

Principles of Distributed Redirection



Michal Szymaniak

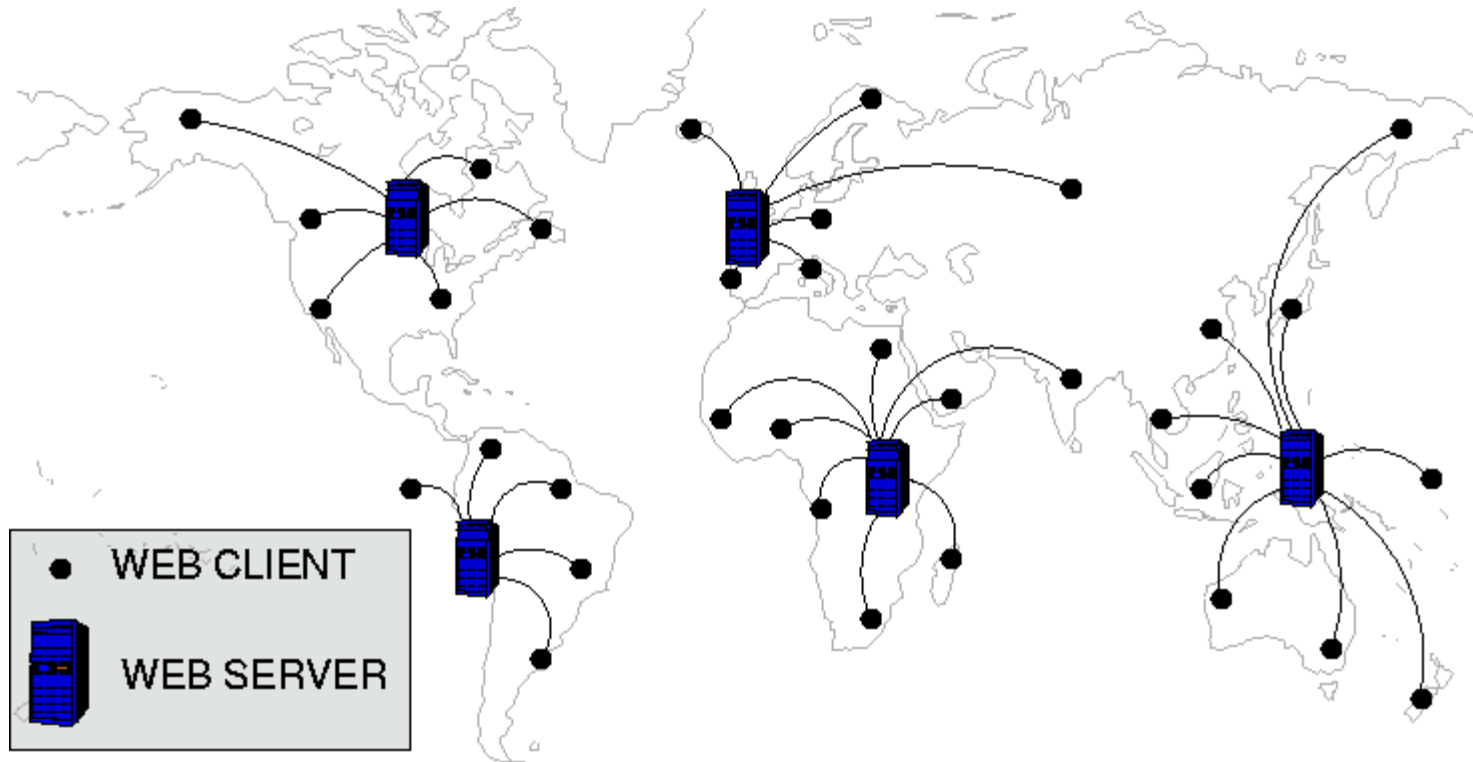
describes the work by

Aline Baggio and Maarten van Steen

Vrije Universiteit Amsterdam

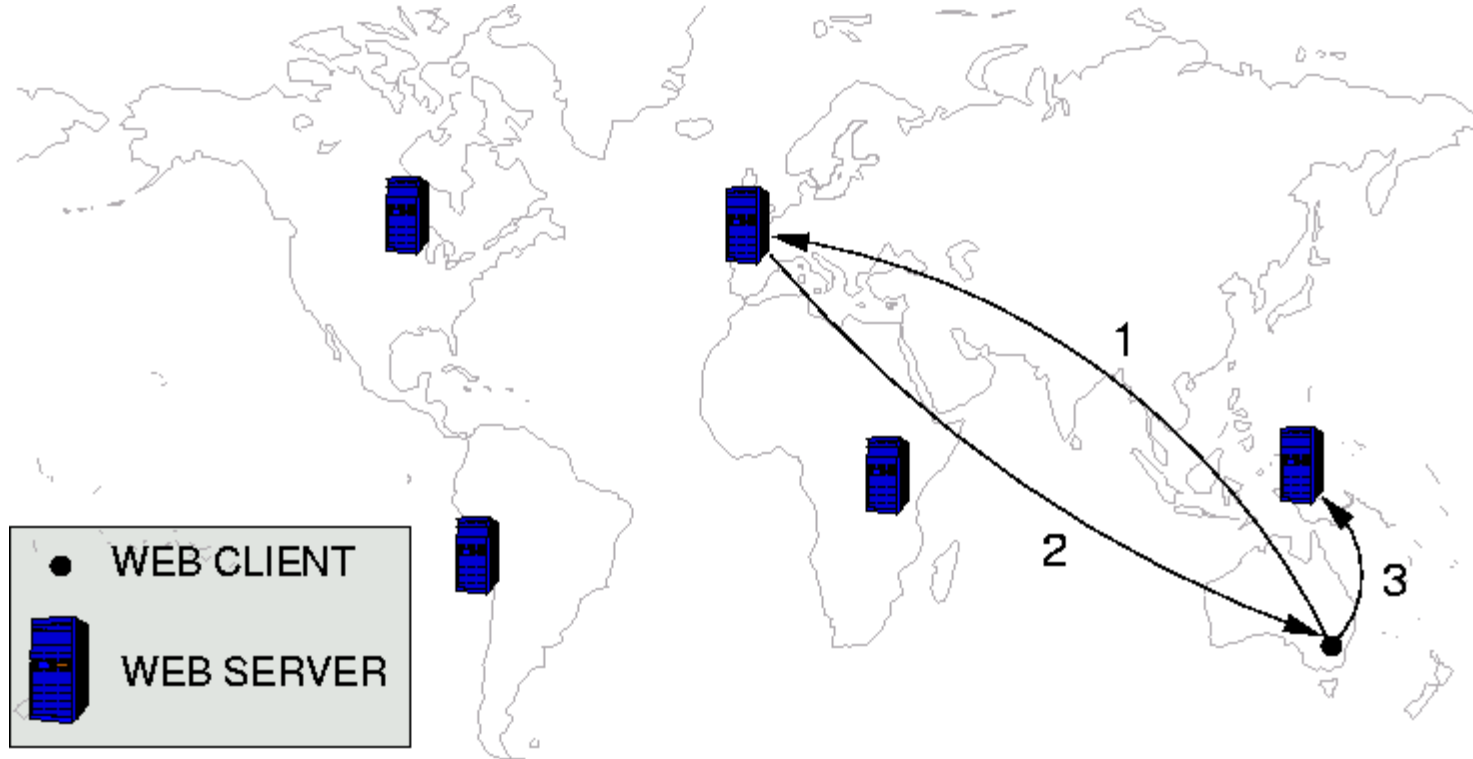
{michal,baggio,steen}@cs.vu.nl

Problem



- Popular Web services are **replicated**
 1. Install many service copies (replicas) hosting **the same data**
