Imperial College London

Building an Internet-Scale Publish/Subscribe System

Peter Pietzuch

Imperial College London

Ian Rose
Rohan Murty
Jonathan Ledlie

Harvard University

Mema Roussopoulos
Matt Welsh

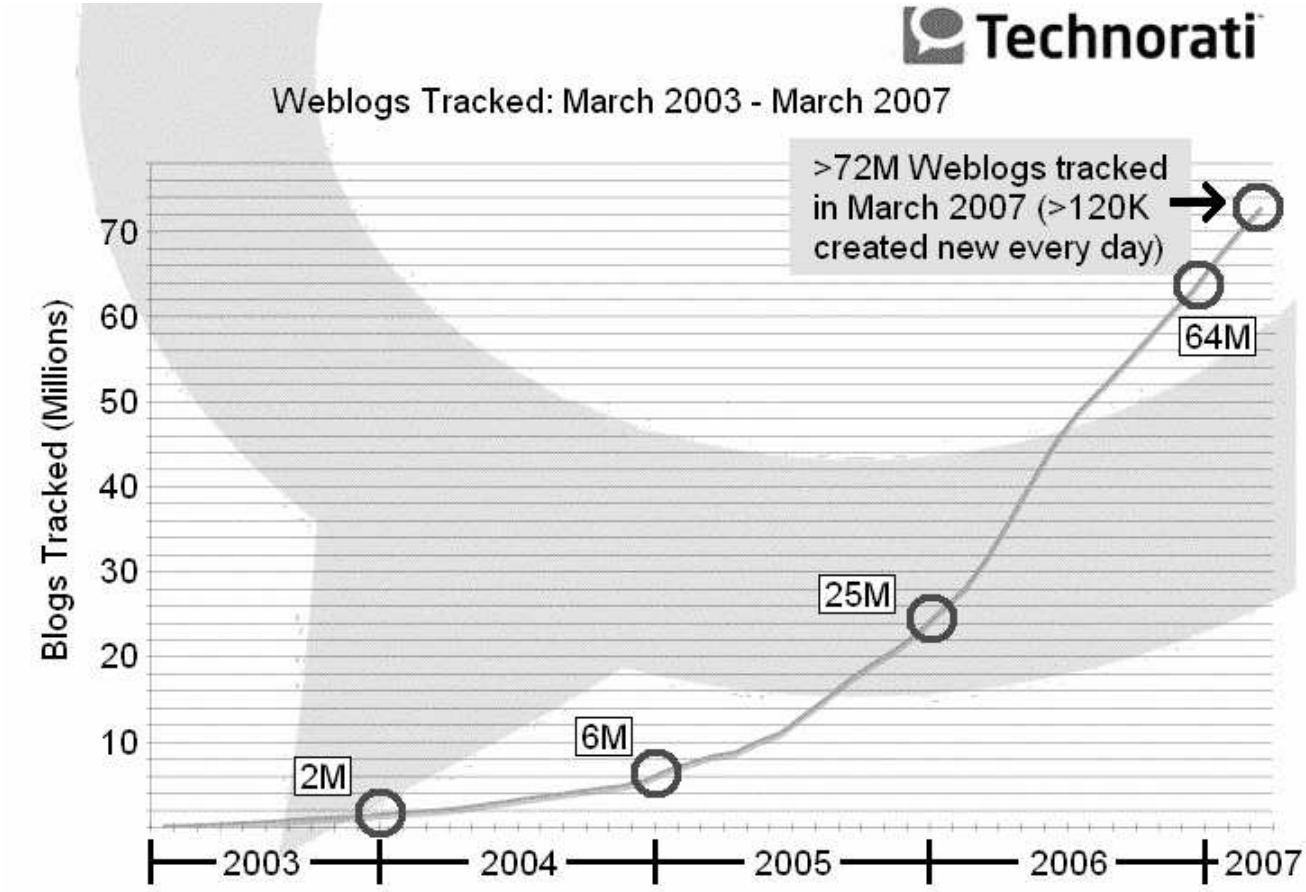
Distributed Software Engineering (DSE) Group

