High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two

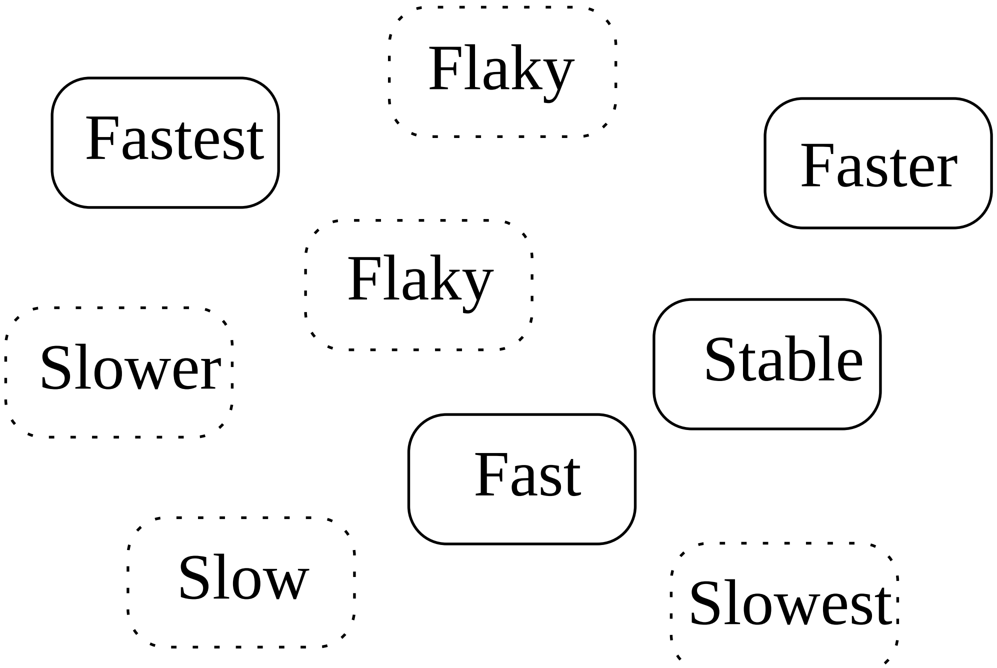
Charles Blake

Rodrigo Rodrigues

cb@mit.edu, rodrigo@lcs.mit.edu.

