An Adaptive Stabilization Framework for DHT

Gabriel Ghinita, Yong Meng Teo
Department of Computer Science
National University of Singapore
{ghinitag,teoym}@comp.nus.edu.sg

Overview

- Background
- Related Work
- Adaptive Stabilization Framework
- Performance Evaluation
- Conclusion
- Q & A

Background

3

P2P Systems

- P2P characteristics
  - large-scale, highly-dynamic environments
  - nodes organize into an overlay network
    - routing state: pointers to other peers
  - design challenges
    - scalability, self-organization, fault-tolerance
- Structured P2P / Distributed Hash Tables (*DHT)*
  - overlay organization abides strict rules
  - lookup process is deterministic
  - upper bound on lookup performance
  - distributed storage, multicast, publish-subscribe

4

DHT Characteristics

- Virtualize node and data items to common key space
  - each peer is assigned a key space subset
- Hashtable interface: `Get(key)`,`Put(key,value)`
  - key *lookup* underlies every DHT operation
- Bounded routing state and lookup complexity
  - *logN / logN* widely-used compromise
- Implementations: Chord, CAN, Pastry, Tapestry, etc

5

Churn

- Nodes join and leave/fail freely
  - routing state inconsistent (routing constraints not satisfied)
  - failed lookup operations (incorrect/incomplete)
  - increased lookup path length
  - disconnection
- Measurement
  - *join rate* (global): # nodes joining/sec
  - *failure rate* (global) or *node lifetime* (local)

6

Stabilization

- Execution of corrective routines
  - check if pointer still alive (refresh)
  - check if pointer still abides constraints (re-pin)
  - may generate considerable *control* traffic
- Performance metrics
  - lookup failure → $\frac{\#\,failed\_lookups}{\#total\_lookups}$
  - average lookup path length
  - communication overhead → $\frac{\#stabilization\_msg}{\#user\_msg}$

7

Related Work

8

DHT Stabilization

- Periodic Stabilization (PS)
  - most widely used (Chord, Pastry, CAN, etc)
  - ad-hoc stabilization rate, no failure lookup bounds
  - unsuitable for variable churn
- Correction-on-use/Correction-on-change
  - limited to DKS DHT [El-Ansary et al, 2003]
- Physics-style approach [Krishnamurthy et al, 2005]
- Accordion [Li et al, 2005]
- Adaptive Stabilization (concept) [Mahajan et al, 2003]

9

Motivation

- Periodic stabilization has limitations
  - stabilization interval fixed at deployment
    - difficult to estimate proper stabilization rate
  - unsuitable for variable churn
- Implications
  - poor overlay performance
    - disconnection, failed lookups, increased path length
  - excessive control messages overhead

10

Chord

- Circular, 1-dimensional *m*-bit identifier space
- Routing state: three sections
  - Successor
  - Successor list
  - Finger table $\{f_i \mid f_i.id = succ(n.id+2^i)\}$
- Separate stabilization timer for each section
  - Successor pointer most frequently checked
  - Finger list least frequently updated
  - Stabilization timer setting expressed as *s/sl/f*

11

PS Evaluation

- Static scenario (no churn)
  - S1 = 1/3/10
  - 1000 nodes, 0.33 lookup/sec/node
  - 450% message overhead
- Dynamic scenario – “half-life”
  - 500 nodes perfect overlay, 500 new nodes join
  - 1000 nodes perfect overlay, 500 nodes fail
  - lookup rate 0.33/sec/node

12

Half-life Scenarios

Churn Rate	Stab. Rate	Lookup Failure %		Comm. Overhead %	
		Double	Halve	Double	Halve
1/sec	S1	0.5	3.2	421	457
	S2	1.1	4.9	211	203
	S3	1.7	6.5	135	139
2/sec	S1	0.8	5.1	414	462
	S2	1.9	9.2	208	205
	S3	2.8	12.3	143	151
5/sec	S1	1.8	8.7	407	445
	S2	2.7	12.7	218	209
	S3	3.6	23.7	154	154

