

---

# Failure Recovery for Structured P2P Network: Protocol Design and Performance Evaluation

---

Huaiyu Liu

Joint work with Prof. Simon S. Lam



*Department of Computer Sciences*

---

*The University of Texas at Austin*

---

# Structured P2P Network

- n The routing scheme: Hypercube routing scheme used by PRR, Pastry, Tapestry, etc.
- n Important issue: Design of protocols to construct and maintain **consistent** neighbor tables under node dynamics
- n Question: *How high a rate of node dynamics can be supported by a structured P2P network?*

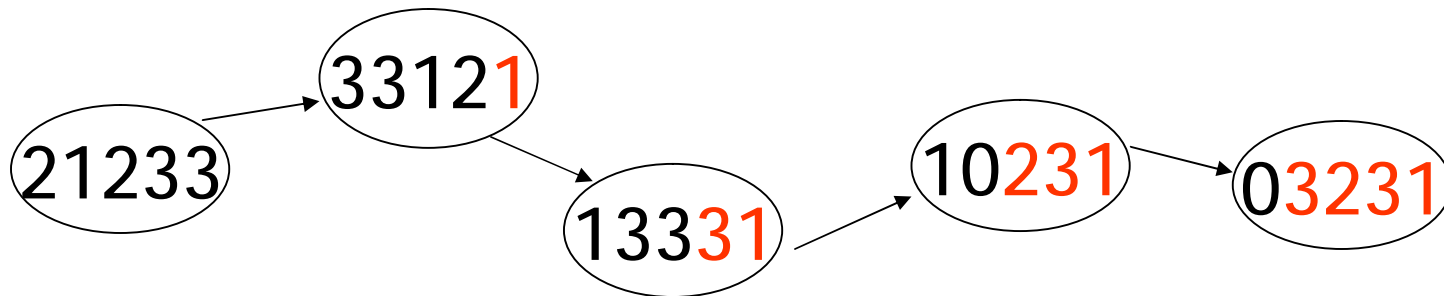
---

# Outline

- n The problem
- n Overview of hypercube routing scheme
- n Our approach
  - q K-consistent network
  - q Basic failure recovery protocol
  - q Integrate failure recovery with a join protocol
- n Churn experiments
- n Conclusions

# Overview of Hypercube Routing Scheme

- n Each node has an ID, represented by  $d$  digits of base  $b$ .
  - q E.g. 10323 ( $d = 5$ ,  $b = 4$ )
- n Routing to a destination node is resolved digit by digit.



Example: source 21233, destination 03231

# Neighbor Table

- $n$   $d$  levels,  $b$  entries at each level
- $n$  Neighbors stored in an entry must have the *required suffix* of the entry

Example: neighbor table of node 21233 ( $d=5$ ,  $b=4$ )

	01233	10233	0233	31033	033	22303	03	01100	0
11233	11233	21233	1233	03133	133	13113	13	33121	1
21233	21233		2233	21233	233	00123	23	12232	2
	31233	03233	3233		333	21233	33	21233	3
Level 4		Level 3		Level 2		Level 1		Level 0	

---

# Outline

- n The problem
- n Overview of hypercube routing scheme
- n **Our approach**
  - q K-consistent network
  - q Basic failure recovery protocol
  - q Integrate failure recovery with a join protocol
- n Churn experiments
- n Conclusions

---

# K-consistent Network: Definition

- n A network is **K-consistent** iff:  
Every table entry stores  $\min(K, H)$  neighbors, where  $H$  is the number of nodes with the required suffix of the entry

---

# K-consistent Network: Benefits

- n K-consistency
  - q implies *consistency*, which guarantees a path for any source-destination pair
  - q provides  $K$  disjoint paths for each source-destination pair with prob. close to 1
  - q facilitates failure recovery

---

# Basic Failure Recovery

- n Assumption:
  - q A network of  $n$  nodes, initially  $K$ -consistent;
  - q  $f$  out of  $n$  nodes fail (fail-stop)
- n Objective: When all failure recovery processes terminate,
  - q all “recoverable holes” are repaired
  - q the network is  $K$ -consistent again

# Basic Failure Recovery Protocol

- n A sequence of search steps, based on **local information** (neighbors and reverse neighbors)

Neighbors of node <b>21233</b>		10233	31033	<del>22303</del>	01100
	11233	21233	03133	02203	23310
	21233	11233	10133	13113	33121
			21233	00013	10131
		03233	03233	00123	23212
				22323	12232
				21233	00013
				03133	21233