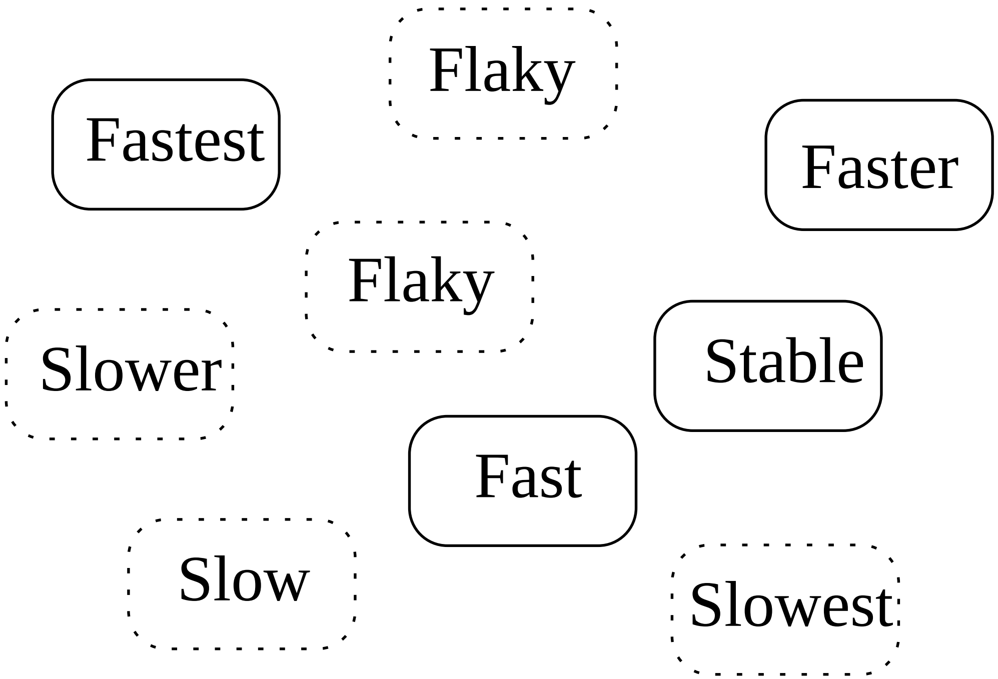
LCS at MIT

The P2P Dream



10..100's of GB/node of
Idle, Cheap Disk

The P2P Dream

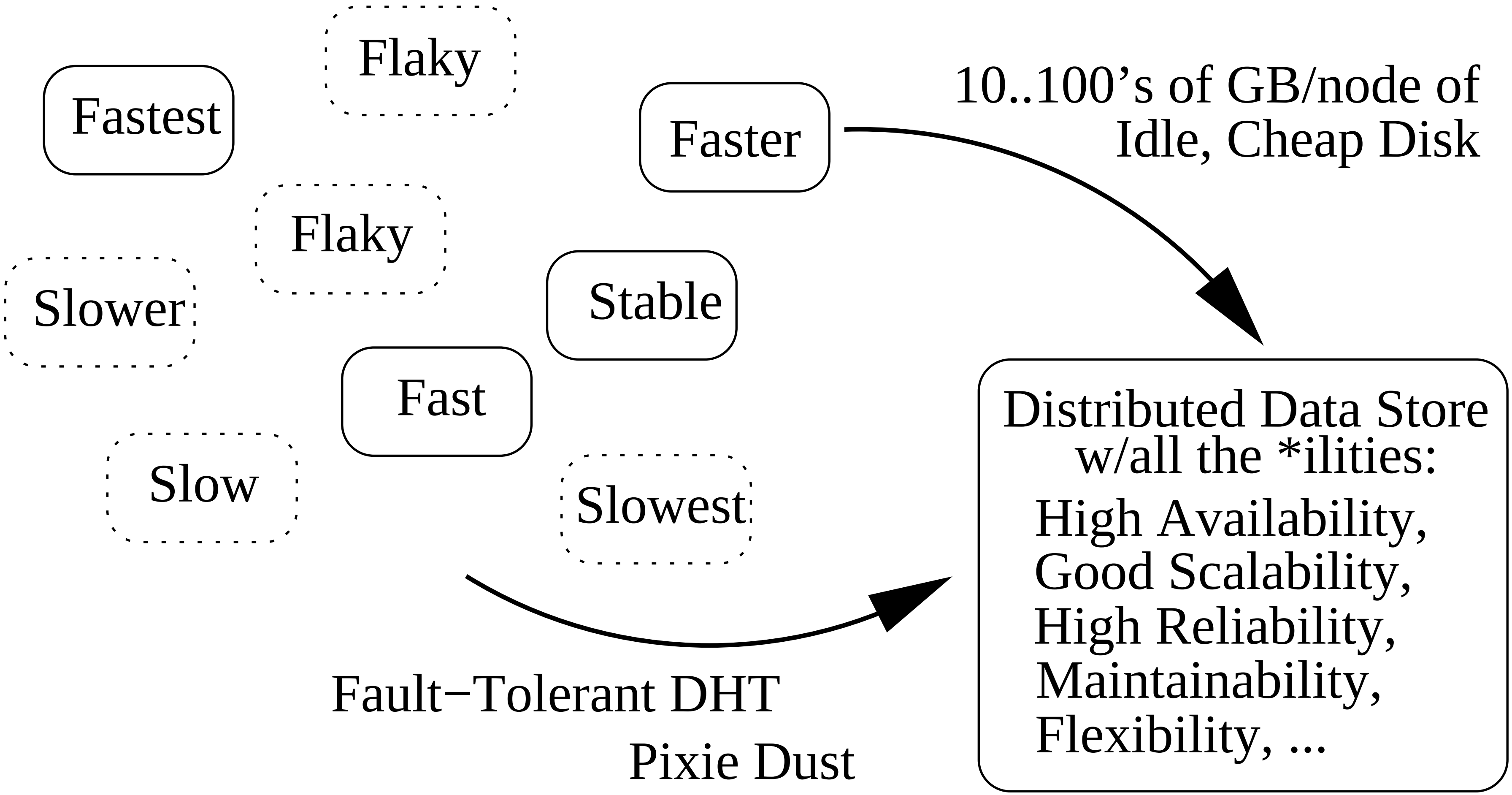


10..100's of GB/node of
Idle, Cheap Disk

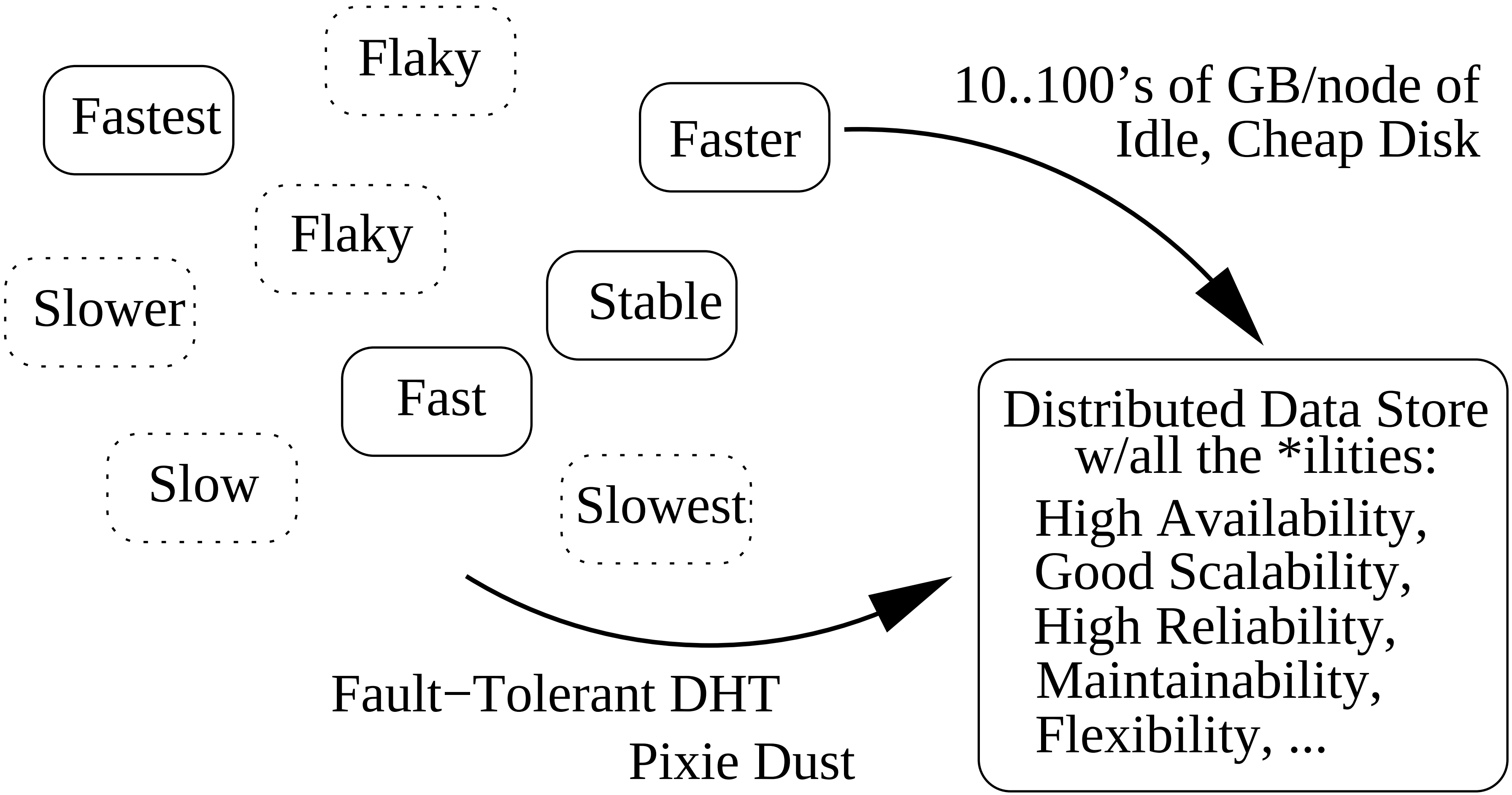
Fault−Tolerant DHT

Pixie Dust

The P2P Dream



The P2P Dream



How realistic is this dream?

Talk Overview

- Basic Scenario
- *Simplified* Model $\longrightarrow$ The Bad News
- Elaborate on Simplifications
- Address Partial Availability
- Hardware Trends
- Gnutella Statistics
- Questions about Basic P2P Premises

Basic Scenario

- N nodes (N probably $\gg$ 10,000)
  ...using similar bandwidth & disk
  ...cooperatively serving D bytes of data
  ...placed randomly about the Internet

Basic Scenario

- N nodes (N probably $\gg$ 10,000)
  ...using similar bandwidth & disk
  ...cooperatively serving D bytes of data
  ...placed randomly about the Internet
- Members can *leave*! (true data loss)