 2. Let each replica service **its nearby clients**
- Data close to the clients → **faster access**
- **But how do clients find their nearby replicas?**

Solution: Classical Redirection

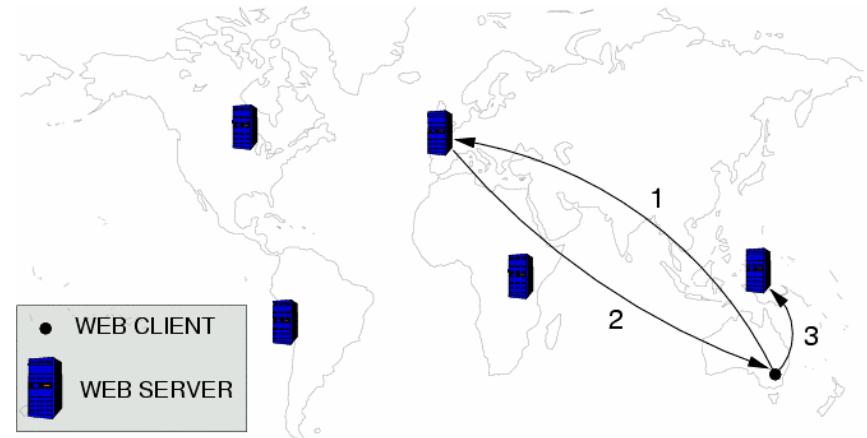


1. Clients first contact the **main** server of the service..
2. ..and obtain an **address** of a nearby replica..
3. ..via which they access the service.