Reverser neighbors of **21233**

{ the set of nodes that stores 21233 as a neighbor }

**STEP (a):** search among neighbors and reverse-neighbors

# Basic Failure Recovery Protocol

- n A sequence of search steps, based on **local information** (neighbors and reverse neighbors)

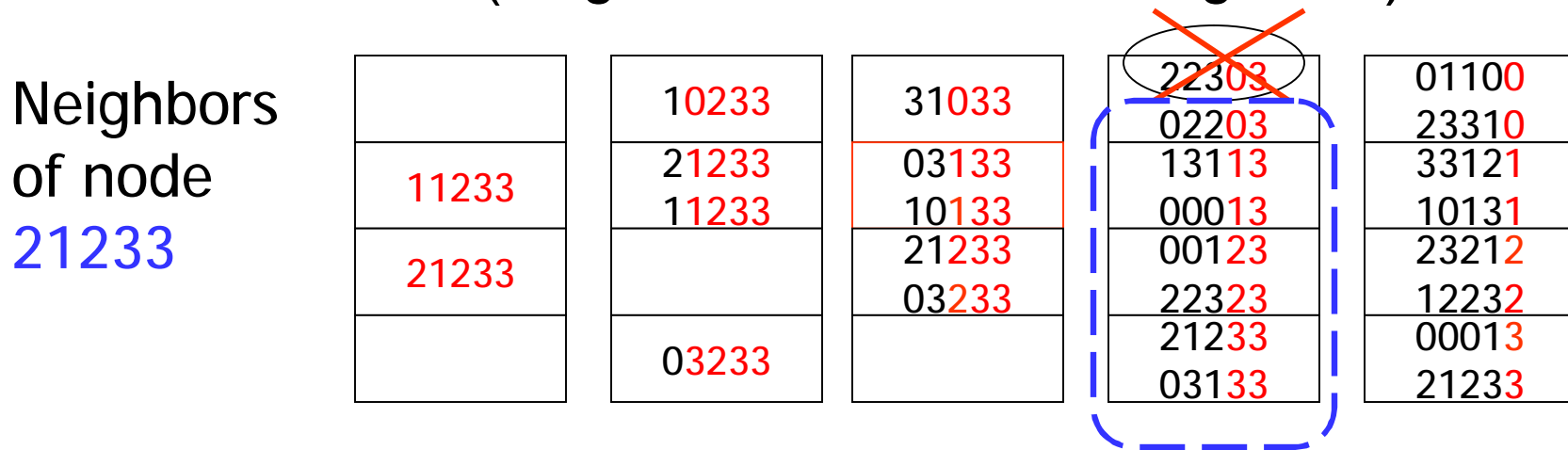
Neighbors of node **21233**

			<del>22303</del>	01100
11233	10233	31033	02203	23310
21233	21233	03133	13113	33121
	11233	10133	00013	10131
		21233	00123	23212
		03233	22323	12232
	03233		21233	00013
			03133	21233

**STEP (b):** query remaining neighbors in the same entry

# Basic Failure Recovery Protocol

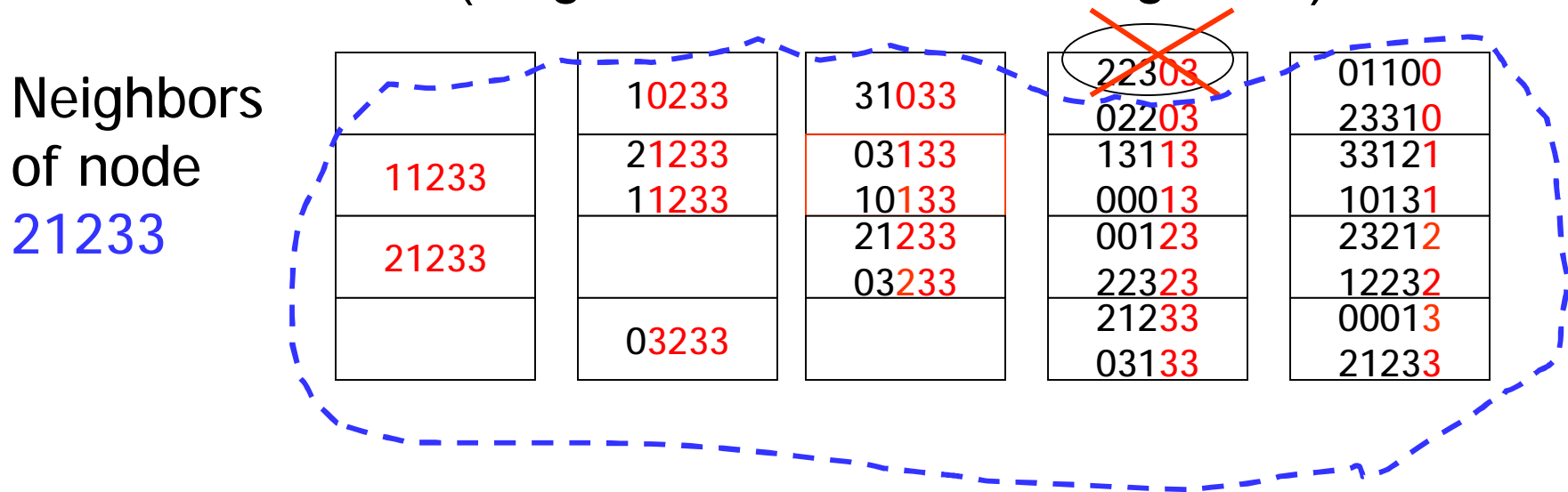
- n A sequence of search steps, based on **local information** (neighbors and reverse neighbors)



