

Distributed Content Delivery with Load-Aware Network Coordinates

**Department of Computing
Imperial College London**

Nicholas Ball

nsb04@doc.ic.ac.uk - nicholas.ball@nodelay.com

Peter Pietzuch

prp@doc.ic.ac.uk

Content Distribution Networks

- **Content Distribution Networks (CDNs)** provide files to millions of Internet users
- Deal with the problem of **mapping clients to servers**
- A **scalable** and **reliable CDN** has
 - Locality Awareness
 - Load Awareness
- We achieved both in our system under **one unified framework** by extending a new mechanism called **Network Coordinates**

Talk Overview

- Locality and Load Awareness
- Network Coordinates
- Load-Aware Network Coordinates
- System Architecture
- Evaluation
- Conclusion
- Further Work

Locality Awareness

- **Exploit proximity relationships within the network**
 - Geographical Distance
 - Round-trip Latency
 - Topological Distance
- **Minimizing distances yields greater QoS**
 - Lower Latencies (closer to data)
 - Improved throughput (less likely to encounter hotspots)
 - Better reliability (fewer links and routers traversed)
 - Decreasing network saturation (data is kept local)

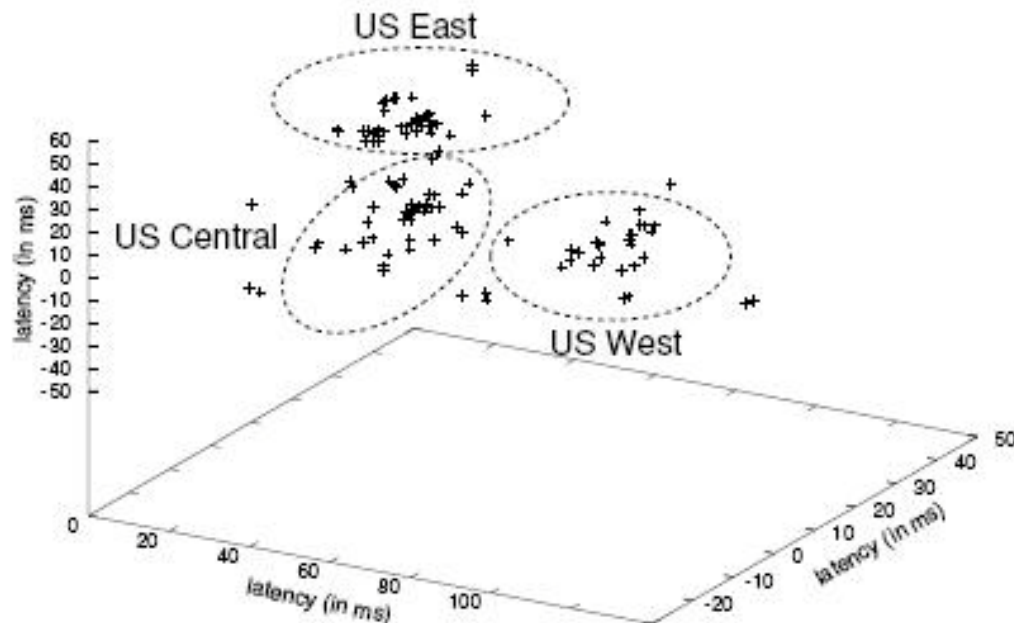
Load Awareness

- Strategy to reduce overloading of content servers
- **Computational Load** - Server may be on a shared machine or have too many concurrent requests
- **Network Congestion** - Routers overloaded, links saturated

- Older CDNs such as DHT based ones
 - **Load-balancing** - distribute content/requests uniformly across content servers regardless of locality and load
- Newer CDNs (CoralCDN)
 - **Load-aware** - pick nearby content servers whilst avoiding overloaded ones

Network Coordinates (NCs)

- Provide each node (host) with a synthetic coordinate within an n -dimensional virtual space
- Each node performs round-trip time (RTT) **latency measurements** to a small **subset of other nodes**
- **Latencies** done at a **network-level (PINGs)**
- Distance between **all** nodes are estimated latencies