What is wrong with the picture above?

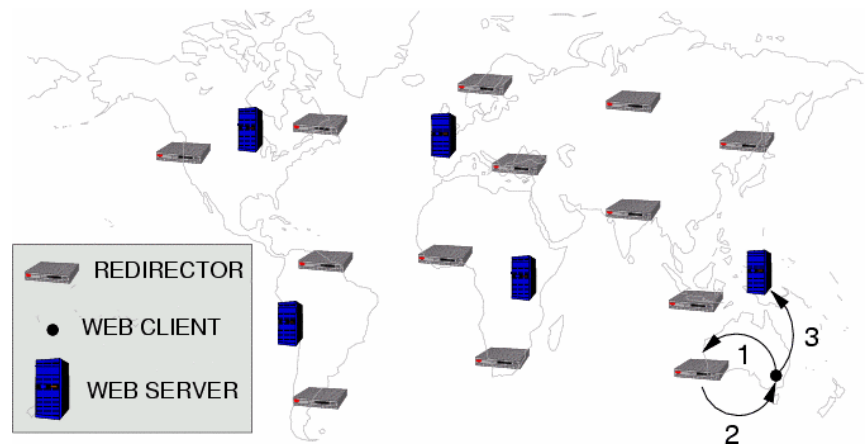
Classical vs. Distributed Redirection

- Clients must contact **main server**
- Main server can be **faaar** away
- **Faaar** away means **looong** waiting for replica addresses



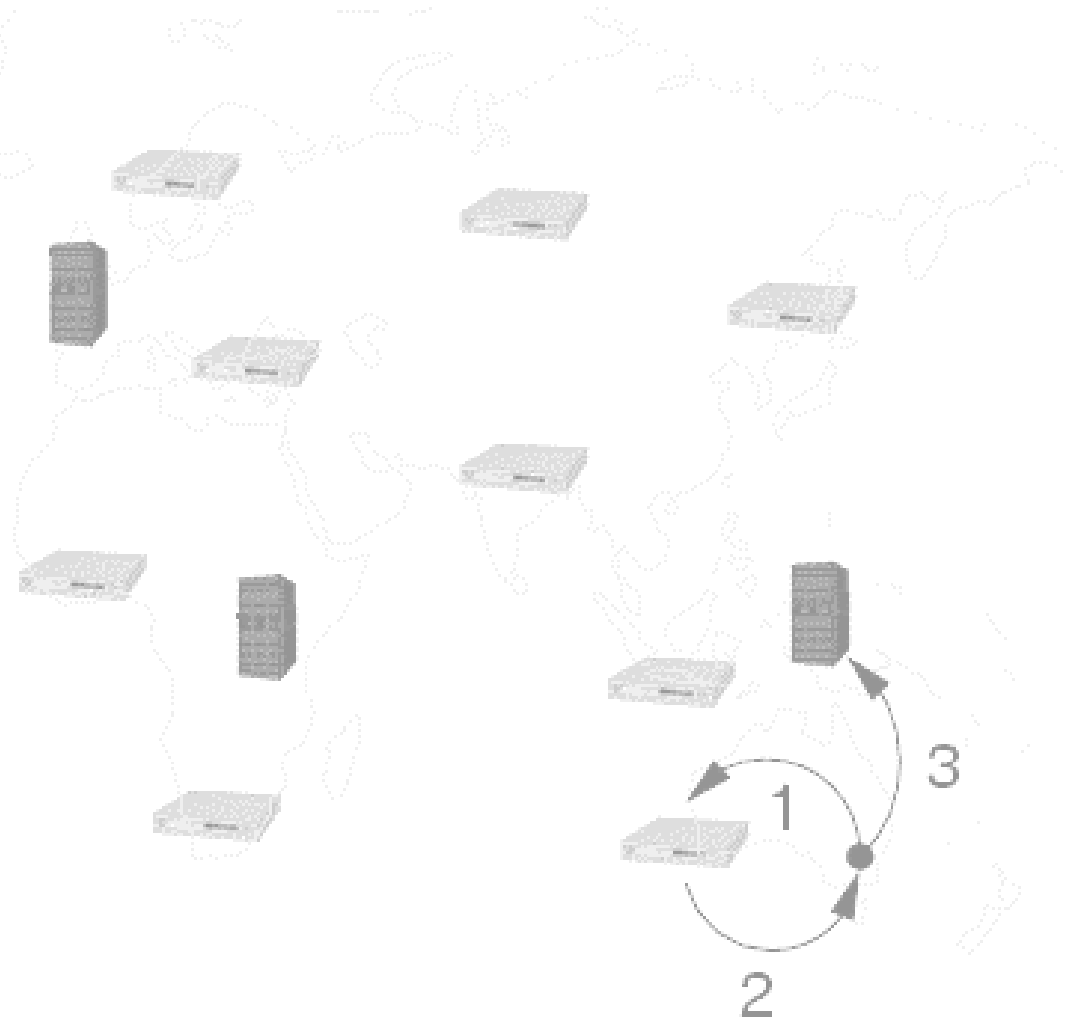
So what can we do about it?