prp@doc.ic.ac.uk

MSN'07 - Cosener's House – July 2007

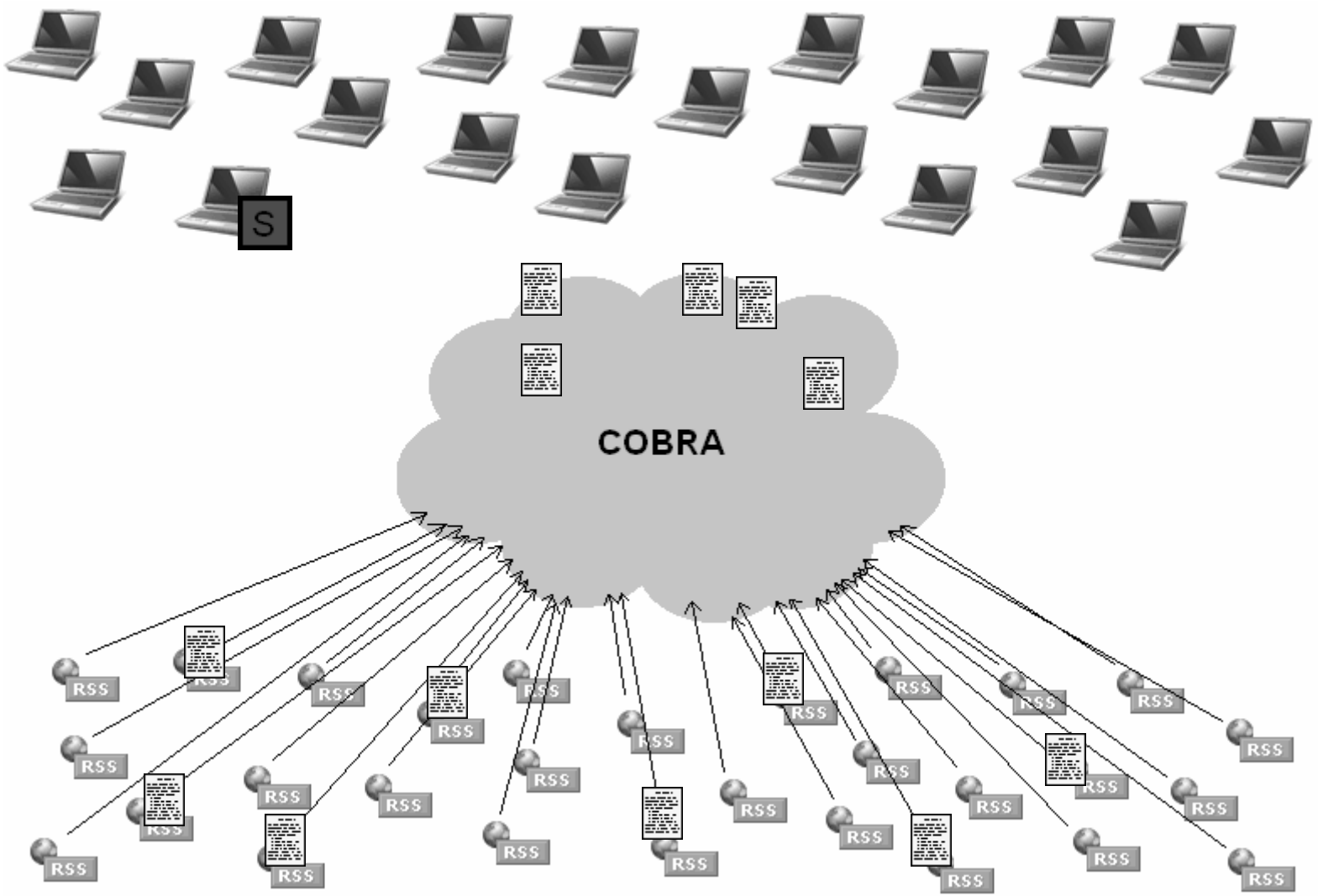
Motivation

Explosive growth of the "blogosphere" and other forms of RSS-based web content



How can we provide an efficient, convenient way for people to access **content of interest** in near-real time?

Content-Based Publish/Subscribe for RSS



Challenges

Scalability

- How can we efficiently support large numbers of RSS feeds and users?

Latency

- How do we ensure rapid update detection?

Provisioning

- Can we automatically provision our resources?

Network Locality

- Can we exploit network locality to improve performance?