$$\begin{aligned} P(Leave)/Time &= Leaves/Time/N \\ &= 1/Lifetime \end{aligned}$$

Basic Scenario

- N nodes (N probably $\gg$ 10,000)
  ...using similar bandwidth & disk
  ...cooperatively serving D bytes of data
  ...placed randomly about the Internet
- Members can *leave*! (true data loss)

$$\begin{aligned} P(Leave)/Time &= Leaves/Time/N \\ &= 1/Lifetime \end{aligned}$$

- Storage *promise* $\Rightarrow$ Redundancy promise
  $\Rightarrow$ data must move as members leave!
  $\Rightarrow$ lower bound on bandwidth usage

BW for Redundancy Maintenance

- Assume average system size, N, stable

BW for Redundancy Maintenance

- Assume average system size, N, stable
- Join = Leave forever rate = 1/Lifetime

BW for Redundancy Maintenance

- Assume average system size, N, stable
- Join = Leave forever rate = 1/Lifetime
- Leaves induce redundancy replacement
  replacement size $\times$ replacement rate

BW for Redundancy Maintenance

- Assume average system size, N, stable
- Join = Leave forever rate = 1/Lifetime
- Leaves induce redundancy replacement
  replacement size $\times$ replacement rate
- Joins cost the same

BW for Redundancy Maintenance

- Assume average system size, N, stable
- Join = Leave forever rate = 1/Lifetime
- Leaves induce redundancy replacement
  replacement size $\times$ replacement rate
- Joins cost the same

$$\therefore \quad Maintenance\,BW > 2 \times Space/Lifetime$$

Space/node < $\frac{1}{2}\times$ BW/node $\times$ Lifetime

QUALITY WAN STORAGE SCALES WITH WAN BANDWIDTH & MEMBER QUALITY

This Scaling is a Problem

- maintenance BW $\approx$ 200 Kbps
- lifetime = Median 2001-Gnutella session
  = 1 hour

$$\begin{aligned} \textbf{served space} &= 90\ MB/node \\ &\ll \textbf{donatable storage}! \end{aligned}$$

This cost is *Conservative*

- We assume served data is *totally static*

This cost is *Conservative*

- We assume served data is *totally static*
- Serious “promise” $\Rightarrow$ *worst case*

