

Symphony

Distributed Hashing in a Small World

Gurmeet Singh Manku
Stanford University

with **Mayank Bawa** and **Prabhakar Raghavan**

Presented By Xiaorong Zhou

Goal: How can **thousands of hosts** cooperatively maintain a **large hash table** in a **completely decentralized** fashion?

Symphony USITS, 10 March 2003

Acknowledgement

The following slides are borrowed from the author's talk at USITS 2003.

Symphony USITS, 10 March 2003

DHTs: The Big Picture

Load Balance

"How do we splice the hash table evenly?"
Nodes choose their ID in the hash space uniformly at random.

Topology Establishment

"How do we route with small state per node?"

Deterministic
(CAN/Chord)
(Pastry/ Tapestry)
Randomized
(Symphony)

Caching, Hotspots, Fault Tolerance, Replication, ...

x --- x

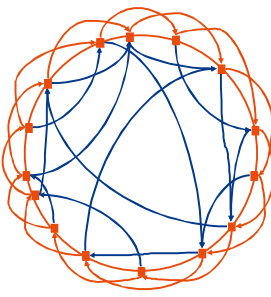
Symphony USITS, 10 March 2003

Spectrum of DHT Protocols

	Protocol	#links	latency
Deterministic Topology	CAN	$O(\log n)$	$O(\log n)$
	Chord	$O(\log n)$	$O(\log n)$
Partly Randomized Topology	Viceroy	$O(1)$	$O(\log n)$
	Tapestry	$O(\log n)$	$O(\log n)$
	Pastry	$O(\log n)$	$O(\log n)$
Completely Randomized Topology	Symphony	$2k+2$	$O((\log^2 n)/k)$

Symphony USITS, 10 March 2003

Symphony in a Nutshell



Nodes arranged in a *unit circle* (perimeter = 1)

Arrival \rightarrow Node chooses *position* along circle uniformly at random

Each node has *1 short link* (next node on circle) and *k long links*

Adaptation of *Small World* Idea: [Kleinberg00]
 Long links chosen from a probability distribution function: $p(x) = 1 / (x \log n)$ where $n = \#nodes$.

Simple greedy routing:
"Forward along that link that minimizes the absolute distance to the destination."

Average lookup latency = $O((\log^2 n) / k)$ hops

Fault Tolerance:
No backups for long links! Only short links are fortified for fault tolerance.

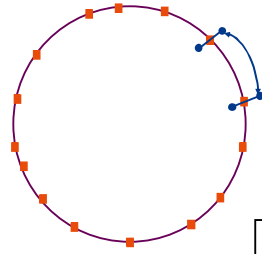
■ node ← long link → short link

A typical Symphony network

Symphony USITS, 10 March 2003

Network Size Estimation Protocol

Problem: **What is the current value of n , the total number of nodes?**

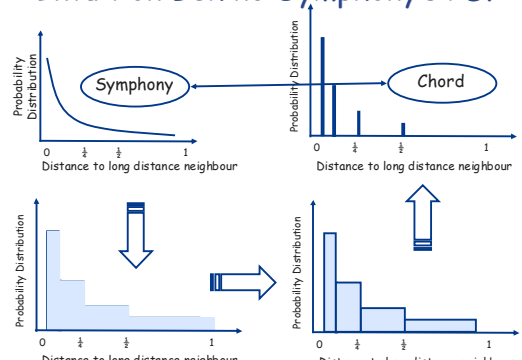


x = Length of arc
 $1/x$ = Estimate of n
 (Idea from Viceroy)

- 3 arcs are enough.
 - Re-linking Protocol not worthwhile.

Symphony USITS, 10 March 2003

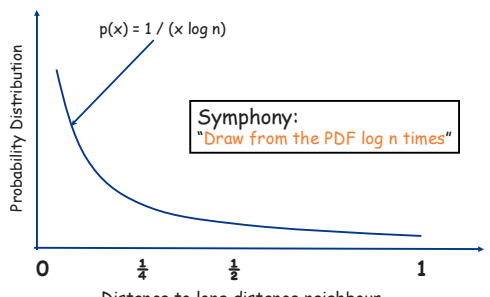
Intuition Behind Symphony's PDF



The top row shows the **Symphony** PDF (a smooth curve) and the **Chord** PDF (a histogram). The bottom row shows the Chord PDF being approximated by the Symphony PDF.