**STEP (c):** query remaining neighbors at the same level

# Basic Failure Recovery Protocol

- n A sequence of search steps, based on **local information** (neighbors and reverse neighbors)



**STEP (d):** query all remaining neighbors

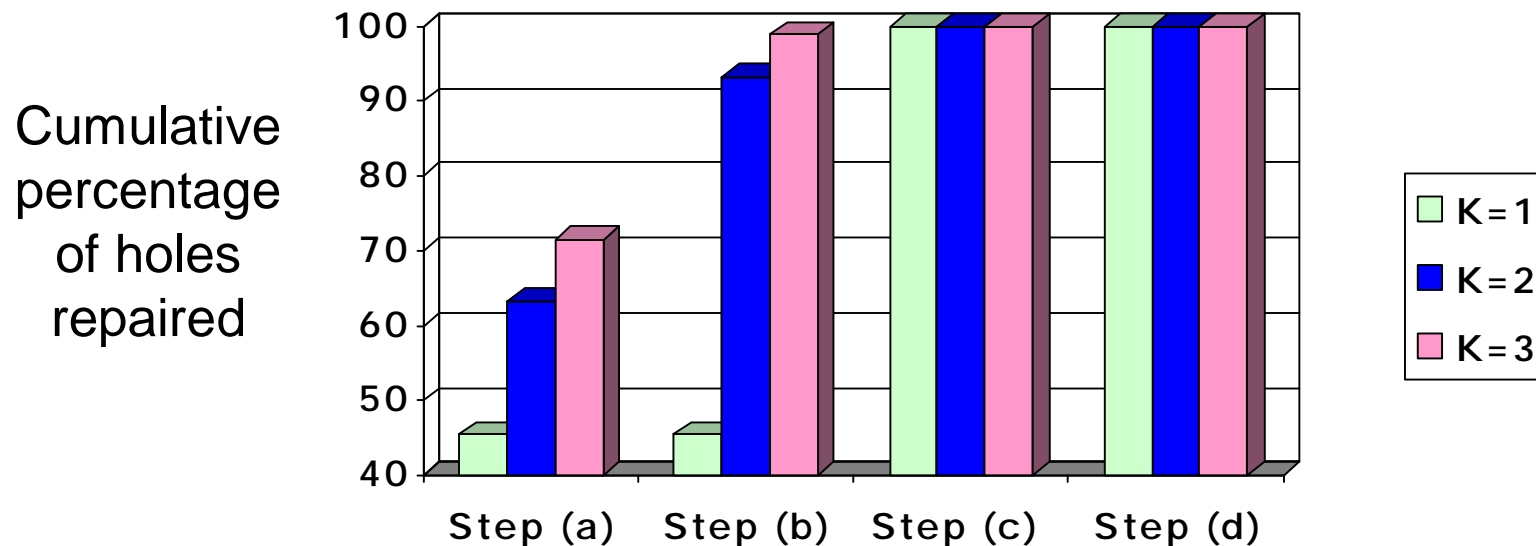
# Failure Recovery is Effective

- n 2,080 experiments,  $K=1\sim 5$ ,  $n=1000\sim 8000$
- n 5% - 50% nodes fail
- n All “recoverable holes” are repaired in each experiment, for  $K\geq 2$