S1 = 1/3/10 ; S2 = 3/5/20 ; S3 = 5/10/30

13

Adaptive Stabilization Framework

14

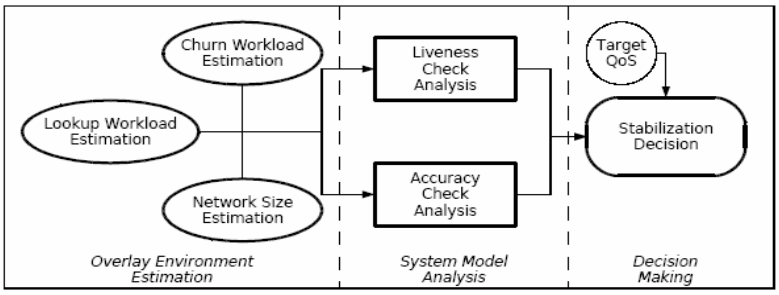
Objective

- Stabilization rate adapted to changing environment
- Design goals
  - decentralization: autonomous decision to stabilize
  - efficiency: maintain *nominal* network performance
  - low cost: minimize message overhead
- Stabilization rate adjusted based on
  - local estimation of *churn rate*
  - local estimation of *overlay size*
  - forwarding probability for each pointer
- DHT-flavor independent

15

Adaptive Stabilization Framework

- Input
  - observations on churn and lookup workload, overlay size
  - target QoS (lookup failure, lookup path length)
- Output
  - stabilization decision



16

AS Framework Overview

- Estimate churn and lookup workload
  - update at both external events and internal timer
- System model analysis
  - predict system behavior
- Stabilization decision
  - decision in agreement with the system model
  - decision based on QoS requirements
    - maximum allowed lookup failure

17

Advantages

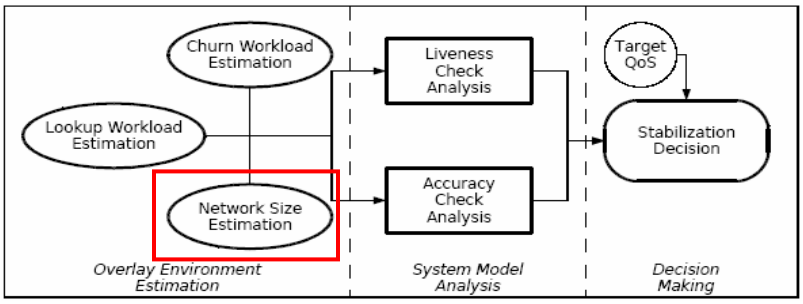
- Informed stabilization decision
  - as opposed to ad-hoc decision in PS
- Adapts to changing network conditions
- Stabilization decision correlated with QoS
- Tight stabilization control on a “per-pointer” basis
- DHT-protocol independent, to a large extent

18

Liveness and Accuracy

- Distinguish between two concepts:
  - **Liveness**: is a pointer in the routing table still alive?
  - **Accuracy**: is a pointer in the routing table still accurate?
- Liveness check: $\mathbf{P_{tout}}$
- Accuracy check: $\mathbf{P_{inacc}}$
- Cost :
  - Liveness: *O(1)* – message travels one overlay hop
  - Accuracy: *O(logN)* - message travels *logN* hops

19



20

Overlay Size Estimation

- Based on density of nodes in identifier space
  - assume node IDs are evenly distributed
    - assumption holds if ***Consistent Hashing*** is used
  - nodes know the IDs of *P* predecessors and *S* successors
  - overlay size estimation is

$$Size = \frac{Identifier\_Space\_Size}{ID_{last\_succ} - ID_{first\_pred}} \times (P + S)$$

Overlay Size Estimation (2)

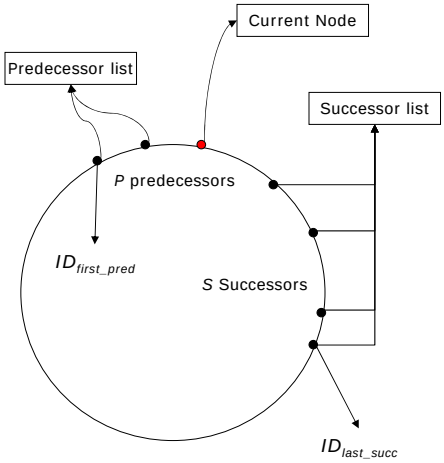
- *P=2, S=4*
- *Expected distance between nodes*