This cost is *Conservative*

- We assume served data is *totally static*
- Serious “promise” $\Rightarrow$ *worst case*
- Identical space & bandwidth

This cost is *Conservative*

- We assume served data is *totally static*
- Serious “promise” $\Rightarrow$ *worst case*
- Identical space & bandwidth
- Fixed population

This cost is *Conservative*

- We assume served data is *totally static*
- Serious "promise" $\Rightarrow$ *worst case*
- Identical space & bandwidth
- Fixed population
- Load-balance, Popular data more available
  Additional redundancy $\longrightarrow$ *more* BW

This cost is *Conservative*

- We assume served data is *totally static*
- Serious “promise” $\Rightarrow$ *worst case*
- Identical space & bandwidth
- Fixed population
- Load-balance, Popular data more available
  Additional redundancy $\longrightarrow$ *more* BW
- Downtime isn’t *leaving* forever

Partial Availability

Let $upfrac \equiv P(typical\ node\ is\ up)$

- N bigger – More “peers”, some down

Partial Availability

Let $upfrac \equiv P(typical\ node\ is\ up)$

- N bigger – More “peers”, some down
- Lifetime longer – Peers less dynamic

Partial Availability

Let $upfrac \equiv P(typical\ node\ is\ up)$

- N bigger – More "peers", some down
- Lifetime longer – Peers less dynamic

- Less effective bandwidth:
  $B \rightarrow upfrac \times B$

Partial Availability

Let $upfrac \equiv P(typical\ node\ is\ up)$

- N bigger – More “peers”, some down
- Lifetime longer – Peers less dynamic

- Less effective bandwidth:
  $B \rightarrow upfrac \times B$
- Redundancy for a promise must be larger
  6 nines — $P(down) \sim 1/million$
  multiple copies: $redun \sim 15/upfrac$
  optimal coding: $redun \sim 3/upfrac$

Partial Availability

Let $upfrac \equiv P(typical\ node\ is\ up)$

- N bigger – More "peers", some down
- Lifetime longer – Peers less dynamic

- Less effective bandwidth:
  $B \rightarrow upfrac \times B$
- Redundancy for a promise must be larger
  6 nines — $P(down) \sim 1/million$
  multiple copies: $redun \sim 15/upfrac$
  optimal coding: $redun \sim 3/upfrac$

$$\mathbf{Data} < \tfrac{1}{6} \times \mathbf{upfrac}^2 \times \mathbf{Lifetime} \times \mathbf{BW}$$

Availability+Edge BW Limit Storage

Put in “fantasy” numbers for grass-roots P2P

- All 10 Million cable modems in the US
  – $100\ Kbps$ “spare” upstream BW
    – $50\ Kbps$ for redundancy maintenance
    – $50\ Kbps$ for downloads
- 100 GB/node $\Rightarrow$ 1 million TB storage
- 25% node availability ($redun \approx 12X$)
- 1 week *average* lifetime

Availability+Edge BW Limit Storage

Put in “fantasy” numbers for grass-roots P2P

- All 10 Million cable modems in the US
  – $100\ Kbps$ “spare” upstream BW
    – $50\ Kbps$ for redundancy maintenance
    – $50\ Kbps$ for downloads
- 100 GB/node $\Rightarrow$ 1 million TB storage
- 25% node availability ($redun \approx 12X$)
- 1 week *average* lifetime

- Usable Space/node = $500\ MB$ = 0.5%
- *Unique* Servable Data = $400\ TB$ = 0.04%

Wait — It Gets Worse

Idle Storage Grows Much Faster than Idle Bandwidth

Year	Disk	Speed (Kbps)	Days to send a disk
1990	60 MB	9.6	0.6
1995	1 GB	33.6	3
2000	80 GB	128	60
2005	*0.5 TB*	*384*	*120*

Utilization will likely get worse

Fantasy upfrac's or Strawman?