Talk Outline

Architecture Overview

- Services: Crawler, Filter, Reflector

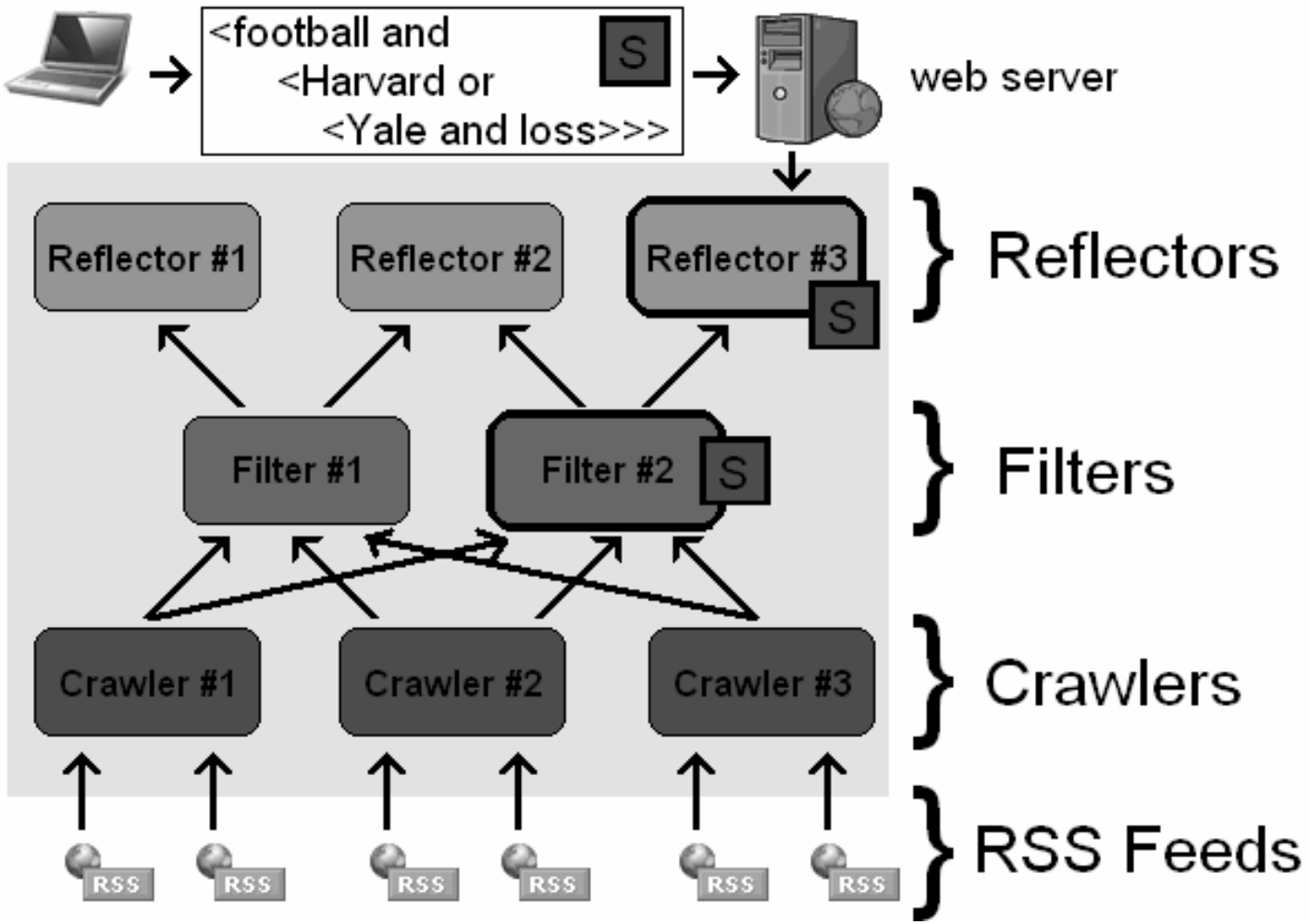
Provisioning Approach

Locality-Aware Feed Assignment

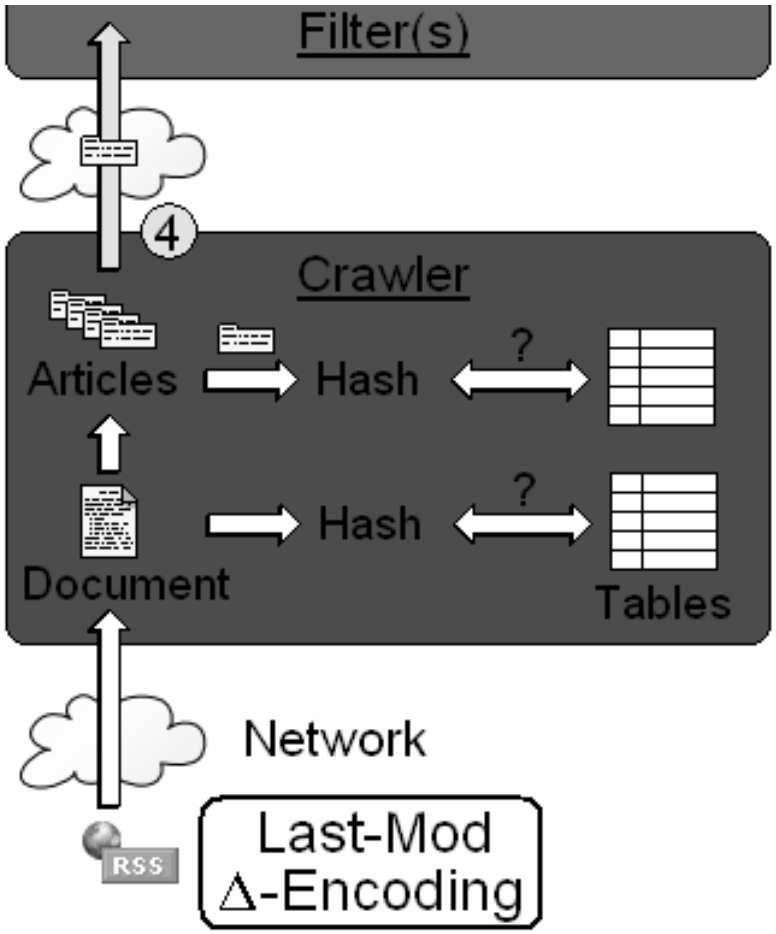
Evaluation

Conclusions

General Architecture

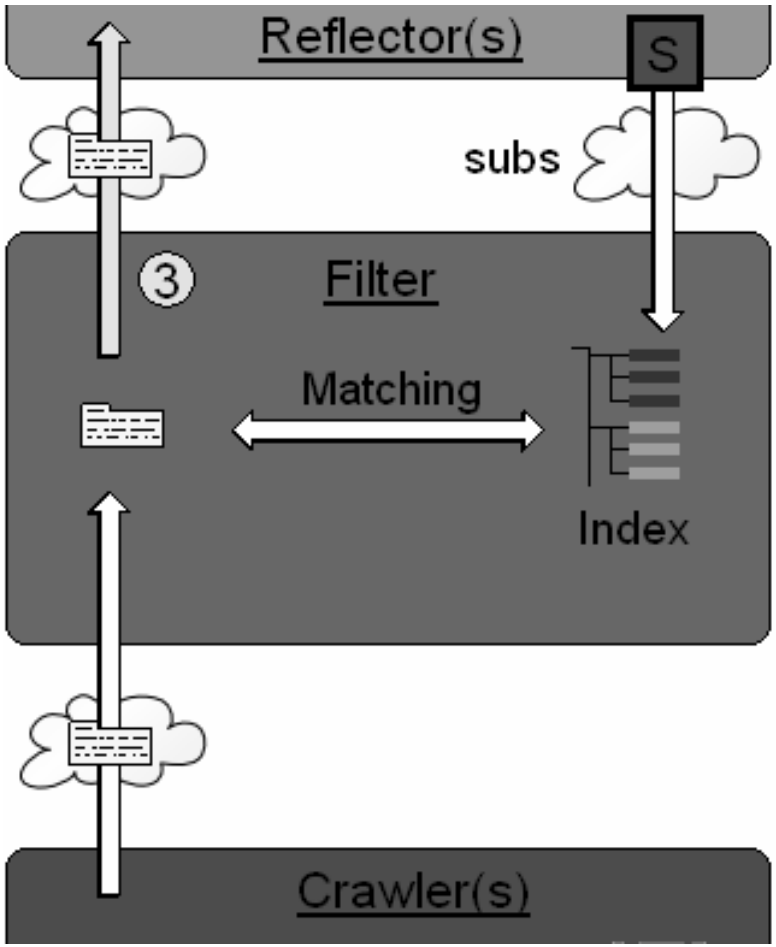


Crawler Service



1. Retrieve RSS feeds via HTTP
2. Hash full document & compare to last value
3. Split document into individual articles; hash each article & compare to last value
4. Send each new article to downstream filters