- Take **many** redirecting servers..
- Install them **close** to clients..
- Tell clients to contact **near-by** redirection servers..
- ..and voila, clients can **quickly** obtain replica addresses



Talk Outline

- Redirector Organization
- Redirector Operation
- Redirector Mining
- Performance
- Summary + Future Work



Redirector Organization: Motivation

- Client's perspective: redirectors translate URLs into IP addresses
 - easy in centralized approach
 - could be easy in distributed one as well..
 - ..but we want our solution to work in Globule

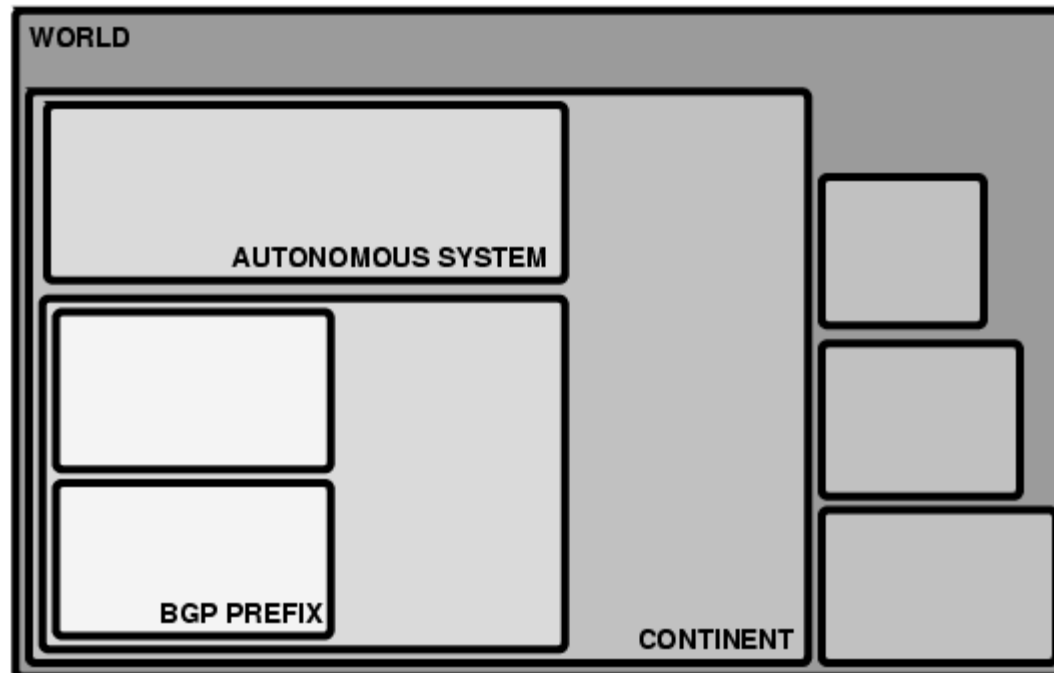


- Globule: Web services get together to share resources
 - in particular: services share a global pool of redirectors
 - **replicas come and go**
 - we do not want **global updates**
 - anyway redirectors need to know only nearby replicas
 - so: **organize redirectors** -- each one can **discover** all replicas
 - **local updates**
 - **fast discovery of nearby replicas**