Spring 2001: 50% (Saroiu, Gummadi, Gribble)

Spring 2003: 15% (Study we just did)

10X more hosts in 2003 than 2001.

Volunteer proliferation $\rightarrow$ availability decline?

967 of 100,000 Gnutella hosts $\rightarrow$ 10% uptime
- individually have $upfrac > 99\%$
- probably more than 10% of BW served

Admission Control + Incentives

- Only admit "reliable nodes"

Admission Control + Incentives

- Only admit "reliable nodes"
- Incentivize nodes staying up
  (high availability alone is not enough)

Admission Control + Incentives

- Only admit "reliable nodes"
- Incentivize nodes staying up
  (high availability alone is not enough)
- Incentivize long lifetimes
  Things that might make lifetimes longer
  seem to make availability lower

Admission Control + Incentives

- Only admit "reliable nodes"
- Incentivize nodes staying up
  (high availability alone is not enough)
- Incentivize long lifetimes
  Things that might make lifetimes longer
  seem to make availability lower

Yes, we can allow/elicit only great nodes, but...

This alters a dynamism/flakiness assumption permeating current evangelical conceptions!

What are we lusting after, exactly?

The 10% reliable Gnutella core could be mimicked by a half-dozen universities.

Cross WAN Bandwidth is the primary cost of WAN-distributed storage

BW for million's of cable modems
$\approx$ BW for hundreds of universities

The unreliable masses only command a small fraction of the world's *SERVICE BW*

Concluding Questions/Issues

- We don’t really know what people will do
  Experience suggests 1 month *generous*
  What resources do millions of flaky users
  really bring to the table anyway?

Concluding Questions/Issues

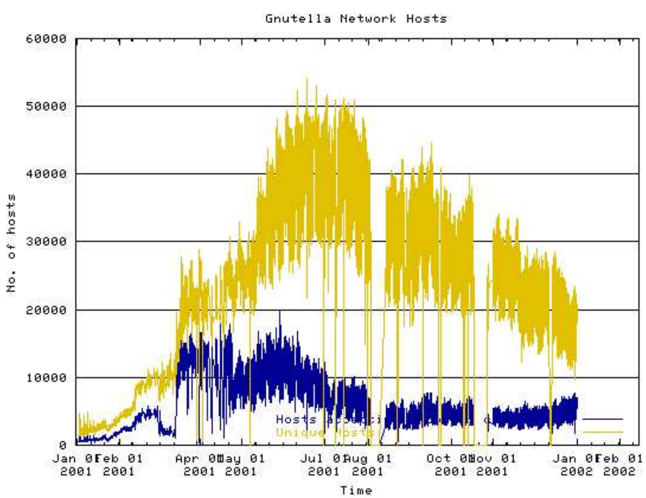
- We don't really know what people will do
  Experience suggests 1 month *generous*
  What resources do millions of flaky users *really* bring to the table anyway?
- Availably scaling randomly placed data needs *stable/available/high BW* hosts
  (Whither small-state lookup optimizations?)