$K, n$	Number of simulations	Number of perfect recoveries	$K, n$	Number of simulations	Number of perfect recoveries
1,1000	100	51	1, 2000	180	96
2,1000	100	100	2, 2000	180	180
3,1000	100	100	3, 2000	180	180
4,1000	100	100	4, 2000	180	180
5,1000	100	100	5, 2000	180	180
1,4000	116	65	1, 8000	20	14
2,4000	116	116	2, 8000	20	20
3,4000	116	116	3, 8000	20	20
4,4000	116	116	4, 8000	20	20
5,4000	116	116	5, 8000	20	20

# Failure Recovery is Efficient

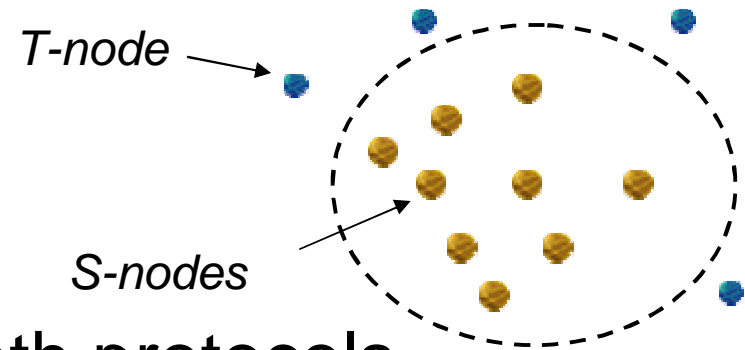
- n Majority of holes repaired in step (a), no communication cost
- n Almost all holes repaired by step (c), at most 2Kb messages for repairing a hole



Example: 800 out of 4000 nodes fail,  $b=16$ ,  $d=40$

# Integrated Protocols

- n Integrate failure recovery with our join protocol [ICDCS'03]
- n Distinguish T-nodes from S-nodes
  - q *S-nodes*: Nodes finished joining
  - q *T-nodes*: Nodes joining a network
- n Requires extensions to both protocols
- n Give failure recovery actions higher priority, to prevent circular reasoning



---

# Results for Concurrent Joins and failures

- n 980 experiments
- n Start with a  $K$ -consistent network
- n Massive joins and failures occur concurrently
- n For  $K \geq 2$ ,  $K$ -consistency is maintained at the end in every experiment

---

# Outline

- n The problem
- n Background
- n Our approach
  - q K-consistent network
  - q Basic failure recovery
  - q Integrate failure recovery with a join protocol
- n Churn experiments
- n Conclusions

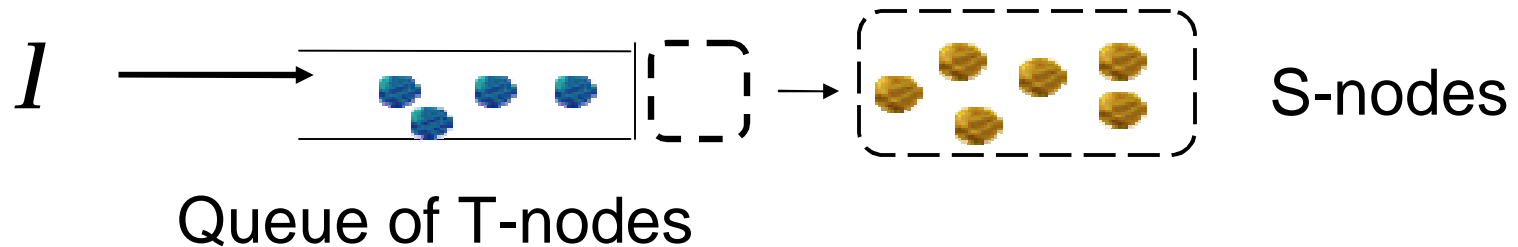
# Churn Experiments

How high a rate of node dynamics can be sustained?

- n Start with a K-consistent network of 2000-node
- n Generate join and failure events for 10,000 simulation seconds
  - q Join rate = failure rate =  $I$  (churn rate)
- n Take a snapshot every 50 seconds
  - q Evaluate connectivity and consistency measures
- n Convergence to K-consistency at the end