Redirector Organization: Regions

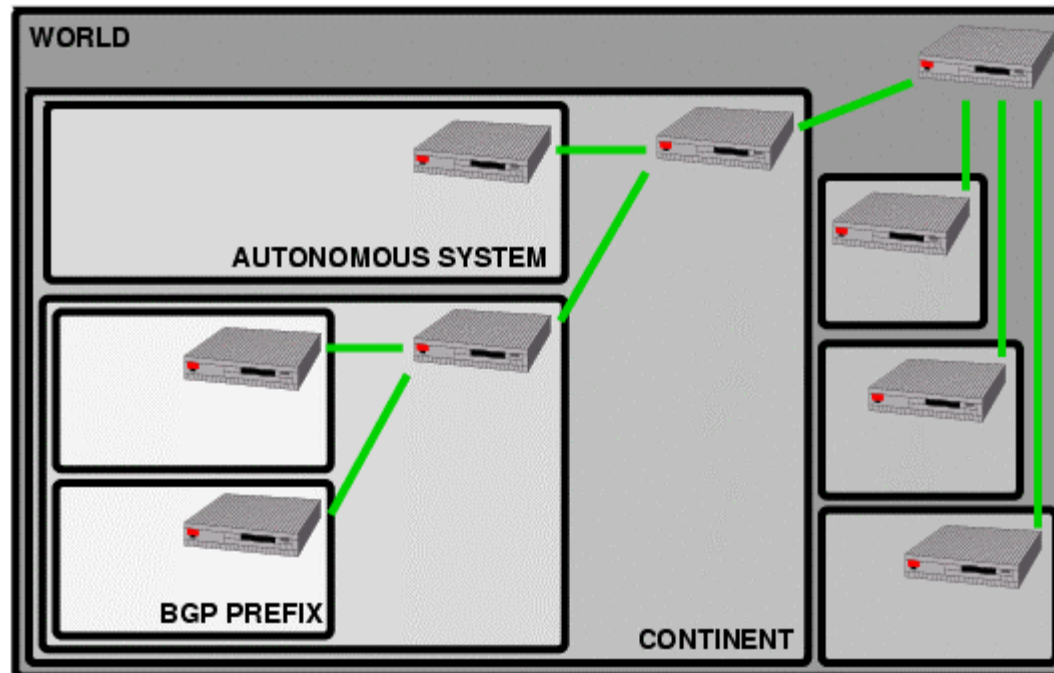
We follow approach used in **global location services**:

- Step 1: divide the Internet into regions
 - Relatively flat hierarchy
 - **quick traversing**



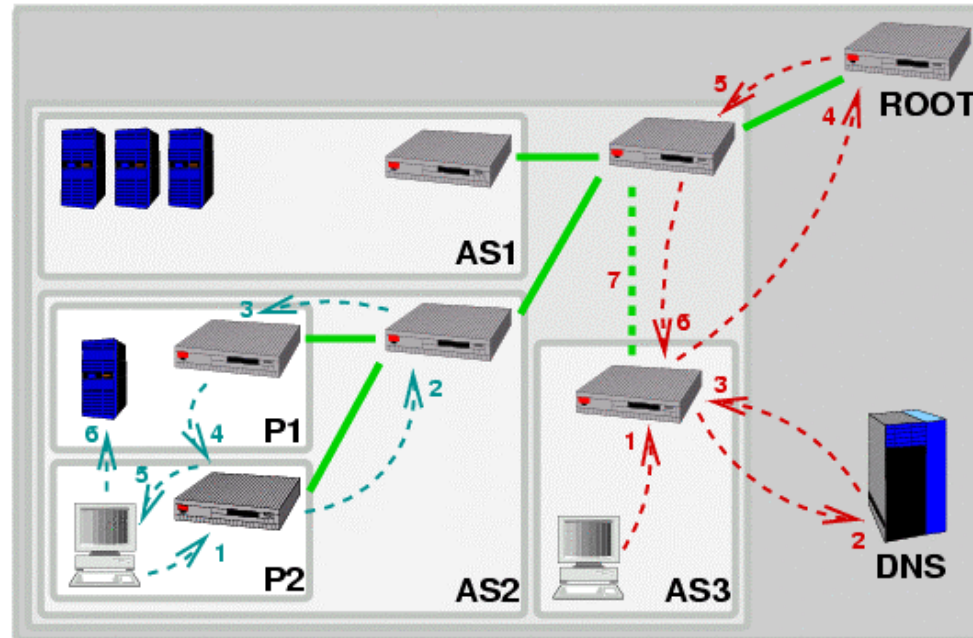
Redirector Organization: X-trees

- Step 2: build hierarchy of redirectors
 - Redirectors mapped to regions
 - **service is in charge**
 - Hierarchy unique per service X (X-tree)
 - **easy update of replica addresses**
 - Replica addresses in leaves
 - internal nodes know URLs only