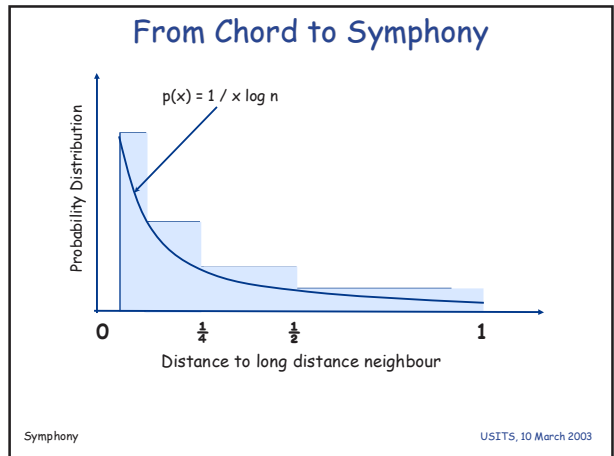
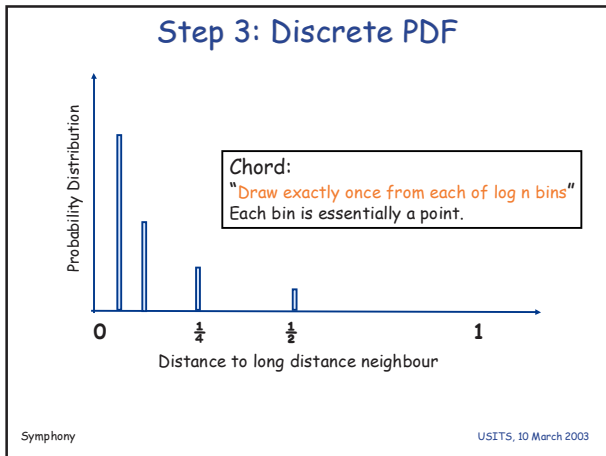
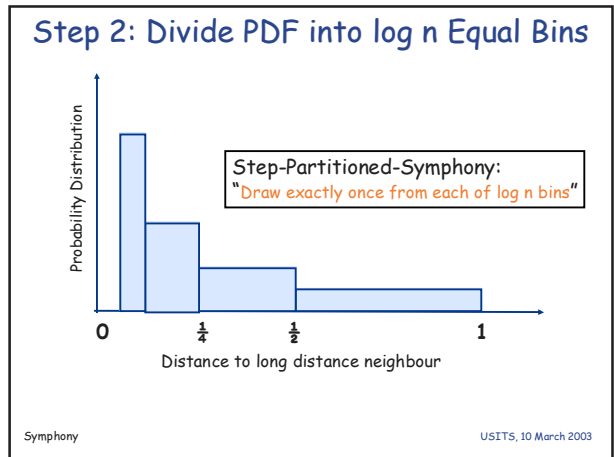
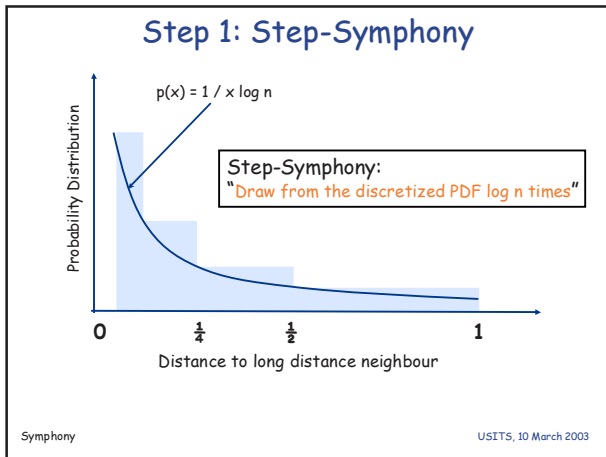
Symphony USITS, 10 March 2003

Step 0: Symphony



Symphony:
"Draw from the PDF log n times"

Symphony USITS, 10 March 2003



Two Optimizations

Bi-directional Routing

- Exploit both **outgoing** and **incoming** links!
- Route to the neighbor that **minimizes absolute distance** to destination
- Reduces avg latency by 25-30%