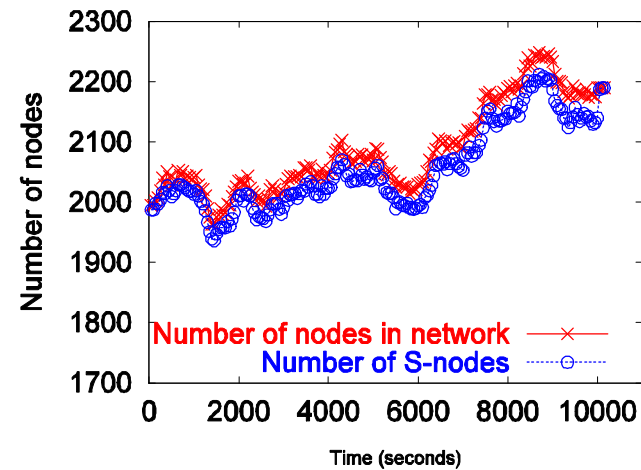
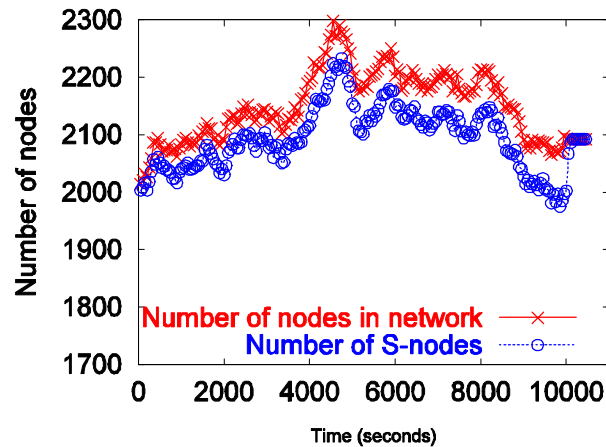
# Observations

- n Sustainable churn rate is upper bounded by the network's *join capacity*
- n Join capacity: the rate at which new nodes can join the network successfully

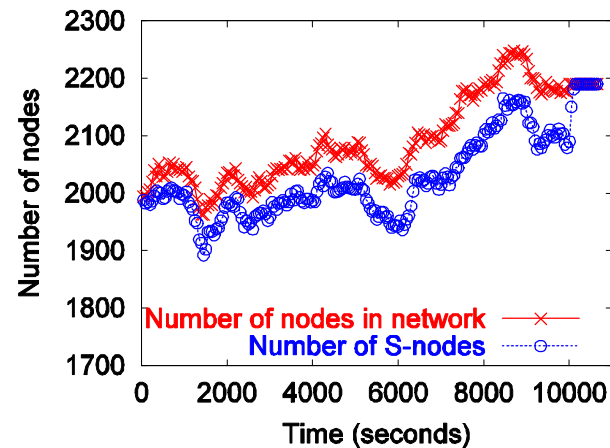
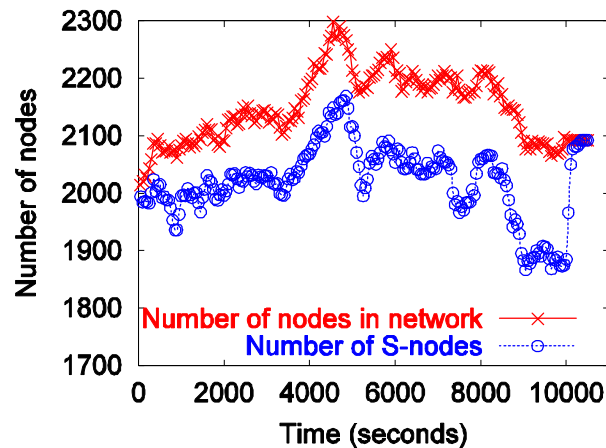


- n The limiting factors
  - q  $K$
  - q failure rate  $I$
  - q timeout value in each failure recovery step