Redirector Operation

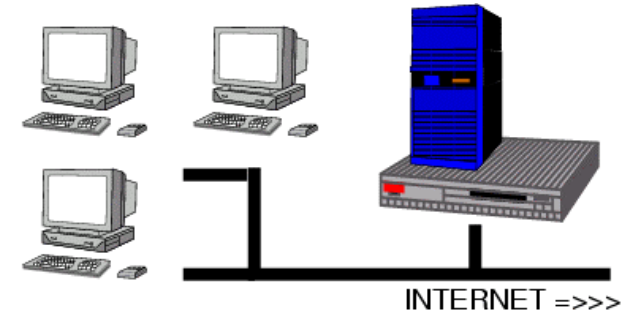
- Suppose: **client knows its redirector** and requests it for URL X
- Easy case: **redirector belongs to X-tree**



- Harder case: **redirector does not belong to X-tree**
 - redirector **locates** X-tree root (current work)
 - redirector **joins** X-tree (for future requests) and **runs easy case**
- Hardest case: **what if clients do not know their redirectors?**

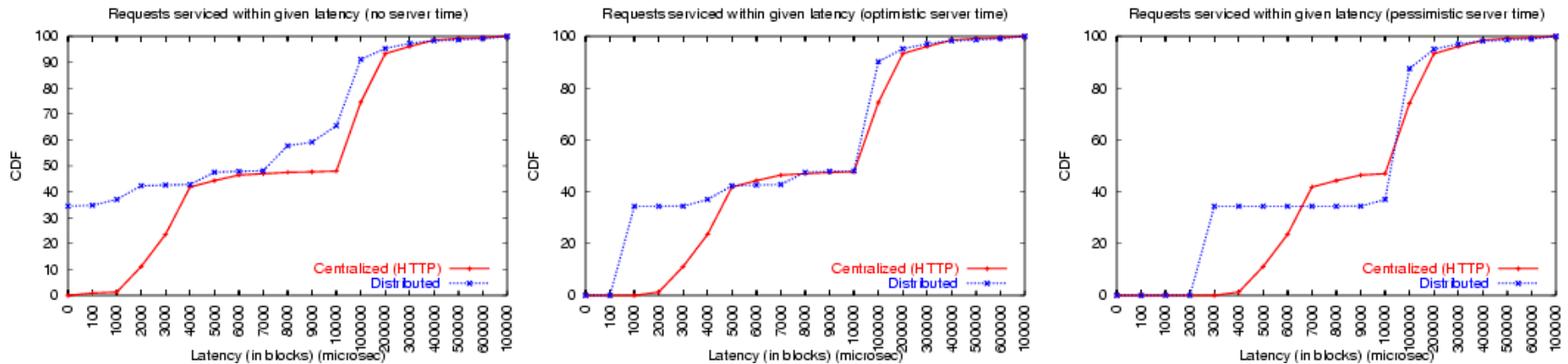
Redirector Mining

- We **cannot configure** clients.. :-(
- ..so let **other people do it** :-)
- Redirectors in/close to user networks
 - (transparent) Web proxy..
 - ..plus (possibly) modified DNS server
 - usually **integrated** with Globule server
 - **anywhere**, but the closer the better
 - either way: clients configure redirectors **themselves**
- Additionally: redirectors inside service-provided Globule servers
- Does it make sense?
 - similar to regular DNS / Web caches
 - after all: users want to improve **their** experience



Performance

- Simulation based on traces from VU Website..
 - 57 weeks (over 1 year)
 - 1.6M unique client addresses
 - 1M unique documents
 - 3.7M replicas
 - 58M requests
- ..and latencies measured with King



- In short:
 - 34% of requests are serviced locally; 43% within 4 millisecc
 - making more than 3 hops eliminates the benefit from dist. redirection

Summary + Future Work

- **Distributed Redirection**
 - makes the redirection process local
 - **no long-haul queries**
 - **reduced redirection latency**
 - exploits **globally-scalable** infrastructure of redirectors
 - redirectors inside Globule servers
 - Globule servers provided by clients and Web services
- **Future work:**
 - **adaptable region hierarchy** (now **globally fixed**)
 - **dynamic X-trees** (now **assumed to be quite stable**)
 - **automatic client-side redirector discovery** (now **manual**)