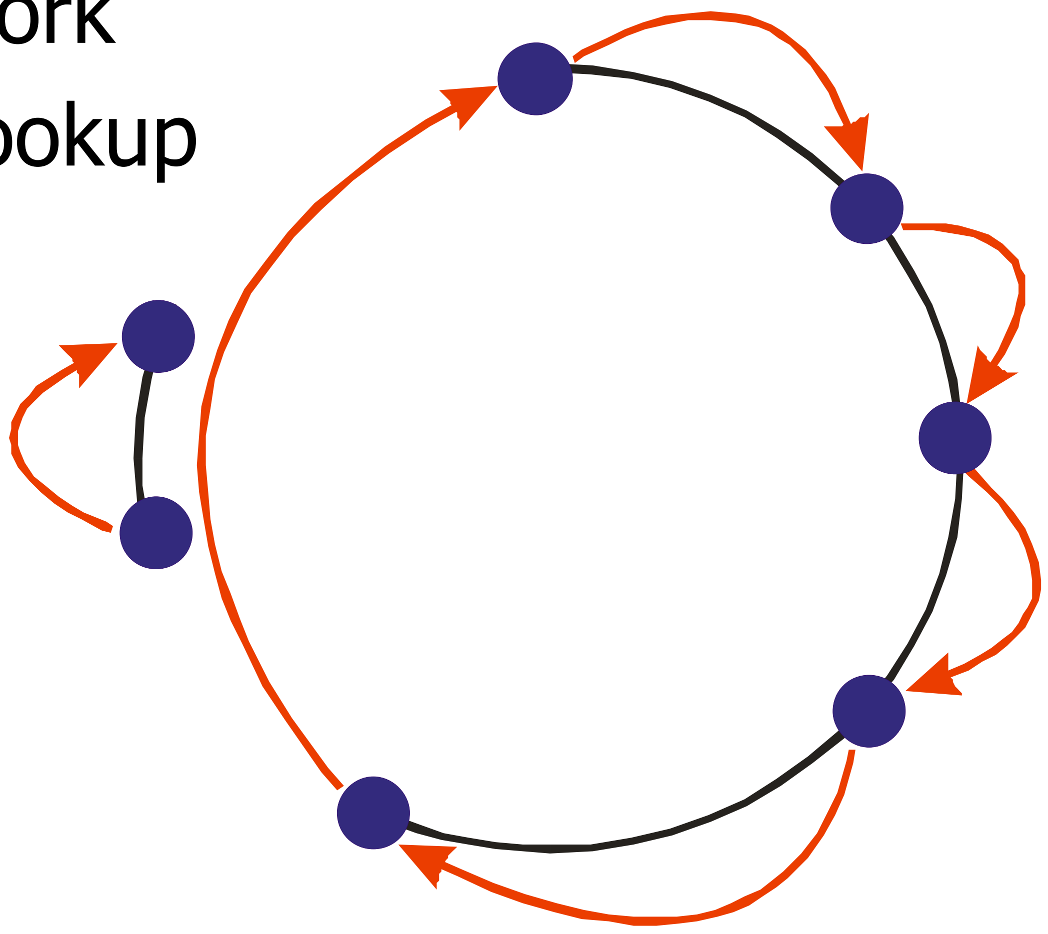
Appendages Problem

- Recently joined nodes
- Non empty tree rooted at any node



Failure Problem

- Split the Network
- Inconsistent lookup



The Ideal Chord State

- Connectivity
  - Exist path between any two nodes
- Randomness
  - Independently and uniformly distributed nodes
- Cycle sufficiency
  - Every node is on the cycle
- Non-loopiness
  - $\forall u \in Cyc \Rightarrow \neg \exists v : v \in (u, u.successor)$