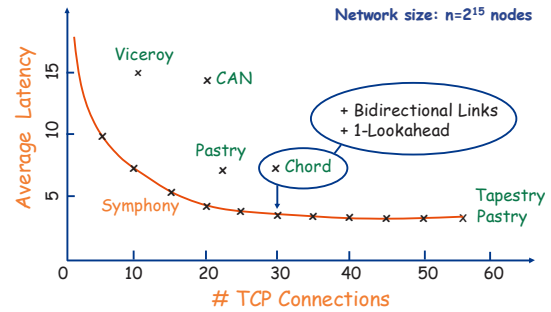
1-Lookahead

- List of neighbor's neighbors
- Reduces avg latency by 40%

Symphony

USITS, 10 March 2003

Latency vs State Maintenance



Many more graphs in the paper.

Symphony

USITS, 10 March 2003

Why Symphony?

1. Low state maintenance

- Low degree --> **Fewer pings/keep-alives**, less control traffic
- Low degree --> Distributed locking and coordination overhead over smaller sets of nodes
- Low degree --> **Smaller bootstrapping time** when a node joins
- Low degree --> **Smaller recovery time** when a node leaves

2. Fault tolerance

- Only short links are bolstered. **No backups for long links!**

3. Smooth out-degree vs latency tradeoff

- Only protocol that offers this tuning knob even at run time!
- Out-degree is not fixed** at runtime, or as a function of network size.

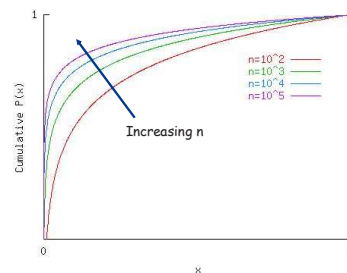
4. Flexibility and support for heterogeneity

- Different nodes can have different #links!**

Symphony

USITS, 10 March 2003

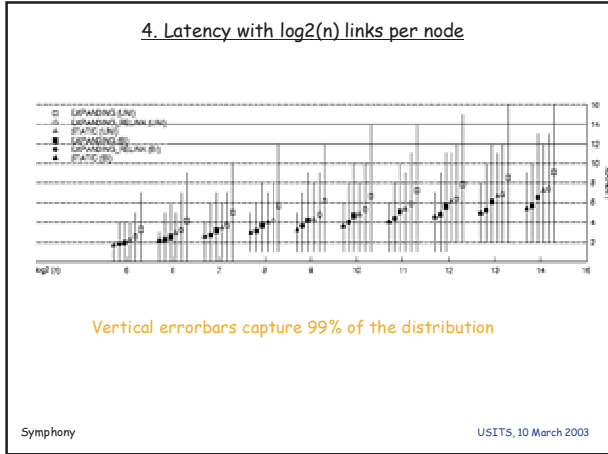
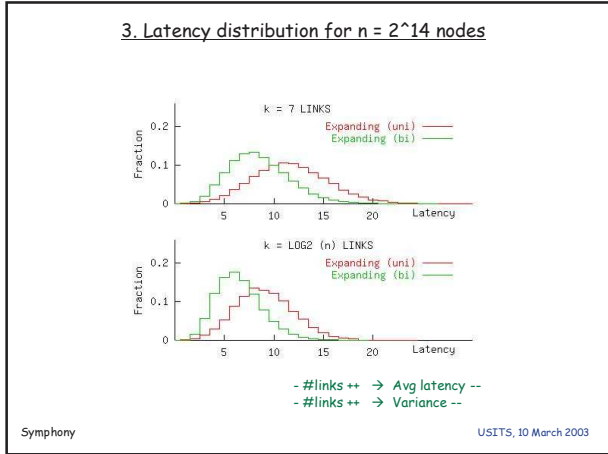
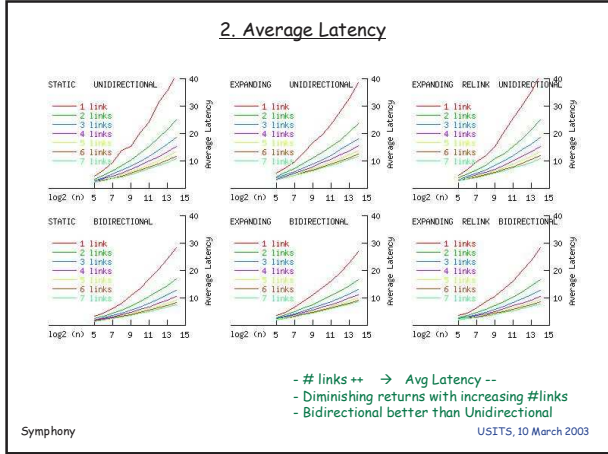
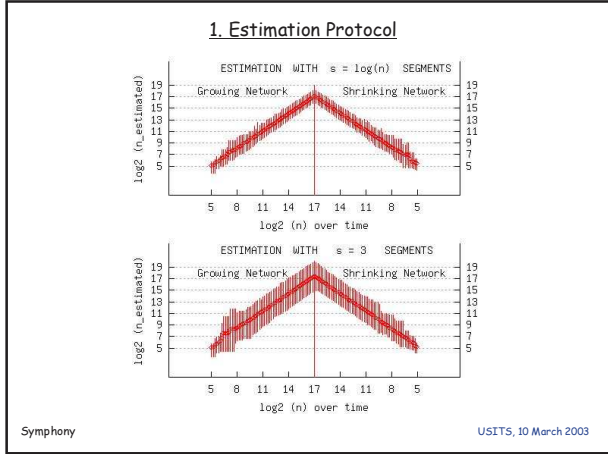
Family of Harmonic Distributions