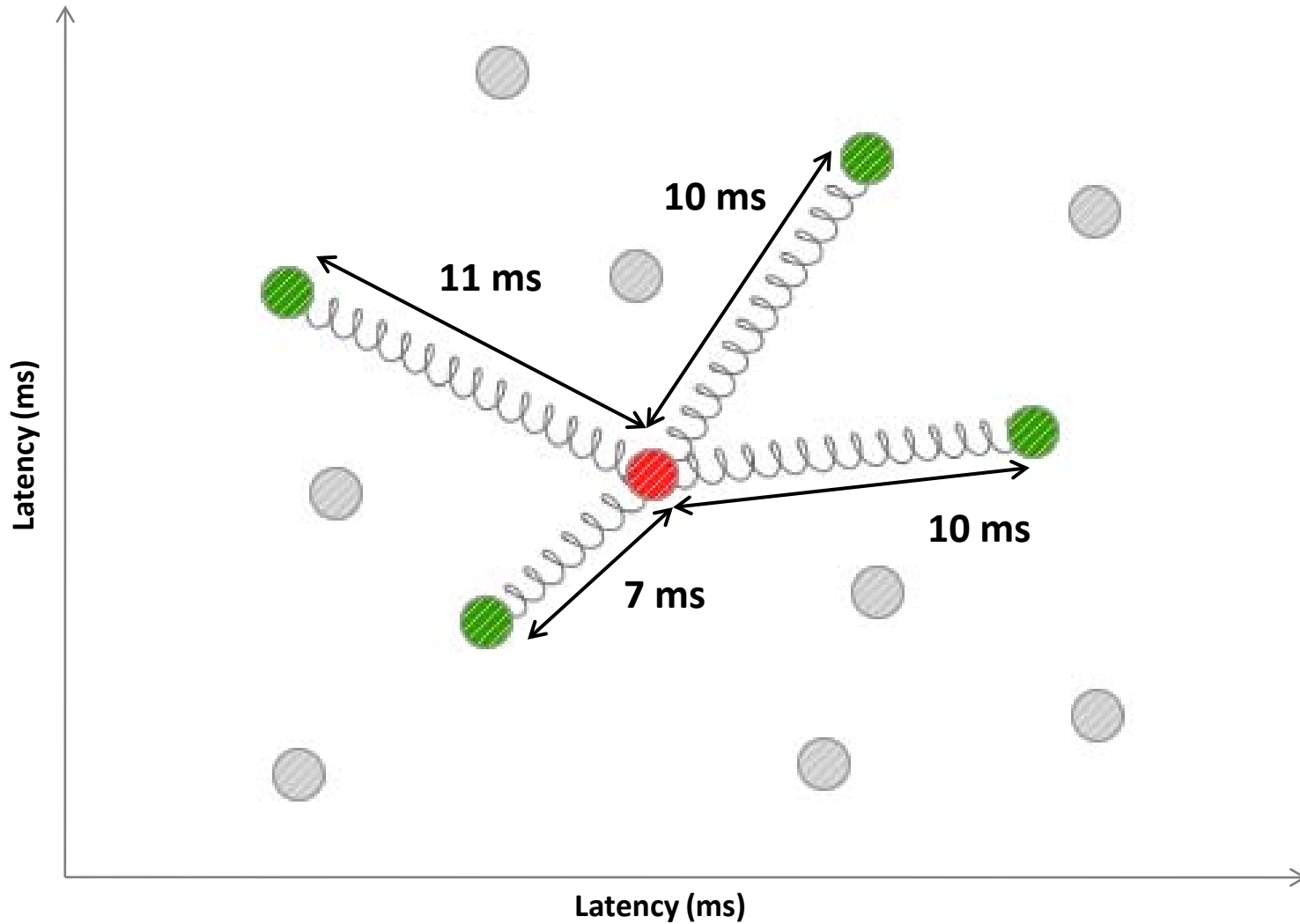


Vivaldi

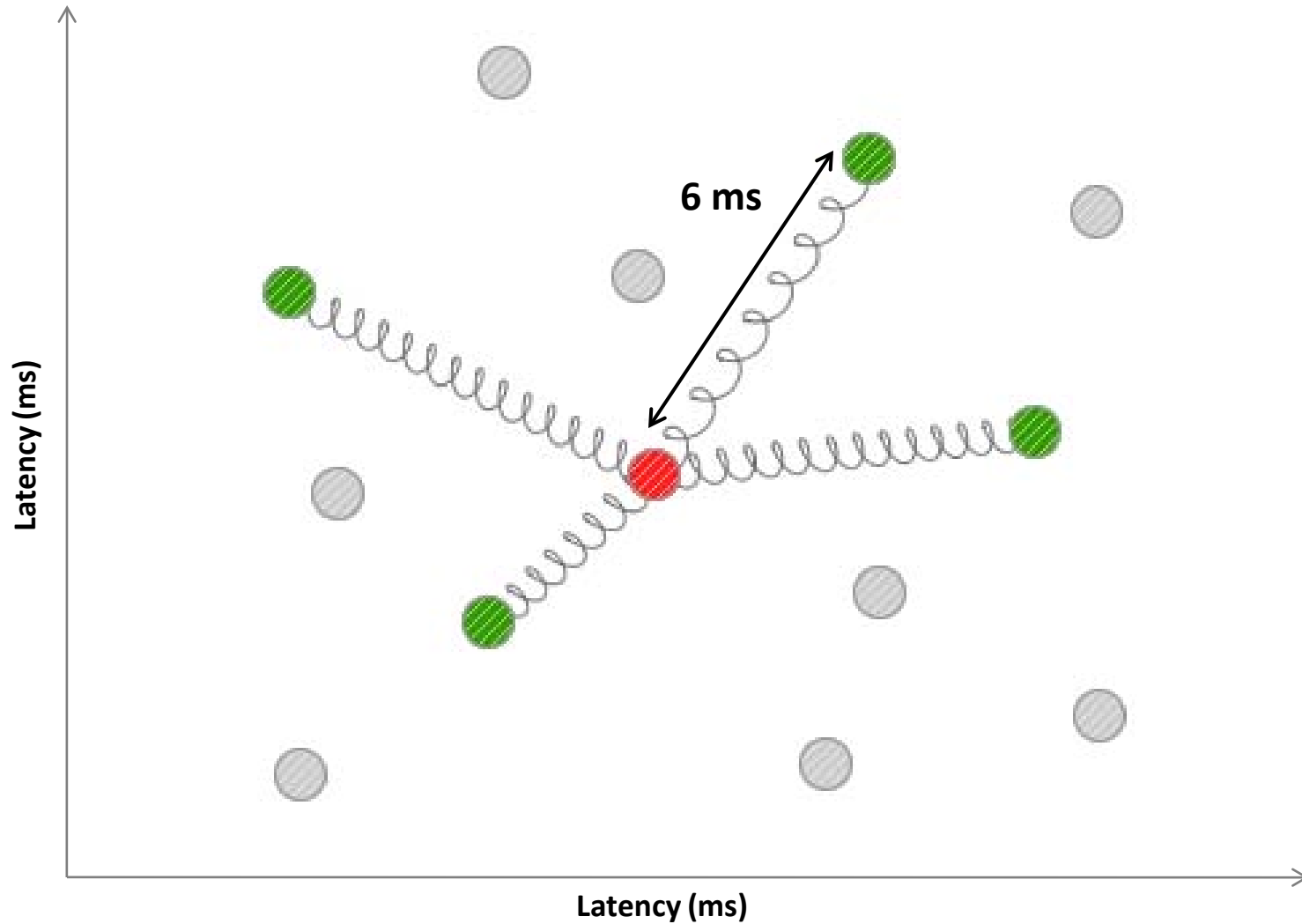
- **Distributed algorithm** to place nodes in a coordinate space
- **Simulation-based**
- **Spring-relaxation** - springs placed between nodes to represent how they **attract** and **repel** to one another