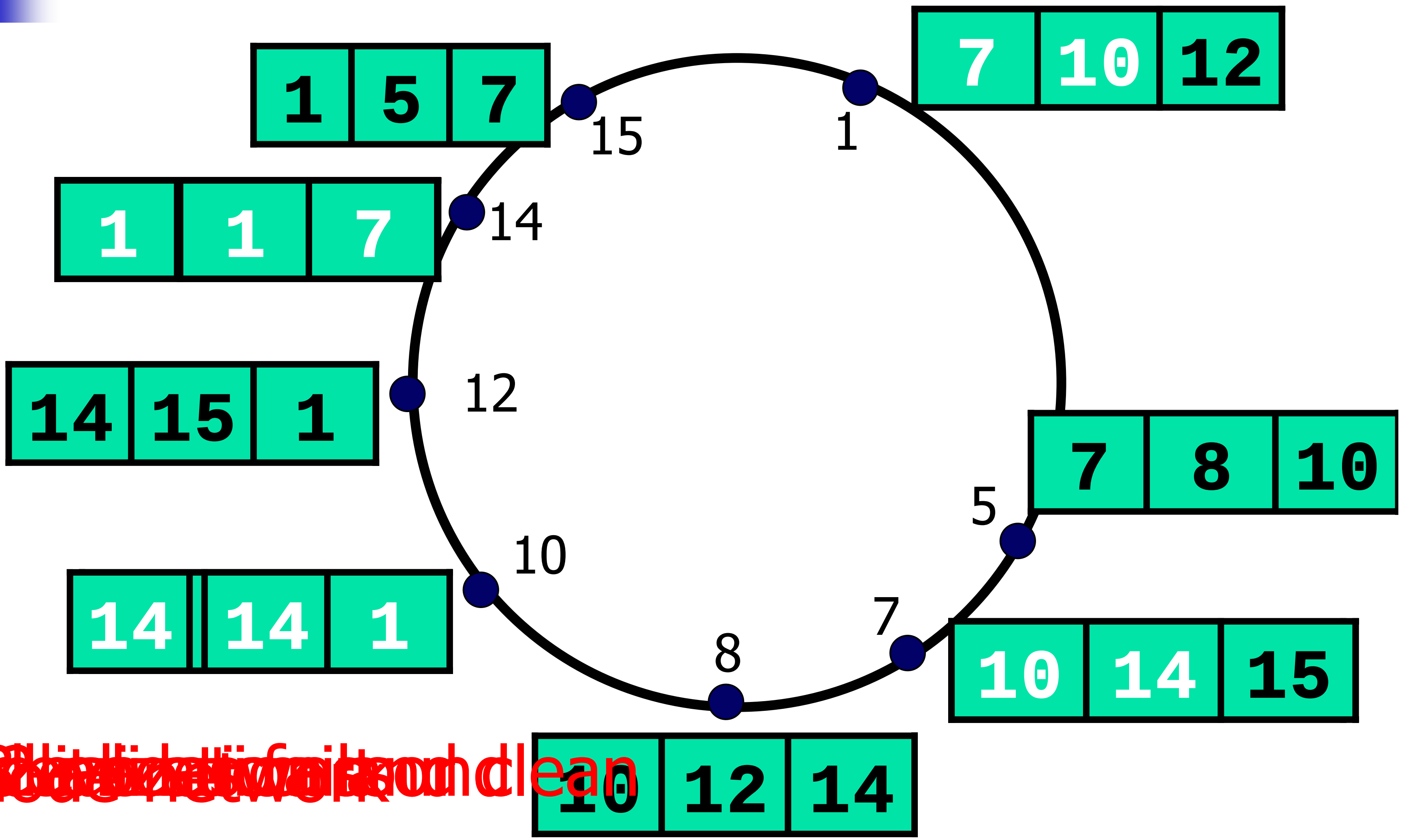
The Ideal Chord State

- Successor list validity
  - Every *u.successor_list* contains the first *c*logN* nodes that follows *u,* c=O(1)
- Finger validity
  - The first node following $u + 2^{i-1}$ is stored as *u.finger[i]*

A Failure Model Definition

- Successor list validity
  - Every $|L_u| \geq (c/3)*\log N$
  - Every L_u contains exacly the first $|L_u|$ live nodes that follow *u*
- Finger validity
  - If *u.finger[i]* is alive then it's the first live node following $u + 2^{i-1}$

A Pure Failure Model



Lemma

For an N node network in with failure, occur $N/2$ oblivious failures during $\Omega(\log N)$ idealization rounds, then:

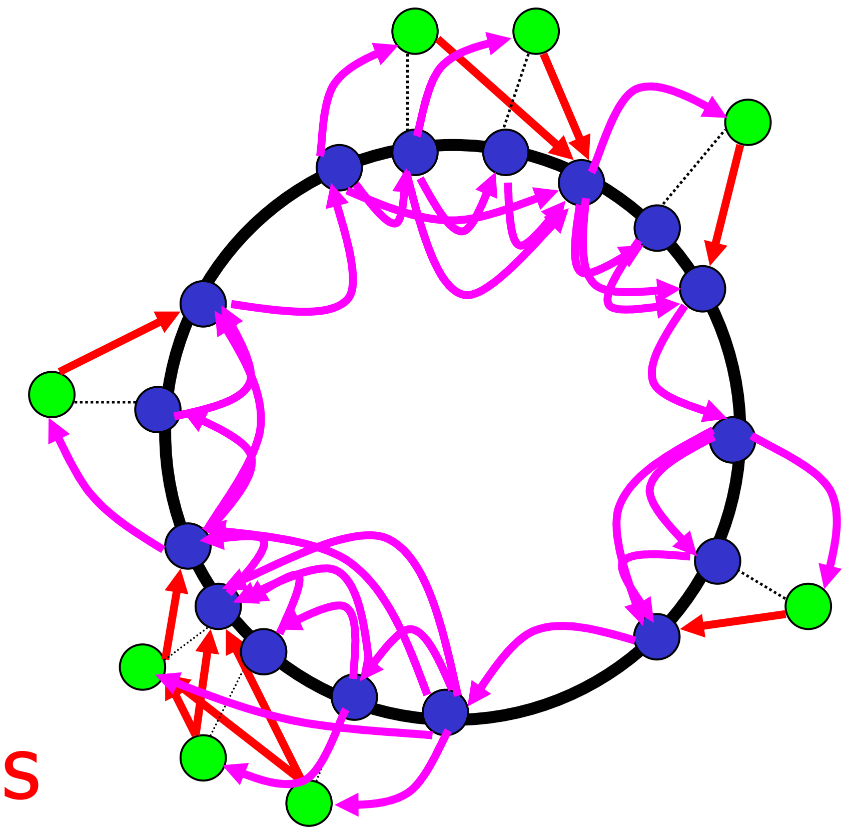
=> Throughout this process, *find_successor(q)* returns the first living successor of q in $O(\log N)$ time

=> Resulting network is in cycle with failures state

A Pure Join Model

O(log *N*) incorporating rounds

O(log² *N*) time fully incorporating



Incorporating round
Between stages

A Join Model Definition

- $|A_u| = \mathrm{O}(\log N)$
- The finger are correct with respect to a constant fraction of the nodes on the system

A_u – appendage nodes rooted at u

Lemma

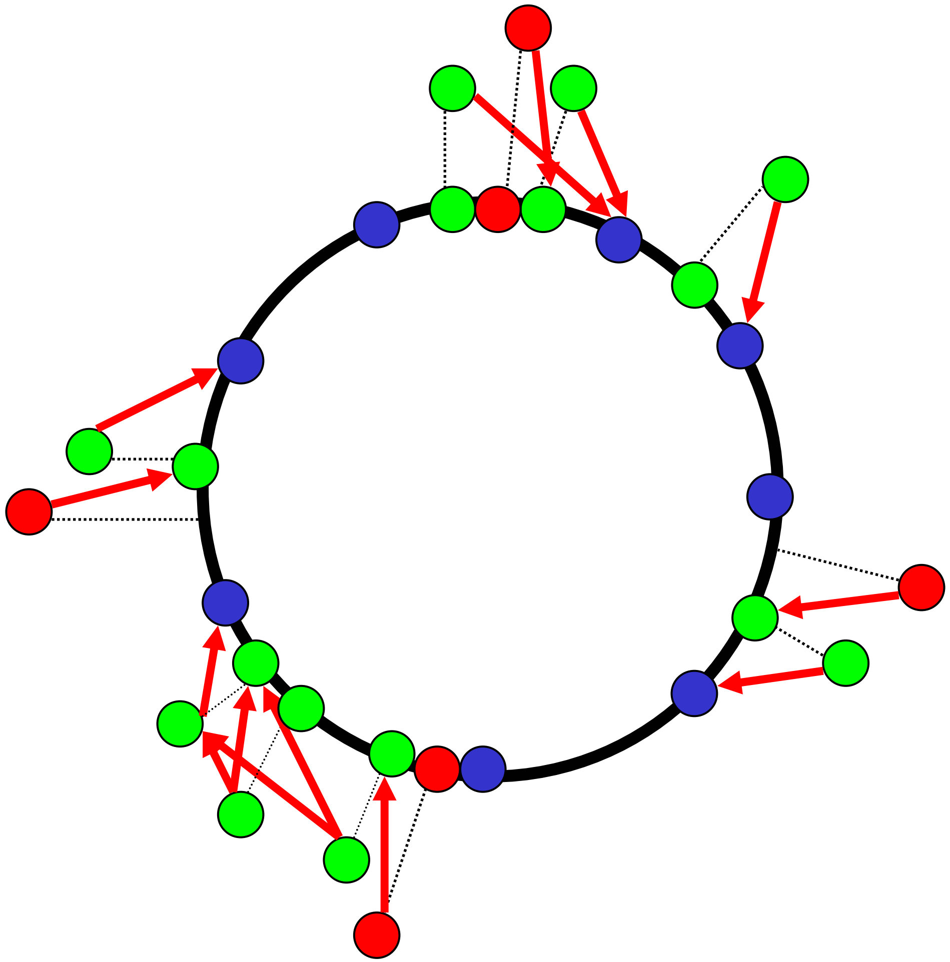
For a N node network with appendages, suppose that during $\Omega(\log^2 N)$ rounds of idealization N nodes join the network, then:

=> Throughout this process, *find_successor(q)* returns in $O(\log N)$ time the successor $s \in$Cyc or $a \in A_s$

=> the resulting network is in cycle with appendages state

Proof

- O(log*N*) time to find any old node
- O(log*N*) round to incorporate the new nodes from the appendages



A Dynamic Model Definition

- Union from the Join Model and the Failure Model

- Cycle sufficiency
  - For any consecutive cycle nodes $u_1 \ldots\ldots u_{\log N}$

=> $$\sum_{i=1}^{\log N} \left| A_{u_i} \right| = O(\log N)$$

Theorem

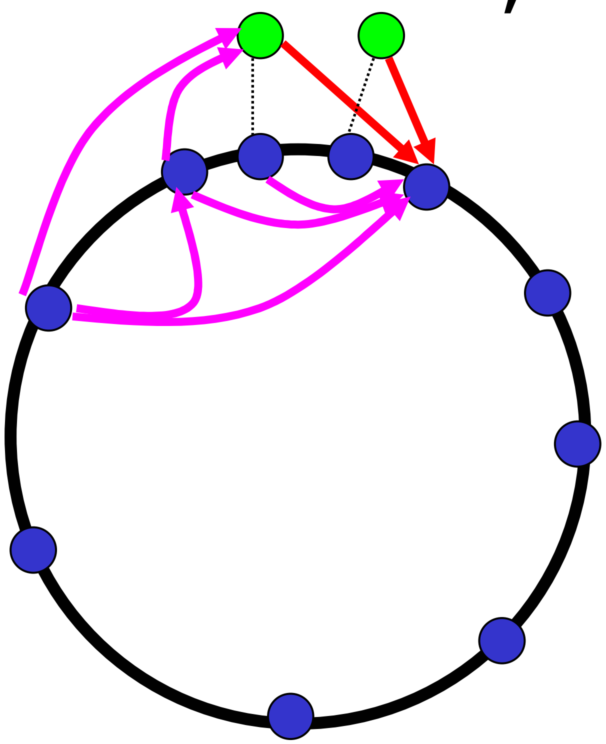
A network of N nodes in cycle with appendages and failures state, allow up to N oblivious joins and $N/2$ failures at time over $D*\log^2 N$ (D=O(1))idealization rounds, then:

=> *find_successor(q)* returns the first living successor of q in $O(\log N)$ time