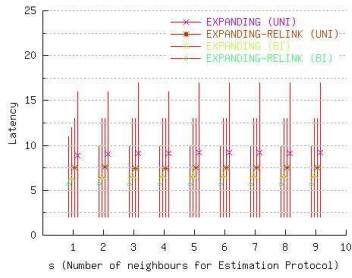
Cumulative PDF: $P(x) = \log xn / \log n$ for x in $[1/n, 1]$
 PDF: $p(x) = 1 / x \log n$ for x in $[1/n, 1]$

Symphony

USITS, 10 March 2003



5. Choice of s for Estimation Protocol

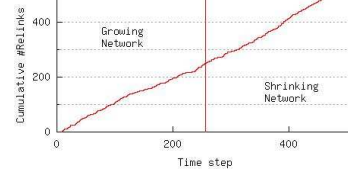


- No change in avg latency with increasing s
 - Vertical errorbars capture 99% of distribution

Symphony

USITS, 10 March 2003

6. Cumulative #relinks for expanding network

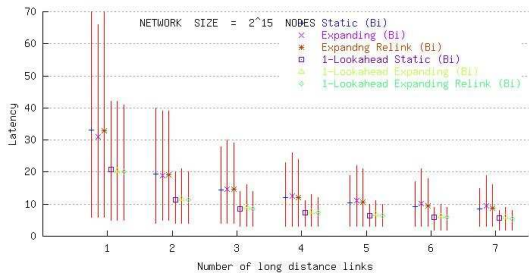


- Exactly one node arrives per timestep

Symphony

USITS, 10 March 2003

7a. Impact of 1-Lookahead

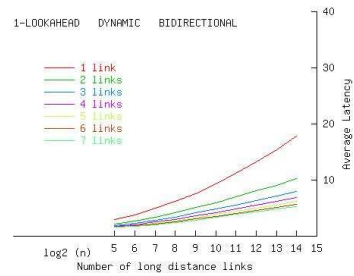


- 1-Lookahead diminishes avg latency by roughly 40%