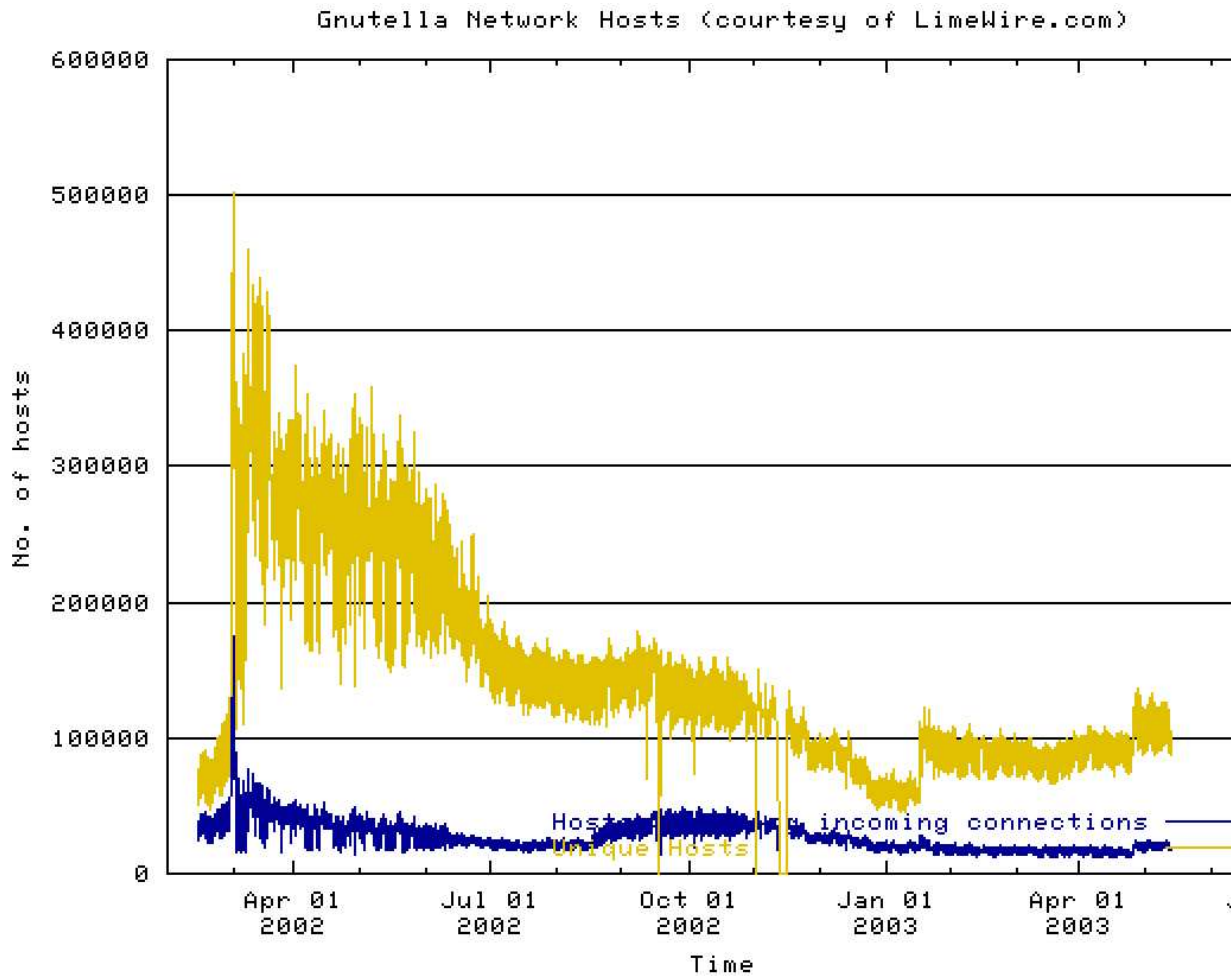
Concluding Questions/Issues

- We don’t really know what people will do
  Experience suggests 1 month *generous*
  What resources do millions of flaky users *really* bring to the table anyway?
- Availably scaling randomly placed data needs *stable/available/high BW* hosts
  (Whither small-state lookup optimizations?)
- If low availability parts are unavoidable, do we give up aggregate availability?
  ...or give up data scale/disk utilization?
  (why use millions when dozens might do?)