Filter Service

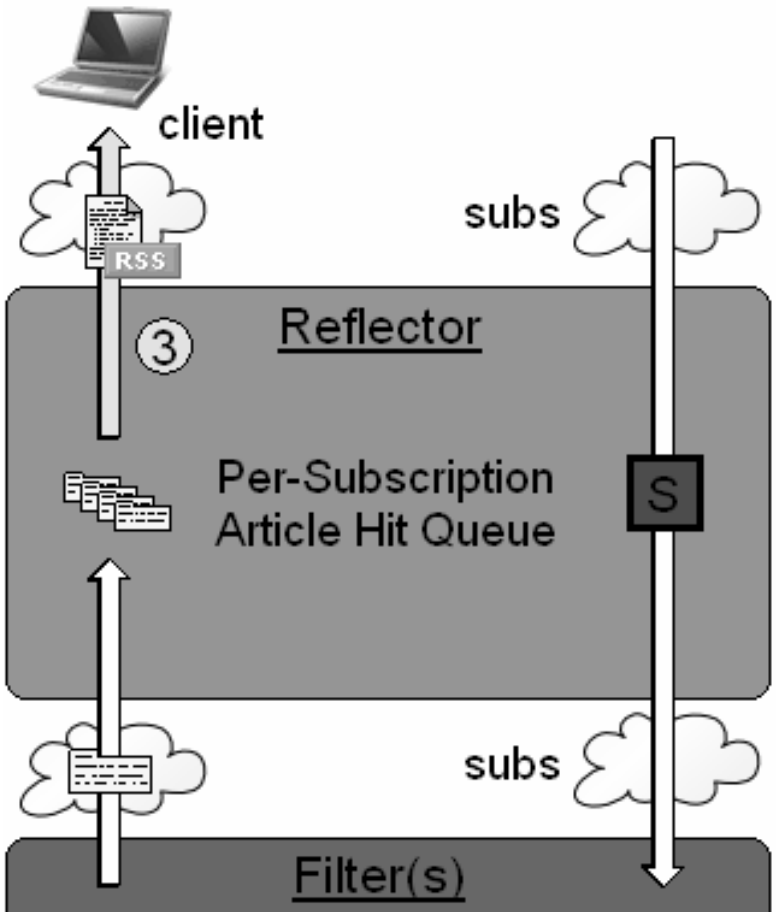


1. Receive subscriptions from reflectors and index for fast subscription matching [Fabret'01]

2. Receive articles from crawlers and match each against all subscriptions

3. Send articles that match >1 subscription to host reflectors

Reflector Service



1. Receive subscriptions from web front-end; create article "hit queue" for each

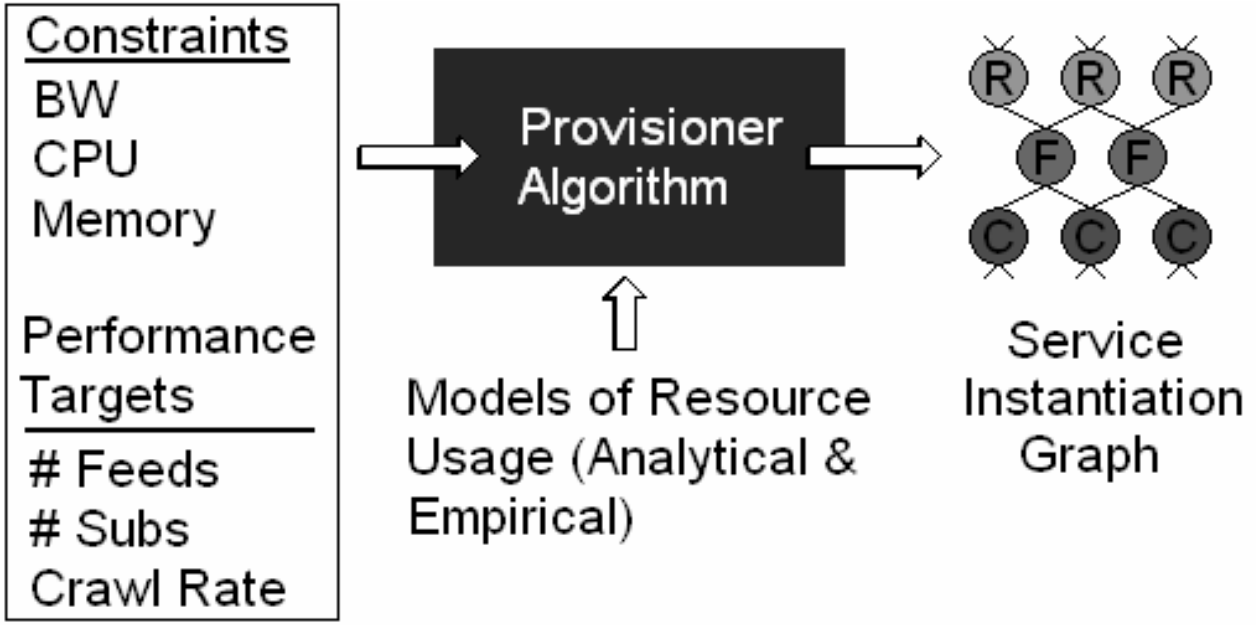
2. Receive articles from filters; add to hit queues of matching subscriptions

3. When polled by client, return articles in hit queue as RSS feed

Provisioning

Cobra services in networked data centers

Iterative, greedy, heuristic to *automatically* determine services required for specific performance targets



Provisioning Algorithm

Algorithm:

1. Begin with minimal topology (3 services)

2. Identify service violation (in-BW, out-BW, CPU, memory)

3. Eliminate violation by "decomposing" service into multiple replicas, distributing load across them

4. Continue until no violations remain

Provisioning: Example

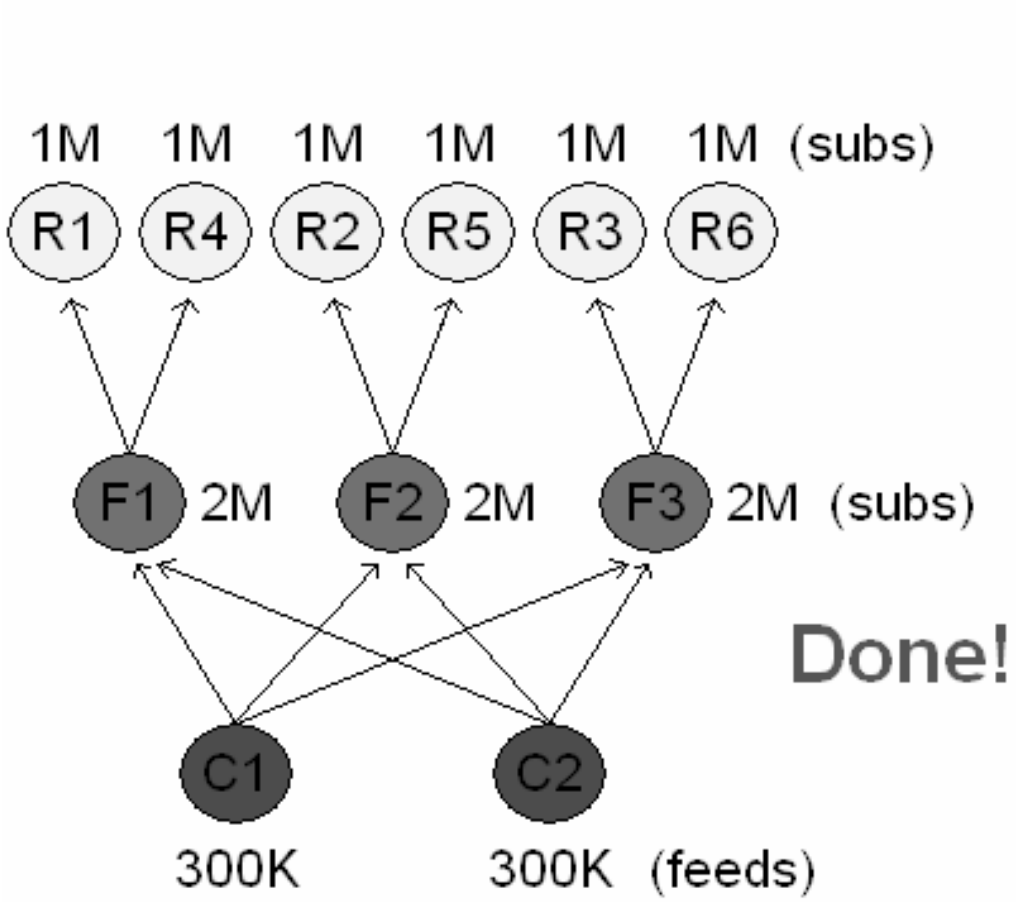
BW: 25 Mbps

Memory: 1 GB

CPU: 4x

Subscriptions:

Feeds: 600K



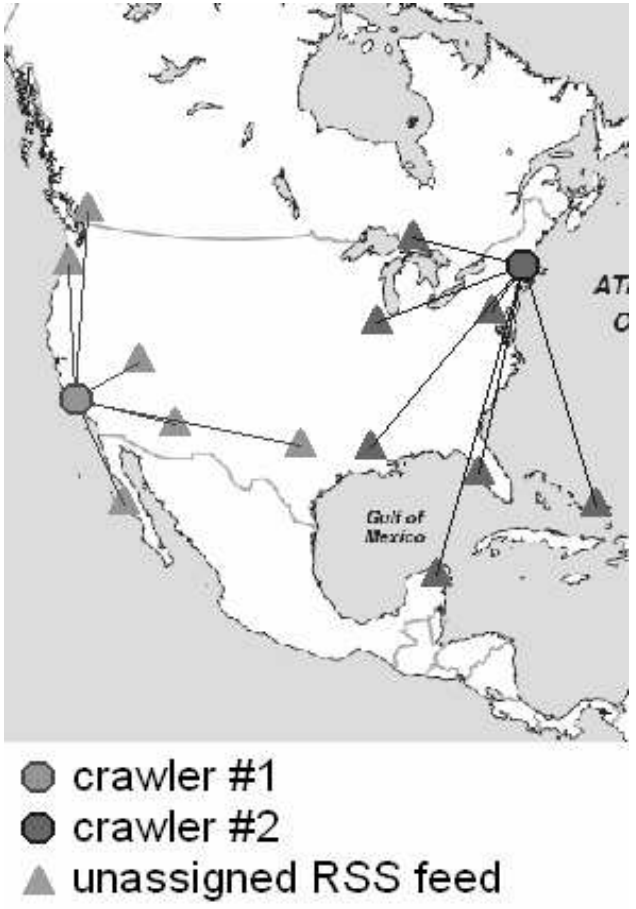
Locality-Aware Feed Assignment

Focus on crawler-feed locality

Offline latency estimates between crawlers and web servers

- Based on DNS indirection [King02[1]]
- Cluster feeds to "nearby" crawlers

18% median reduction in crawl time



[1]Gummadi et al., King: Estimating Latency between Arbitrary Internet End Hosts

Evaluation Methodology

Synthetic evaluation on EmuLab

- Synthetic user queries based on Yahoo! query data
- Trace of 102,446 real feeds from syndic8.com
- Scalability in terms of resource/bandwidth consumption

Live deployment on PlanetLab

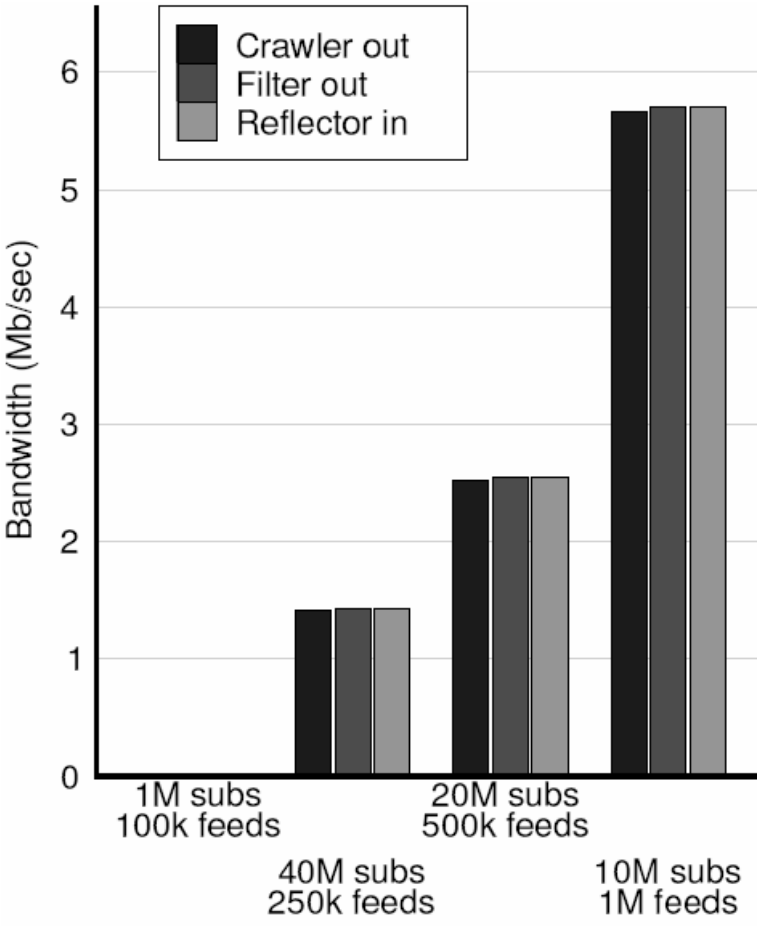
- Benefit of intelligent crawling
- Locality-aware crawler-to-feed assignment
- Intra-network latency

Scalability Evaluation: BW

Four workloads evaluated on Emulab w/ synthetic feeds:

Subs	1M	10M	20M	40M
Feeds	100K	1M	500K	250K
Total Nodes	3	57	51	57
Crawlers	1	1	1	1
Filters	1	28	25	28
Reflectors	1	28	25	28

Bandwidth usage scales well with feeds and users



Conclusions

Provisioning important but often overlooked

- Provisioning by hand is hard
- Simple provisioning algorithm with room for improvement

Reproducible evaluation on PlanetLab hard

- Emulab better for controlled experiments
- Hard to find good workloads for synthetic benchmarks