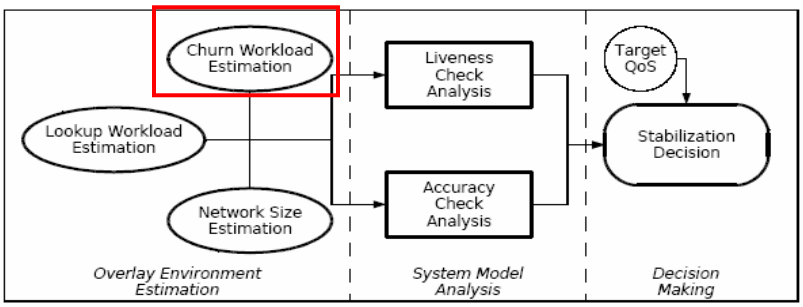
$$d = \frac{ID_{last\_succ} - ID_{first\_pred}}{P + S}$$

- *Overlay size is*

$$Size = \frac{Identifier\_Space\_Size}{d}$$

$$= \frac{Identifier\_Space\_Size}{ID_{last\_succ} - ID_{first\_pred}} \times (P + S)$$



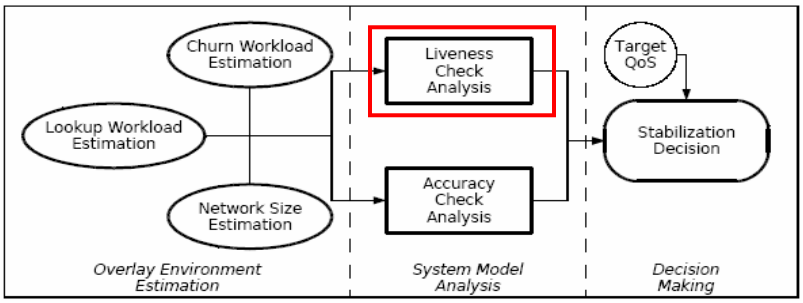


23

Churn Rate Estimation

- *Stabilization rate* adjusted in response to *churn rate*
- Intuitively,
  - large churn rate → faster stabilization rate
  - small churn rate → slower stabilization rate
- Each node timestamps its routing table entries
  - T_s^p – time pointer p was last known to be alive
  - T_{join}^p – time pointer p joined the network
- Global churn rate computation factors in overlay size

24



25

Liveness Check Analysis

- Each node performs analysis locally
  - each pointer is considered separately
  - assume lookup destinations uniformly distributed
  - determine ***forward probability*** for each pointer
    - factor in relative importance of each pointer
  - for pointer *p*

$$P_{tout}^{p} = P_{fwd}^{p} \times P_{dead}^{p}$$

26

Formulation of P_{dead}

- Depends on node join/failure workload only
  - independent of DHT flavor, routing algorithm, etc
- Assume exponential distribution of node lifetime
  - other distribution easily supported

$$P_{dead}^{p}(t) = 1 - e^{-\mu(t - T_s^p)}$$

μ=estimated average node lifetime

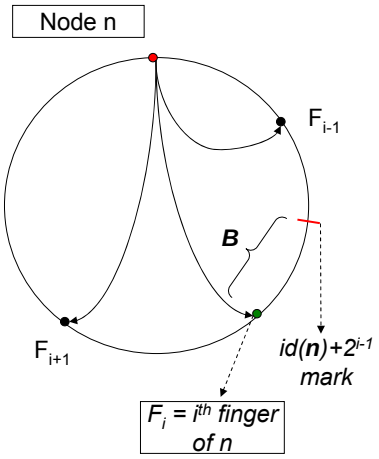
t = current time

T_i^p = time when *p* was last checked

27

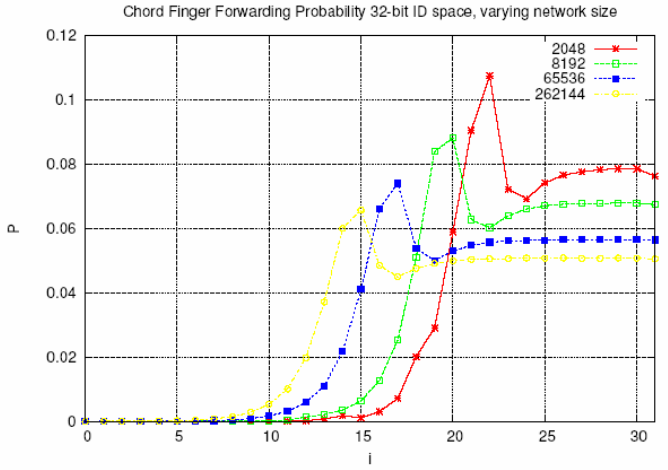
Finger Forwarding Probability

- DHT-flavor dependent
- Chord: hops not exact power of 2
  - bias on average $B=2^{m-1}/N$
- P_{fwd} varies with index
  - low for close-by fingers
  - highest for *i* = *logB*
    - successor pointer
  - saturates for high index