# Number of Nodes and S-nodes vs. Time



$I = 1$



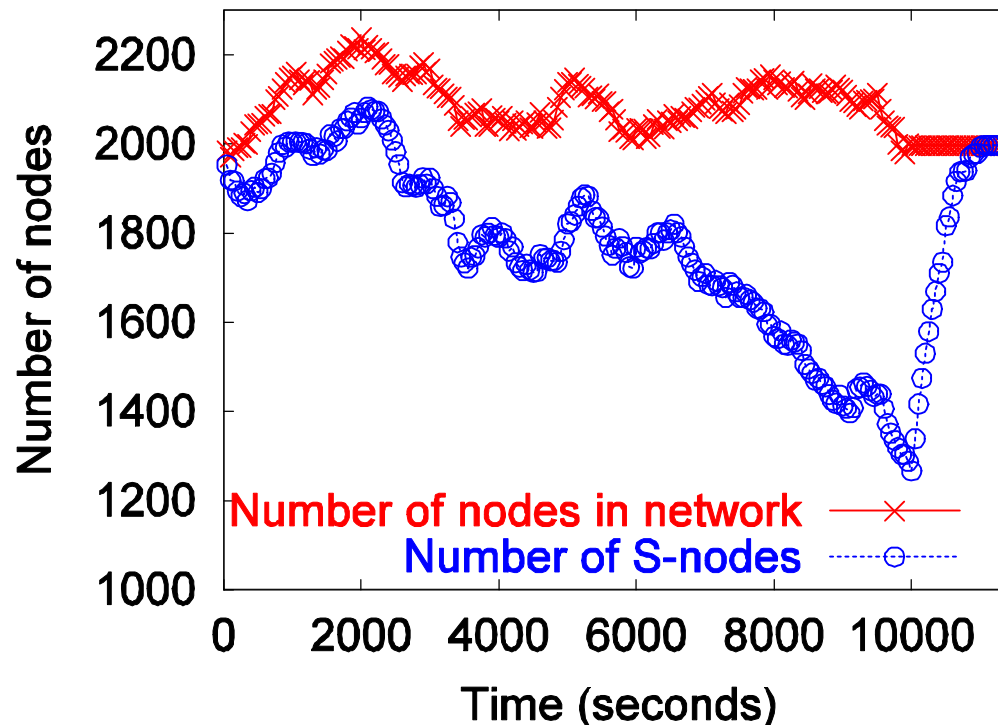
$I = 1.5$

Timeout = 10sec, K=3

Timeout = 5sec, K=3

# When Join Capacity is Exceeded

- n Number of T-nodes keeps increasing
- n Unable to converge to K-consistency at the end