Locality on the Internet matters

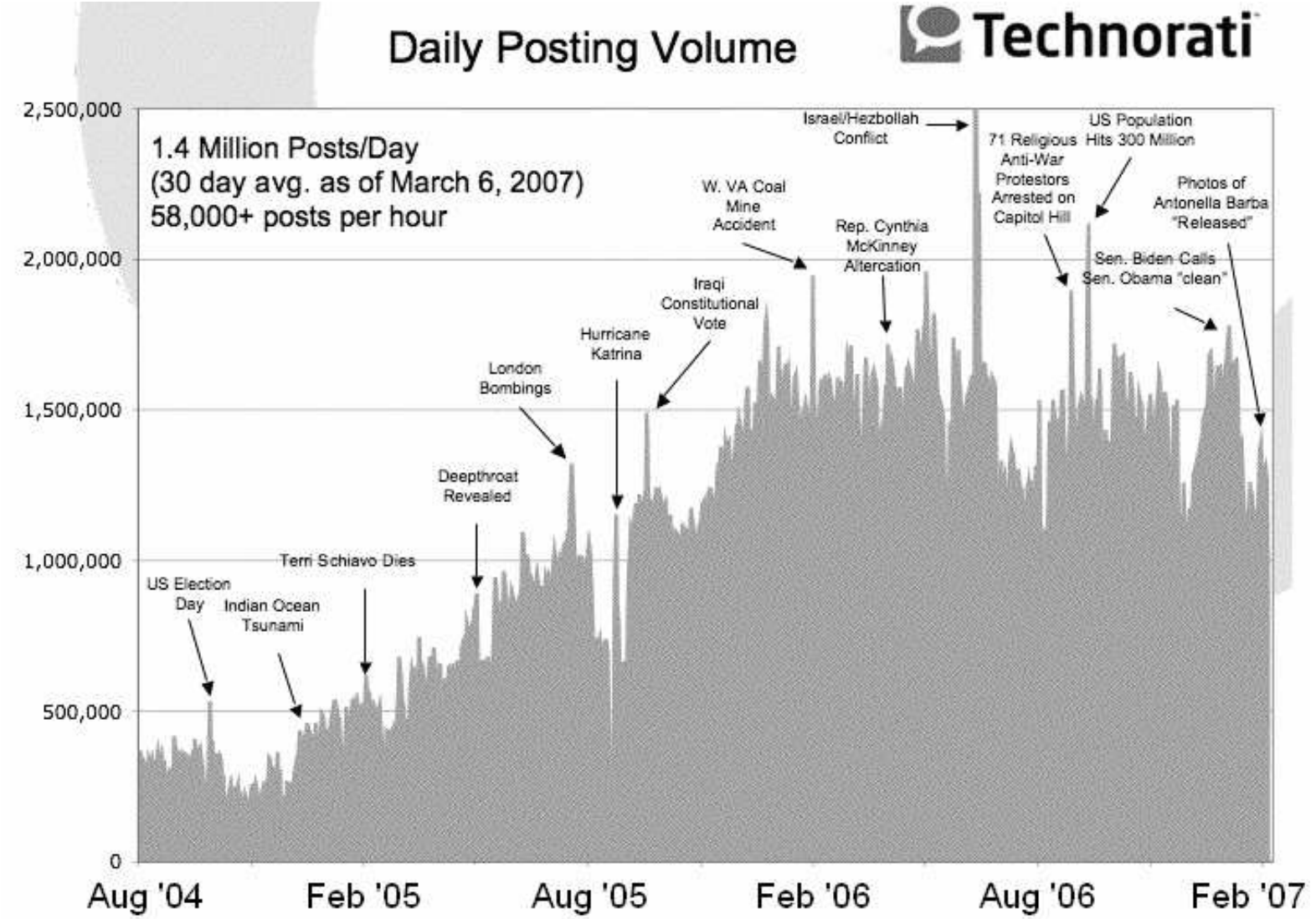
- But network measurements can be expensive

Thank you

Any Questions?

Peter Pietzuch
Department of Computing
Imperial College London
http://www.doc.ic.ac.uk/~prp
prp@doc.ic.ac.uk

Backup


Daily Posting Volume
Technorati
1.4 Million Posts/Day
(30 day avg. as of March 6, 2007)
58,000+ posts per hour
2,500,000
2,000,000
1,500,000
1,000,000
500,000
0
Aug '04
Feb '05
Aug '05
Feb '06
Aug '06
Feb '07
US Election Day
Indian Ocean Tsunami
Terri Schiavo Dies
Deepthroat Revealed
London Bombings
Hurricane Katrina
Iraqi Constitutional Vote
W. VA Coal Mine Accident
Israel/Hezbollah Conflict
Rep. Cynthia McKinney Altercation
71 Religious Anti-War Protestors Arrested on Capitol Hill
US Population Hits 300 Million
Sen. Biden Calls Sen. Obama "clean"
Photos of Antonella Barba "Released"

Source:

Current Approaches

RSS Readers (Thunderbird)

- topic-based (URL), inefficient polling model

Topic Aggregators (Technorati)

- topic-based (pre-defined categories)

Blog Search Sites (Google Blog Search)

- closed architectures, unknown scalability and efficiency of resource usage

Related Work

Traditional distributed pub/sub systems, e.g. *Siena* (Univ. of Colorado):

- Address decentralized event matching and distribution.
- Typically do not (directly) address overlay provisioning.
- Often do not interoperate well with existing web infrastructure.