=> The resulting network is in cycle with appendages and failures state

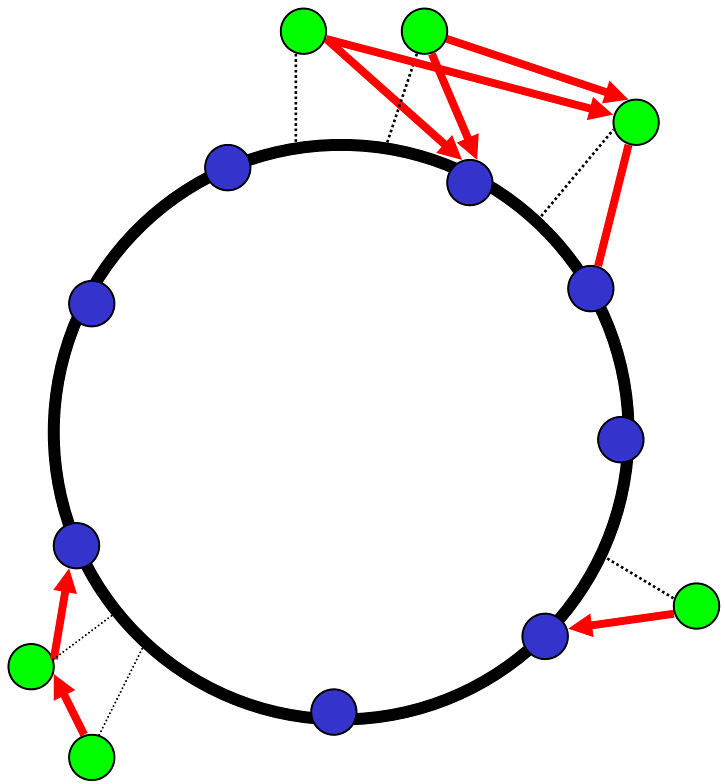
Proof

- Incorporation of node from each appendage in each round is not sure, since cycle node fails



Proof

- Cycle nodes fails, causing their appendages merge together



Summary

- The Chord system works good in existence of dynamic departures and joins
  - Resolve queries efficient O(log*N*) time
  - The maintenance work can be reduced by logarithmic factors

Thanks?

Questions?

Summary

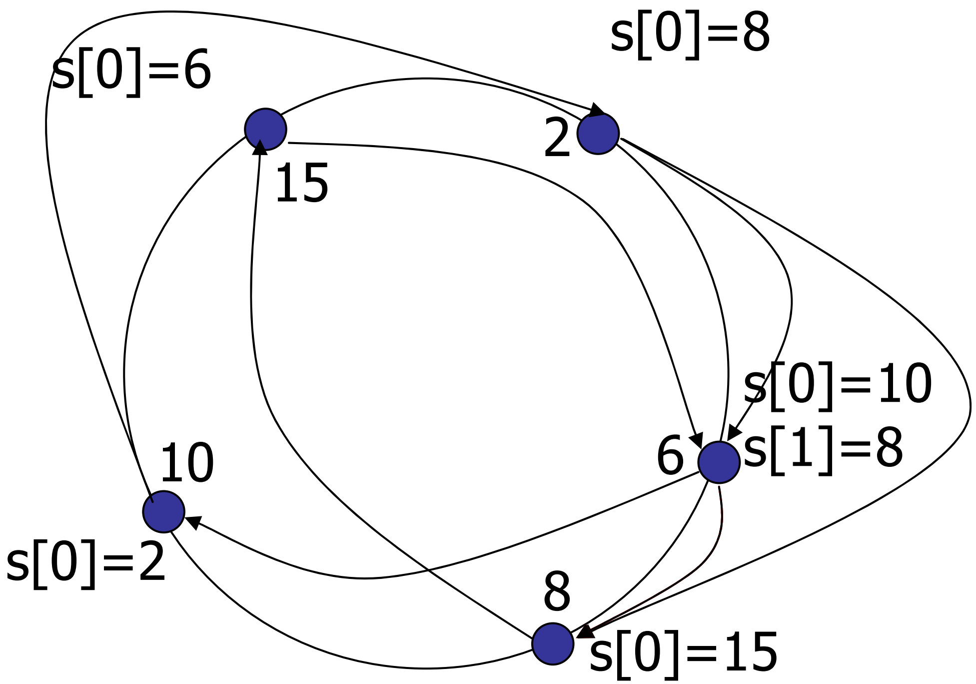
- The Chord system works good in existence of dynamic departures and joins
  - Resolve queries efficient O(log*N*) time
  - The maintenance work can be reduced by logarithmic factors
- How can we find the Half-Life?

Proof

- Connected Network since
  - log*N* live entries in *successor_list*
  - nodes fail with constant probability
- *u.find_successor(k)* is efficient
  - the next node's i-th finger is up with 1/2 probability
  - each forwarding halves the distance
- Clean the old failures in *successor_list*

Proof

- An *N* Node weak ideal network without loopy states have *N* successor pointer



Theorem

Within $O(N^2)$ rounds of strong idealization, an arbitrary connected Chord network becomes strongly ideal

How to Find a Loopy State

- A Self Search