$$l = 2$$

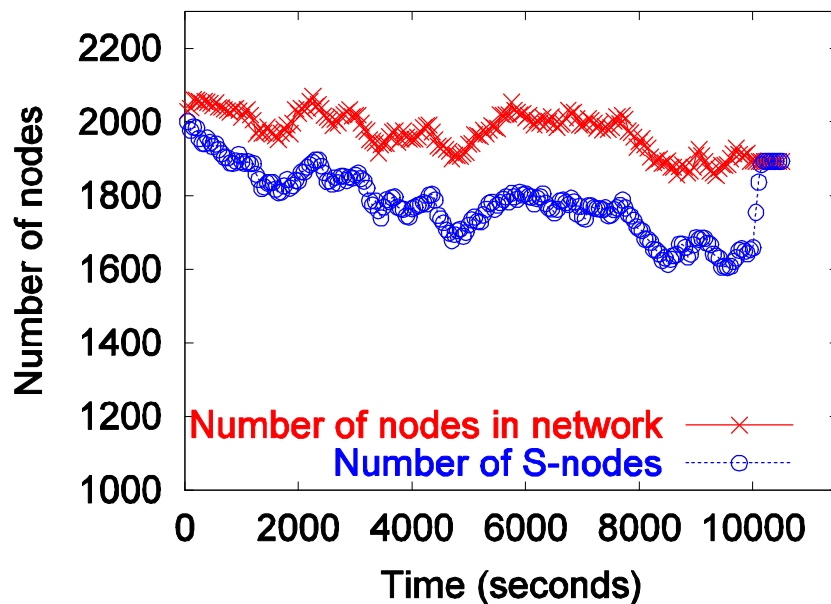
$$K=3$$

Timeout = 10sec

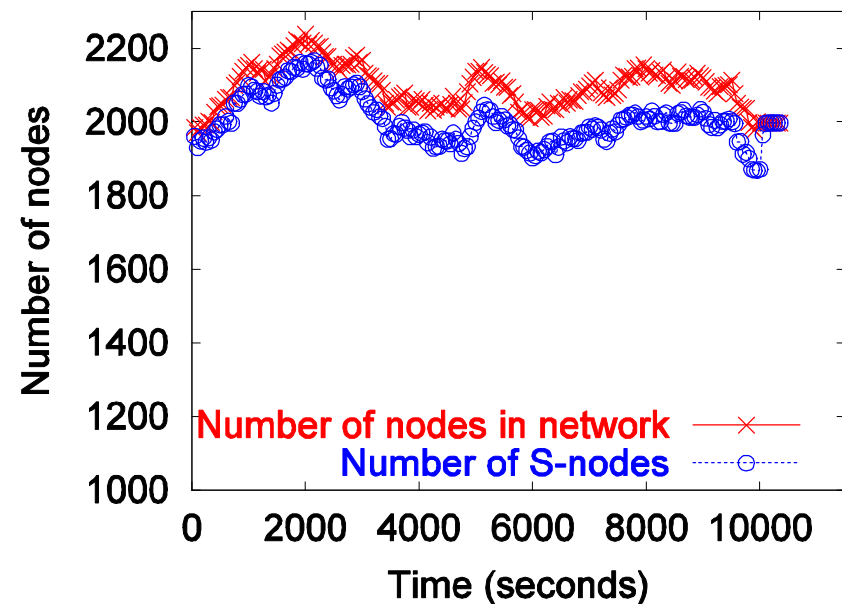
# How to Increase Join Capacity

- n Choose a smaller K or a smaller timeout value

$$l = 2$$



K=2, timeout = 10 sec



K=3, timeout = 5 sec

# Churn Experiment Summary

$\lambda$ (#joins/sec = #failures/sec)	0.25	0.5	0.75	1	1.25	1.5	2
number of joins	2413	5095	7621	10080	12474	15011	19957
number of failures	2473	5066	7423	9890	12468	14919	19960
% snapshots, 3-consistency-SAT	100	100	100	100	100	100	100
convergence to 3-consistency at end	yes	yes	yes	yes	yes	yes	no
convergence time (seconds)	150	200	400	350	450	400	—
% snapshots, 1-consistency	100	100	99.5	97.5	97.5	88.5	62
% snapshots, full connectivity	100	100	99.5	98	98	98.5	92
average %, connected S-D pairs	100	100	99.99998	99.99991	99.99993	99.99991	99.9996

Summary of churn experiments,  $n = 2000$ ,  $K = 3$ , timeout = 10 sec

---

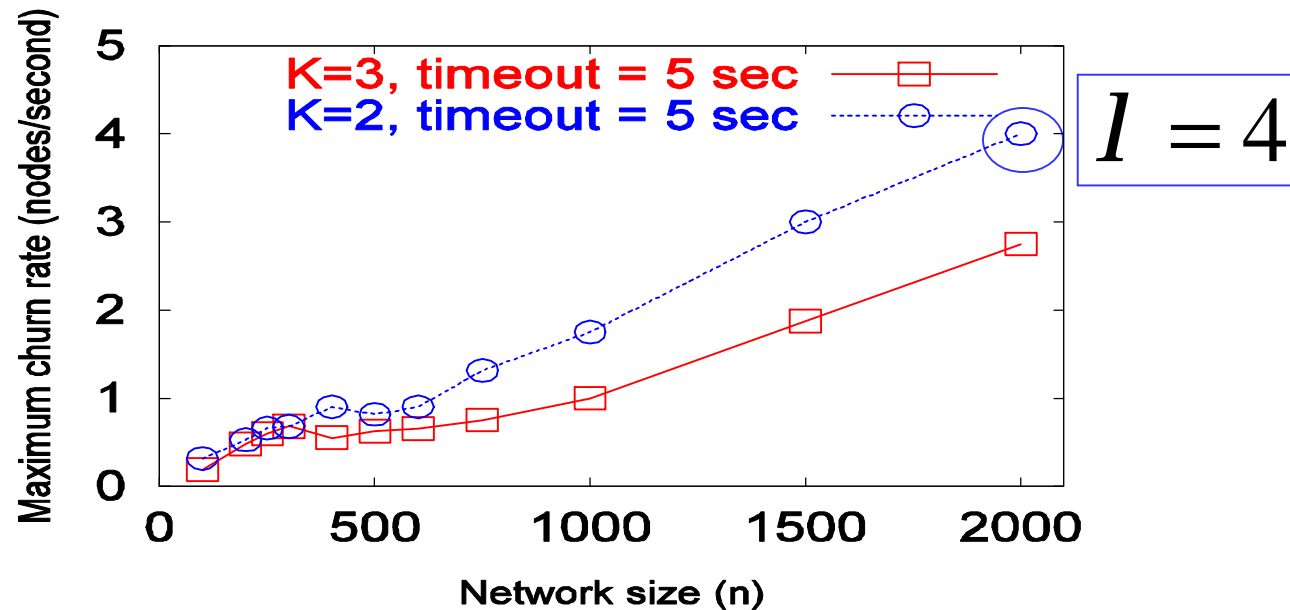
# Churn Experiment Summary

$\lambda$	0.25	0.5	0.75	1	1.25	1.5	2
Avg. % connected S-D pairs	100	100	99.99998	99.99991	99.99993	99.99991	99.99996