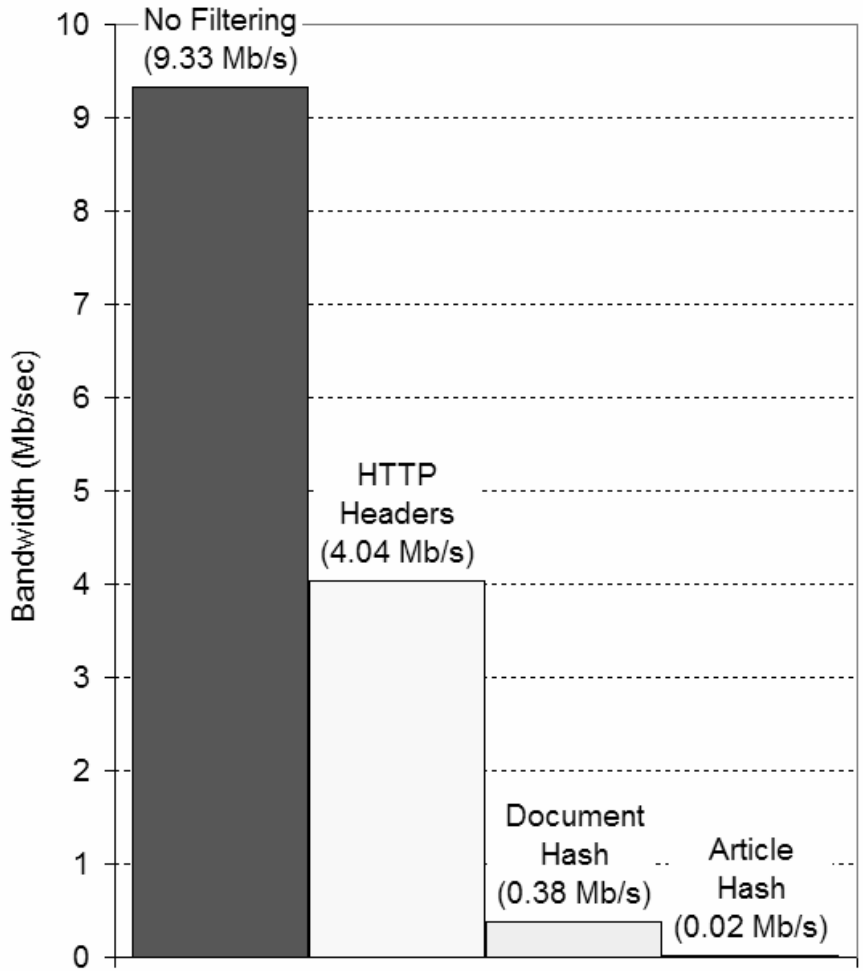
Corona (Cornell) is an RSS-specific pub/sub system

- topic-based (subscribe to URLs)
- Attempts to minimize both polling load on content servers (feeds) and update detection delay.
- Does not specifically address scalability, in terms of feeds or subscriptions.

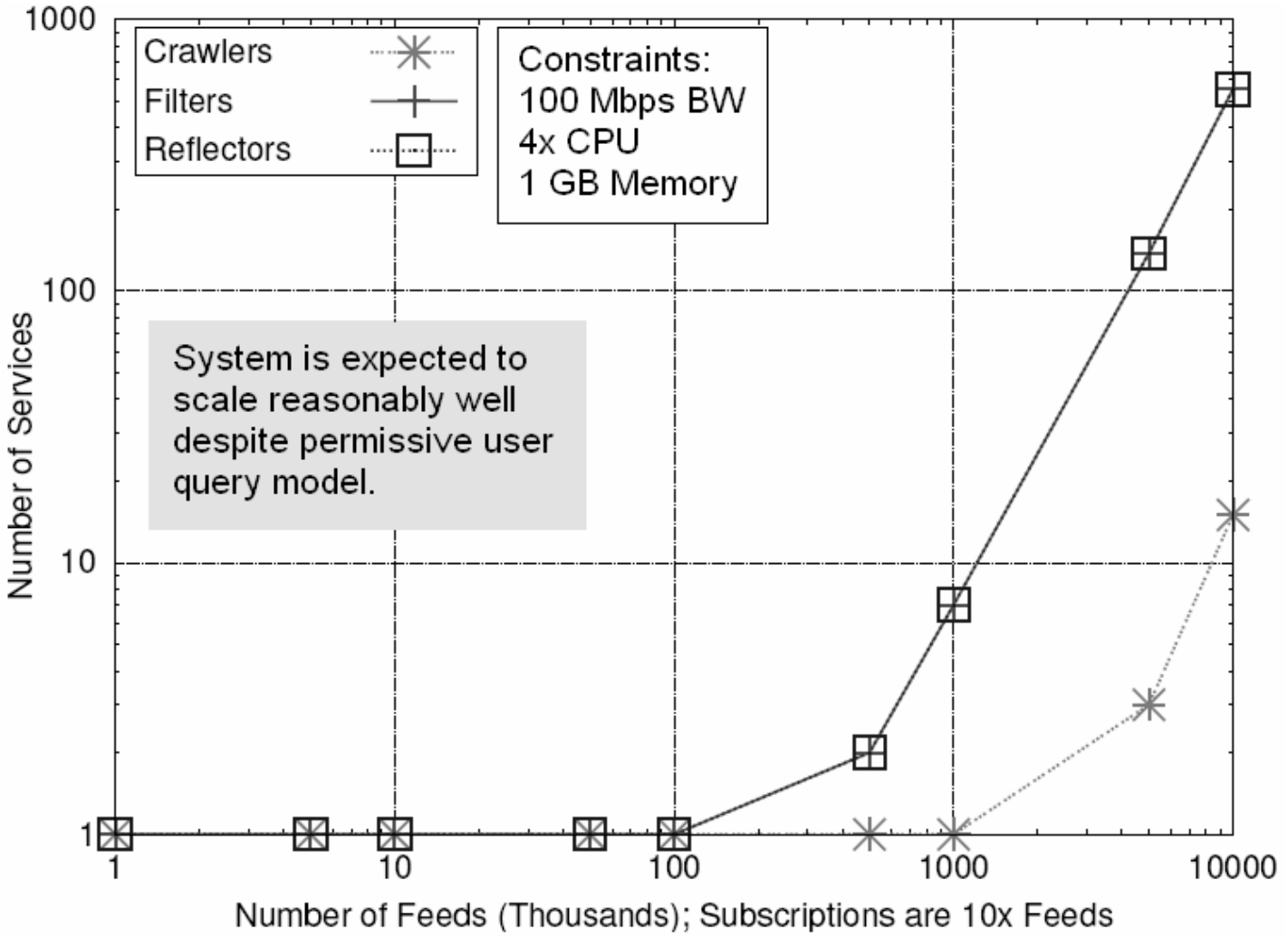
Benefit of Intelligent Crawling

One crawl of all 102,446 feeds over 15 minutes, using 4 crawlers. Bandwidth usage recorded for varying filtering levels