- Used by **Pyxida** - Open-source network coordinate library

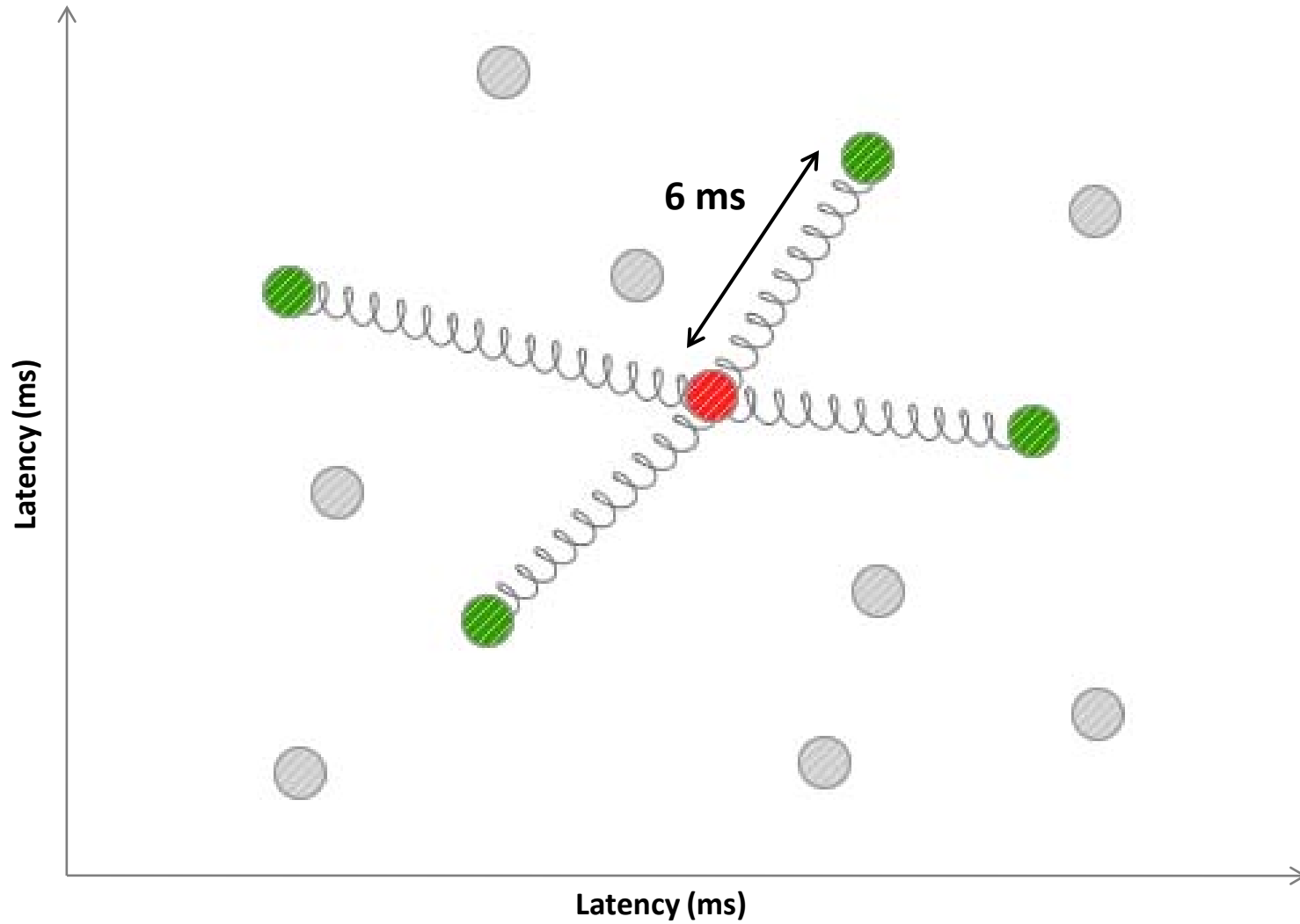
Spring Relaxed



Spring Tense

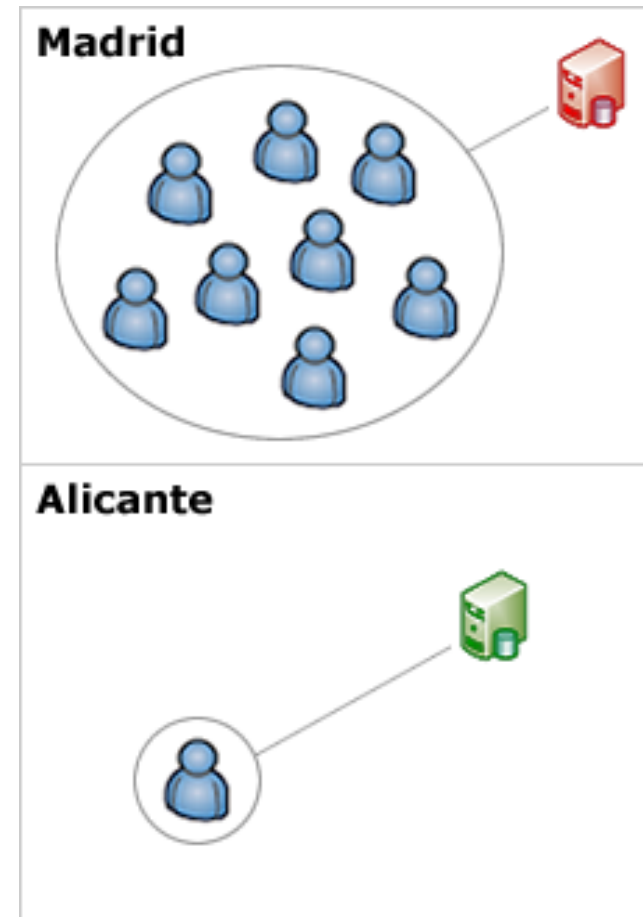


Repositioning