Symphony

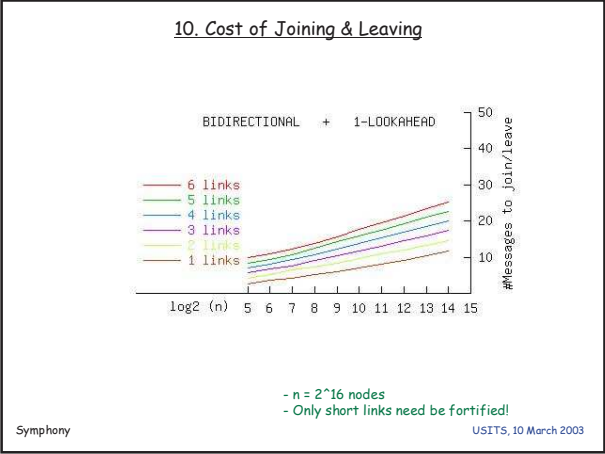
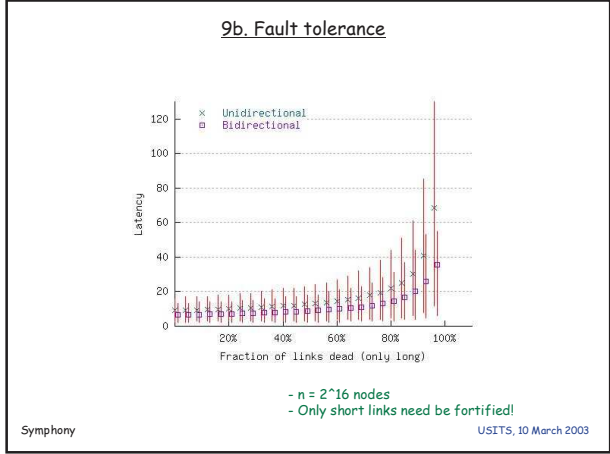
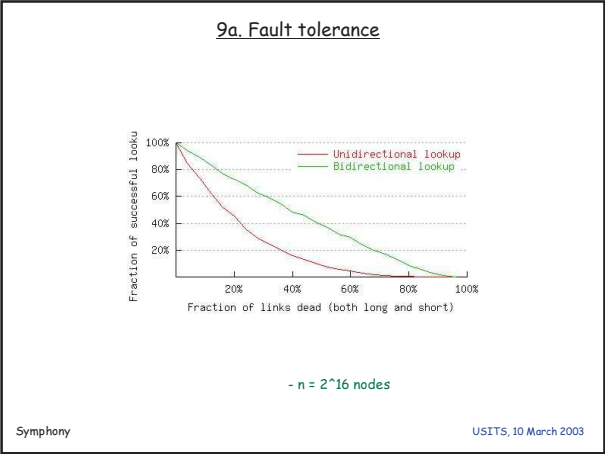
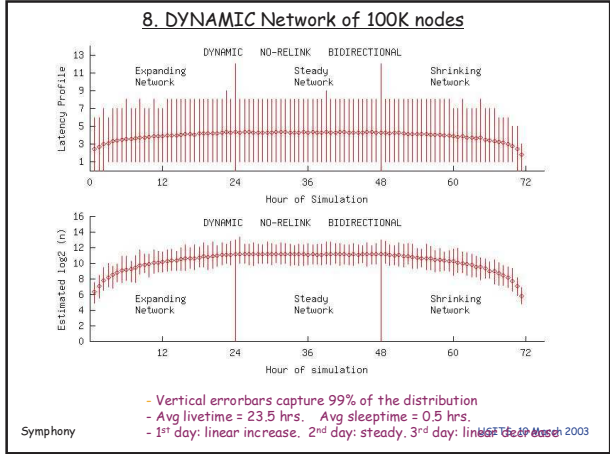
USITS, 10 March 2003

7b. Impact of 1-Lookahead

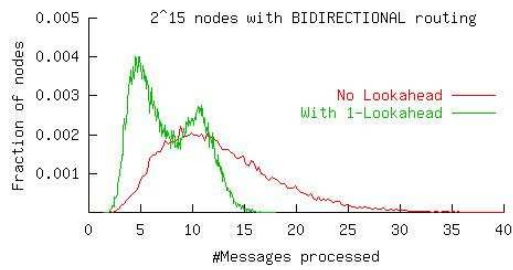


Symphony

USITS, 10 March 2003



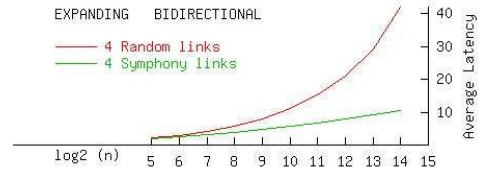
11. Bandwidth Profile



Symphony

USITS, 10 March 2003

12. Comparison with Uniformly Random Links



Symphony

USITS, 10 March 2003

- Resources
 - The author's homepage
 - <http://www.cs.stanford.edu/~manku>
 - Stanford Peers
 - <http://www-db.stanford.edu/peers/>

Question?

Symphony

USITS, 10 March 2003