Support Slides

2.5 Years of Gnutella Behavior

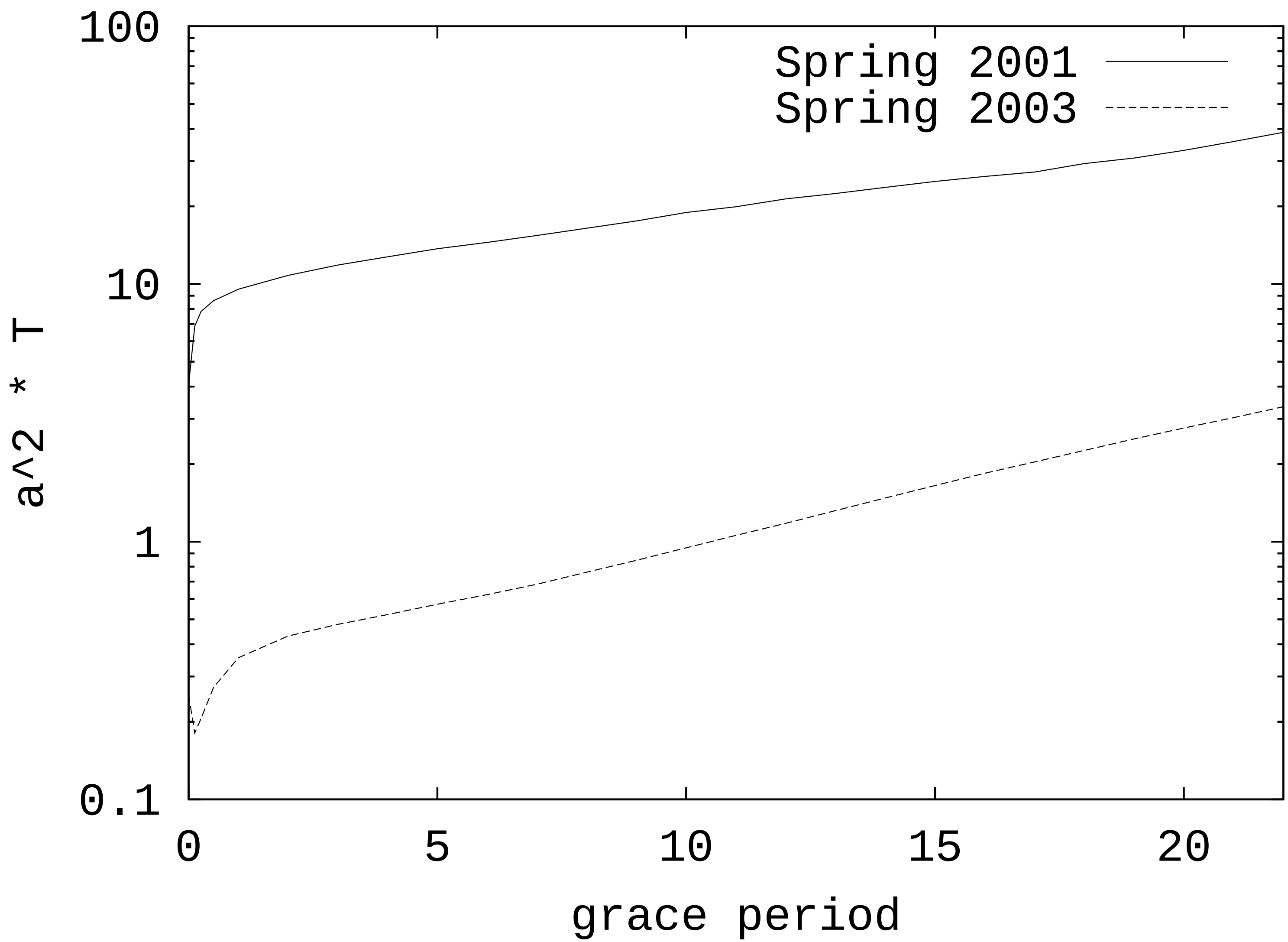




Left Graph Y-Scale 10X smaller

Dark ≈ available, Light ≈ total members

$upfrac^2 \times lifetime$: Then & Now



Why not use small-state lookup?

Isn't designing around bad nodes just good defensive programming?

It's neither free nor necessary

Full info about servers $\longrightarrow$

- Minimum latency access
- Maximum bandwidth access
- *user-specified* QoS selection
- security – everyone tracks/knows everyone
- :

In the next talk, Anjali shows how to disseminate events at rates 600 X the true membership dynamics to 100,000 nodes.