$n = 2000$ ,  $K=3$ , timeout = 10 sec

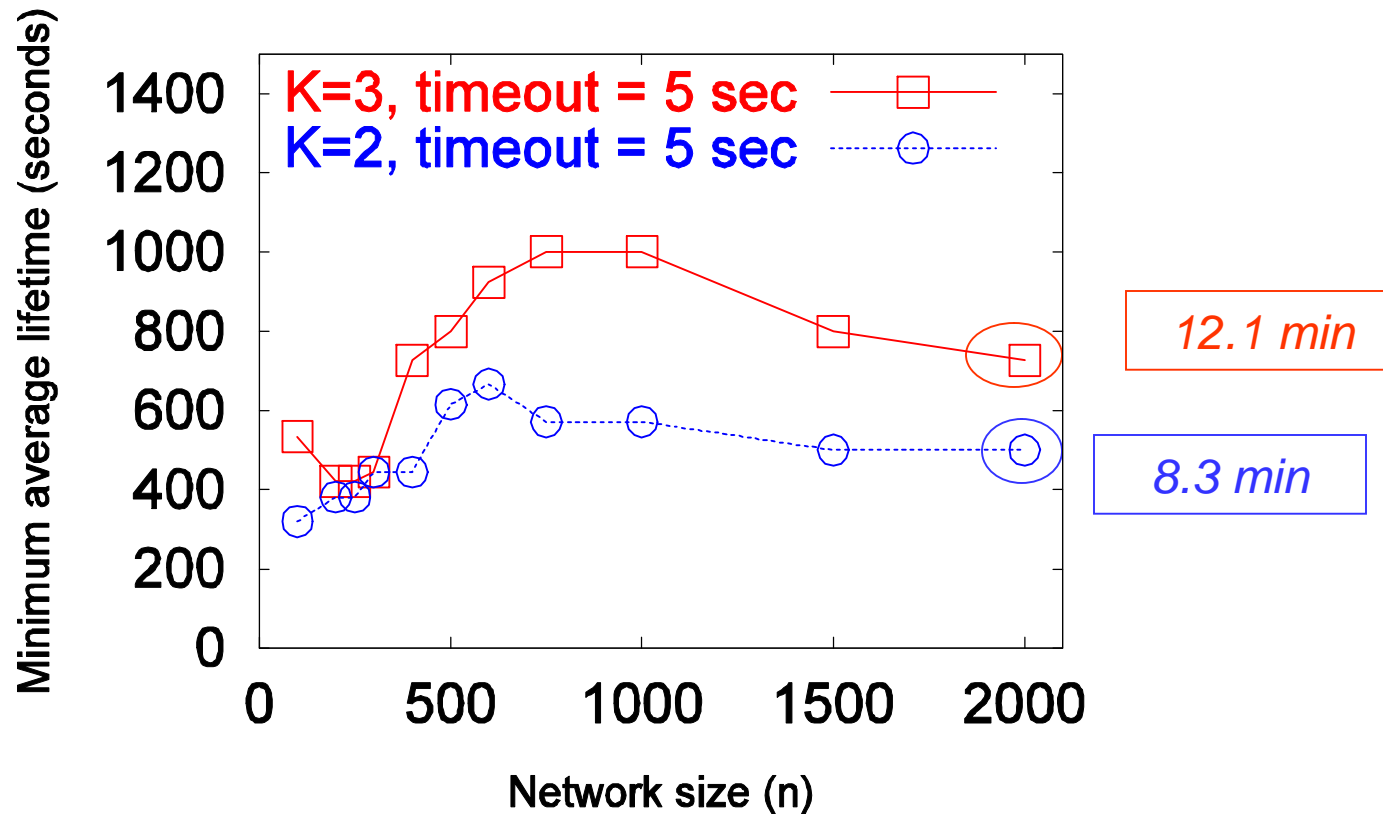
# Max Churn Rate vs. Network Size

- n Max sustainable churn rate increases at least linearly with network size
- n Stability improves when number of S-node increases
- n Smaller K leads to higher join capacity



# Min Avg. Lifetime vs. Network Size

- n The trend suggests:  
when  $n > 2000$ , avg. lifetime  $< 12.1$  min for  $K=3$ ,  
 $< 8.3$  min for  $K=2$



---

# Conclusions

- n Our protocols are effective, efficient, and stable, for average **node lifetime** as short as **8.3** min, given  $n=2000$ ,  $K=2$ ,  $timeout = 5sec$
- n Each network has a **join capacity** that
  - q upper bounds its join rate
  - q decreases when failure rate increases
  - q can be increased by a smaller  $K$  or a smaller timeout value
- n Recommended values for  $K$ :
  - q for network with a high churn rate,  $K=2$  or  $3$
  - q for network with a low churn rate,  $K=3$  or higher (say,  $4$  or  $5$ )