Problem with Network Coordinates

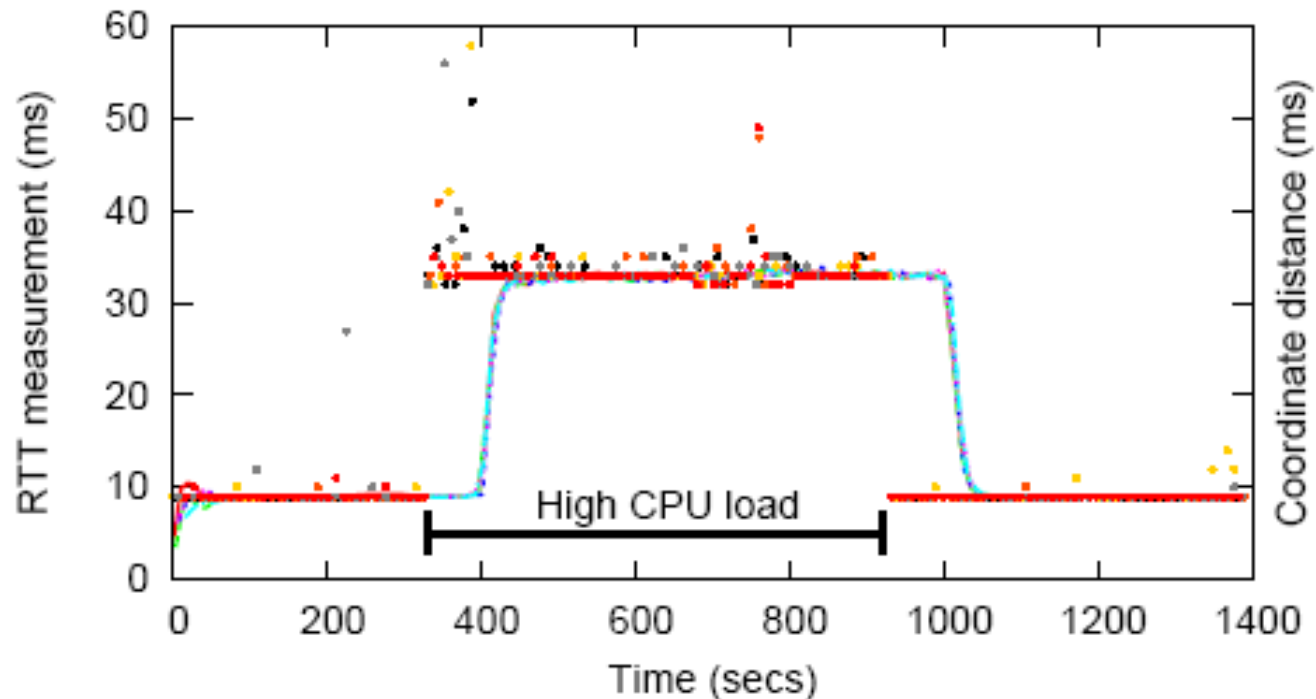
- Too much locality and **No Load Awareness!**
- **Overloading** of servers in **densely-populated** areas
- **Spare capacity** in **sparse** regions