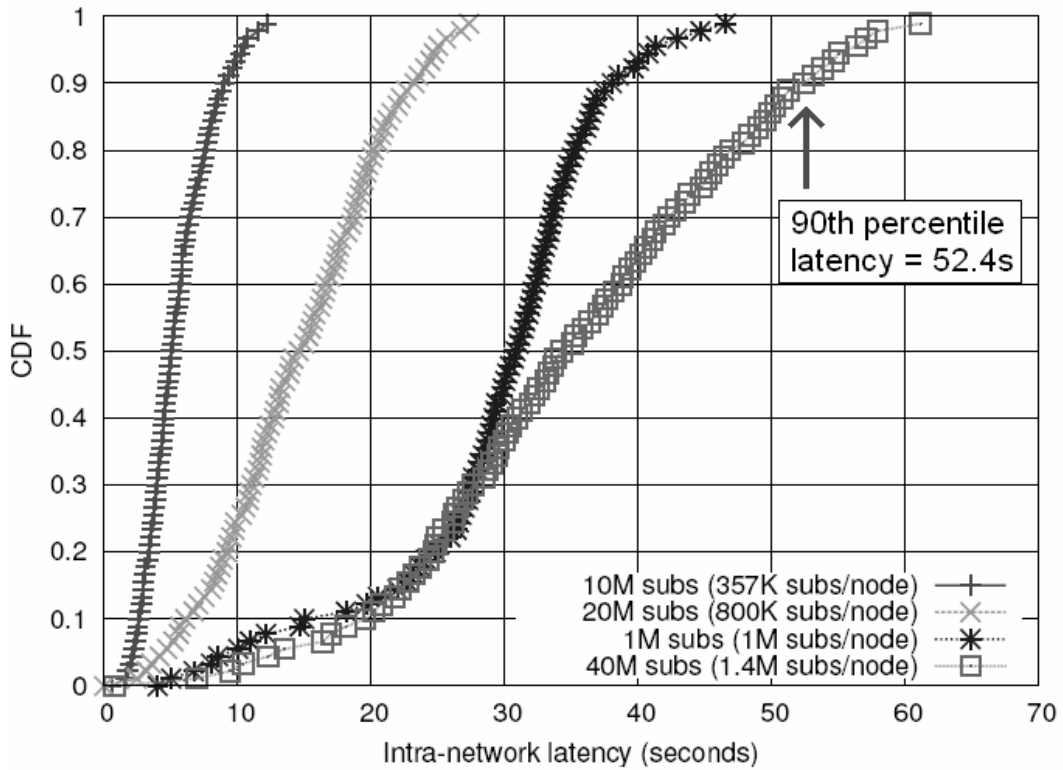
Crawlers able to reduce bandwidth usage by **99.8%** through intelligent crawling



Provisioner-Predicted Scaling



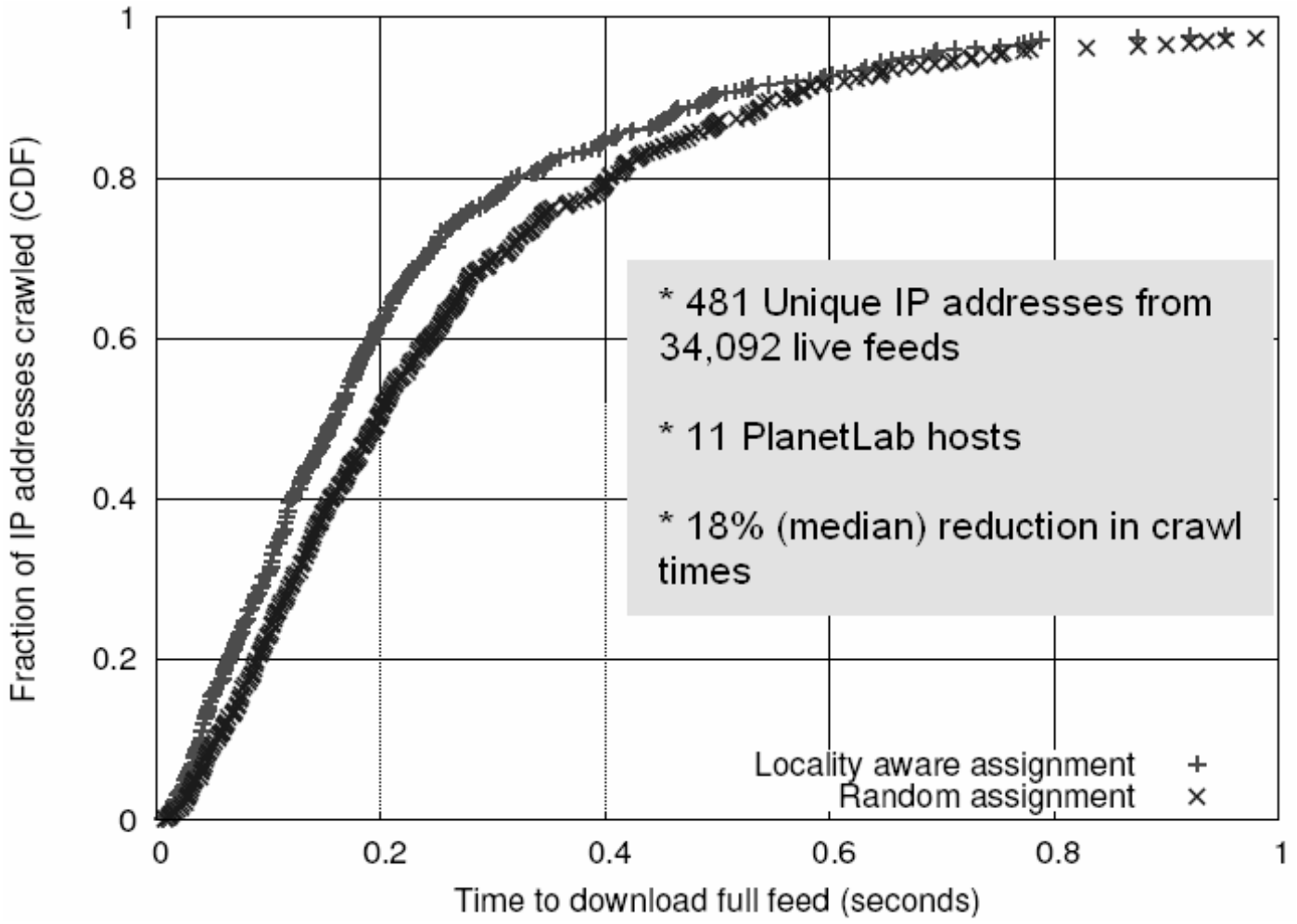
Intra-Network Latency



Total user latency = crawl latency + polling latency + intra-network latency

Intra-network latencies largely dominated by crawling and polling latencies

Locality-Aware Feed Assignment



Future Work

Many open directions:

- Evaluating real user subscriptions & behavior
- More sophisticated filtering techniques (e.g. rank by relevance, proximity of query words in article)
- Subscription clustering on reflectors
- How to discover new feeds & blogs
- Adapting Cobra to a peer-to-peer setting may also be possible