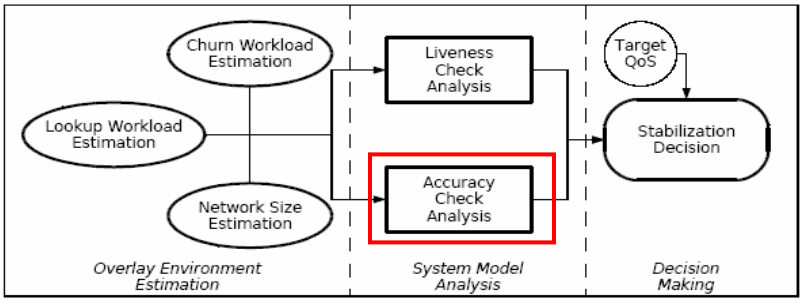
28

Formulation of P_{fwd}



Measured Forwarding Probability for Chord fingers

29



30

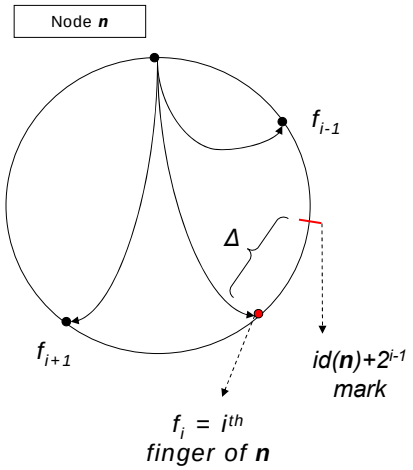
Accuracy Check Analysis

- Each node performs analysis locally
  - each pointer is considered separately
  - DHT-flavor dependent
  - assume joining nodes' IDs uniformly distributed
  - analysis for each finger
    - probability of node join that affects DHT constraints
    - estimate gain in correctness with a better pointer
      - might not be worth it to re-pin finger

31

Dealing with joins

- P_{inacc}^{i} for f_i of node n
  - same as P[*join in interval Δ*]
  - based on estimate join rate
- P_{inacc}^{i} low
  - join in *Δ* unlikely
  - performance gain after repin is low



32

Formulation of P_{inacc}

- P_{inacc} only affected by node join rate
- P_{inacc} NOT affected by node failure rate

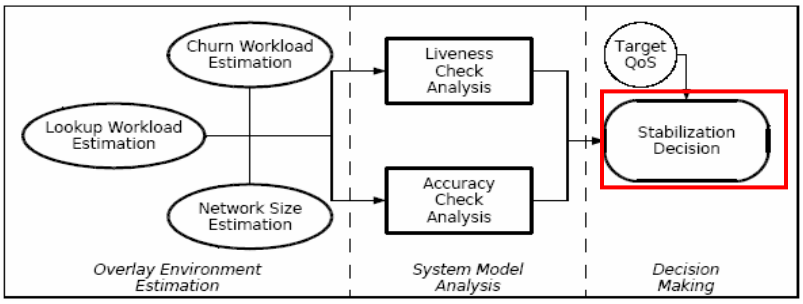
$$P^{i}_{inacc} = \frac{f_i.id - n.id - 2^i}{2^m} \times \lambda \times (t - T^{i}_{pin})$$

λ = estimated arrival rate of new nodes

t = current time

T_i^{pin} = time when i^{th} finger was last pinned (looked up)

33



34

Stabilization Decision

- Factors to consider
  - relative importance of different pointers
  - upper and lower bounds of the stabilization interval
  - relative impact of different type of events: join/fail
- Evaluate probability of finding node *p* alive at time *t*
  - last stabilization is origin of time axis
  - stabilize at τ s.t.