Load-Aware Network Coordinates (LANCs)

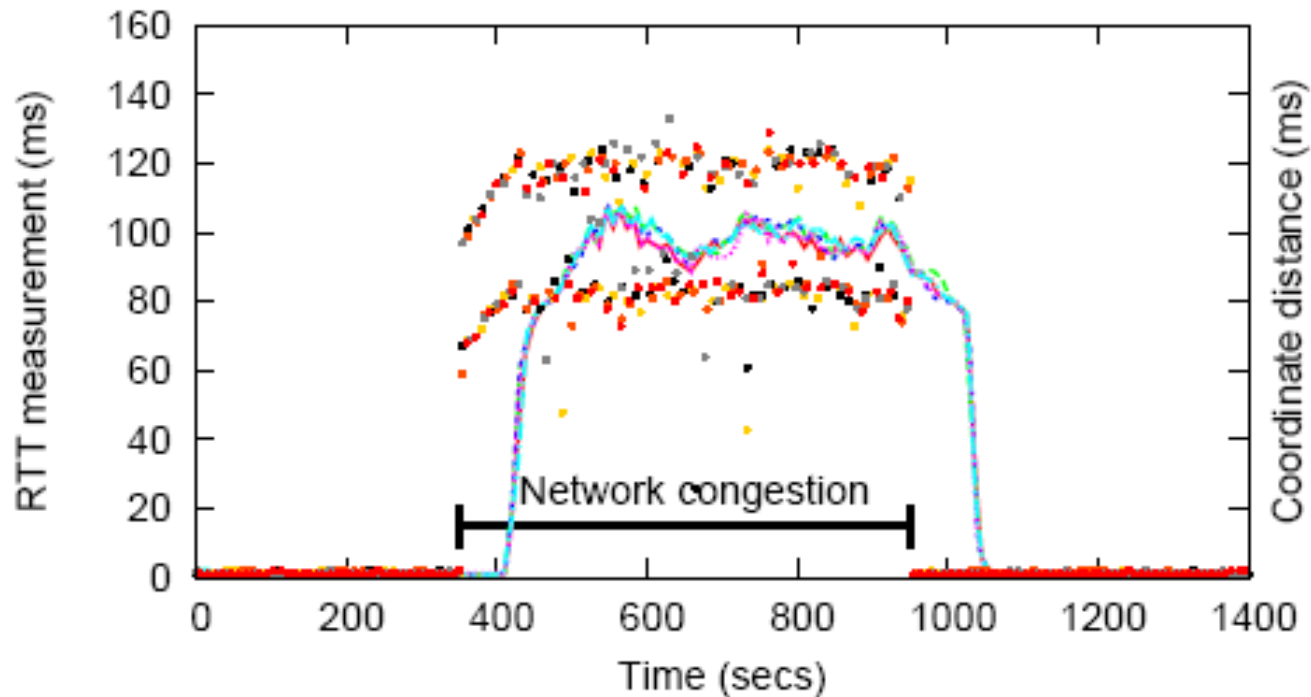
- **Replace network-level** RTT latency measurements (PINGs) with **application-level ones** (applications exchanging data)
- Two load metrics for the price of one
 - **Computational Load** - Schedule application to send and receive data on CPU
 - **Network Congestion** - Internet Standard treats packet loss and network congestion as synonyms