    P[p_dead_before_ τ] < threshold

35

Example

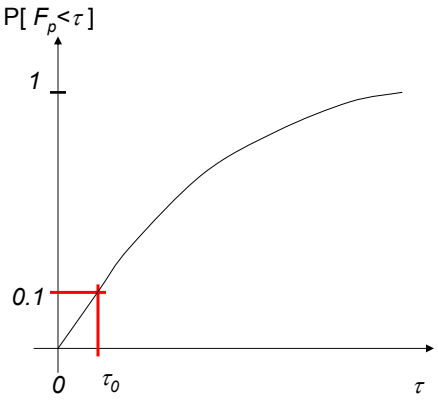
- Lifetime model: exp.distribution with mean *μ =10/sec*
- F_p = moment node pointed by current pointer p fails

$$P\langle F_p < \tau \rangle = 1 - e^{-\mu\tau}$$

- bound for 10% failure ratio

$$P\langle F_p < \tau \rangle < 0.1$$

$$\tau_0 = -\frac{1}{\mu \ln 0.9}$$



36

Setting Thresholds

- Desired lookup failure ratio F_1
  - average path length is *logN/2*
  - $P_{tout} < F_1^{2/logN}$
- Desired disconnection probability F_2
  - *O(logN)* successors
  - $P_{tout} < F_2^{1/O(logN)}$

37

Performance Evaluation

38

Experimental Settings

- *AS* prototype implemented on top of *p2psim*
- Constant churn rate
  - three join/failure rate: 1, 2 and 5/sec
  - target lookup failure set at 3%
- Variable churn rate
  - two "steady-states" with low/moderate churn
    - 500 and 2500 nodes respectively
  - two periods of "peak" churn
    - considerable variation in size
  - target lookup failure set at 2%

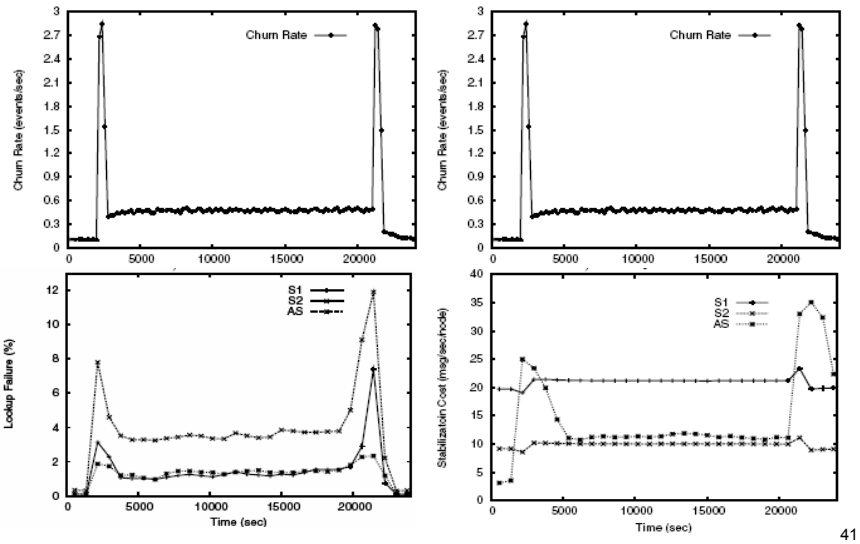
39

Constant Churn Rate

Churn Rate	Stab. Rate	Lookup Failure %		Comm. Overhead %	
		Double	Halve	Double	Halve
1/sec	S1	0.5	3.2	421	457
	S2	1.1	4.9	211	203
	S3	1.7	6.5	135	139
	AS	0.9	2.9	141	142
2/sec	S1	0.8	5.1	414	462
	S2	1.9	9.2	208	205
	S3	2.8	12.3	143	151
	AS	0.9	3.1	296	305
5/sec	S1	1.8	8.7	407	445
	S2	2.7	12.7	218	209
	S3	3.6	23.7	154	154
	AS	1.1	3.4	489	552

40

Variable Churn Rate



41

Performance Evaluation

- AS outperforms PS at all accounts
  - lookup performance
  - stabilization cost
- AS superior in all test scenarios
  - constant churn rate
  - variable churn rate
    - AS shows good reaction to changes in churn rate
  - AS achieves target QoS!
- AS has superior performance-cost tradeoff
  - safeguard against extreme scenarios

42

Conclusions

- We propose an *adaptive stabilization framework* DHT
  - identify the fundamental principles behind DHT stabilization
  - devise mechanisms to estimate environment dynamism
  - devise a QoS-driven decision-making mechanism
  - DHT-independent to a considerable extent
- To do
  - extend framework to suit other churn models
  - factor in lookup workload distribution
  - relax assumptions to generalize model applicability
  - employ machine-learning elements - learn from history
  - develop robustness for tuning system parameters

Q & A