Computational Load



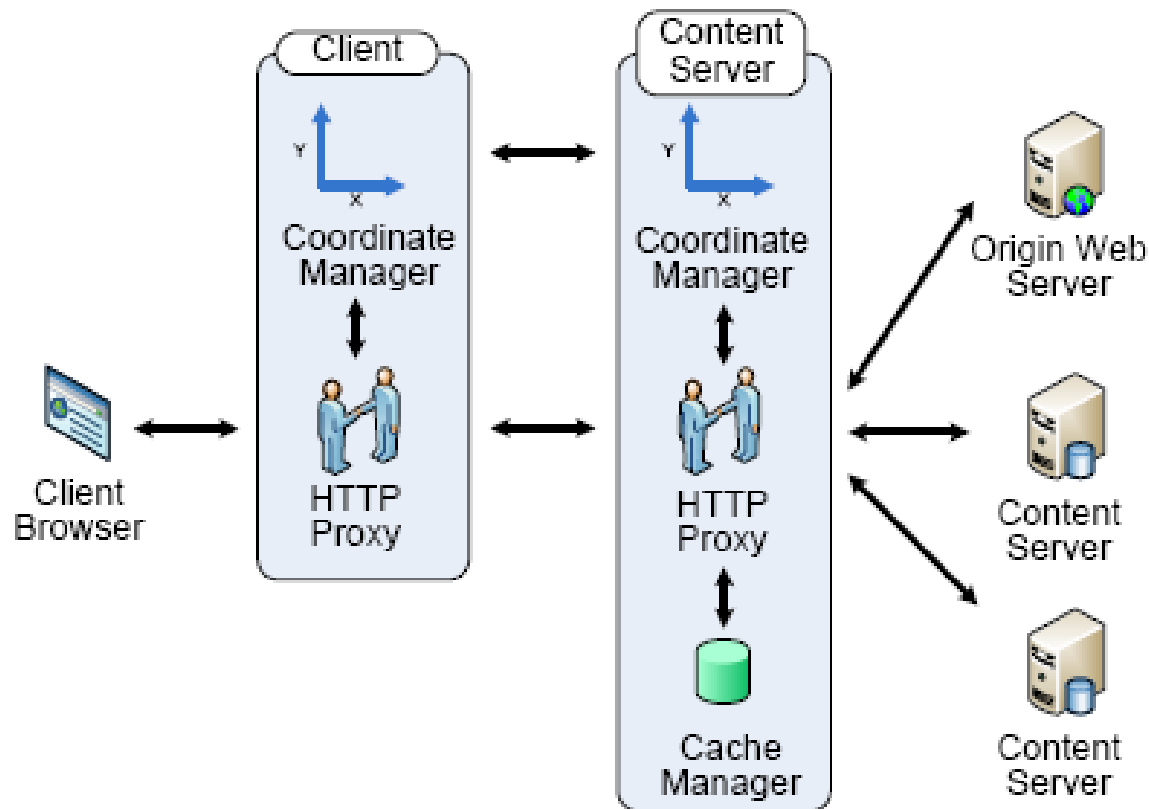
Increase in RTT and coordinate distances under high content server load

Network Congestion



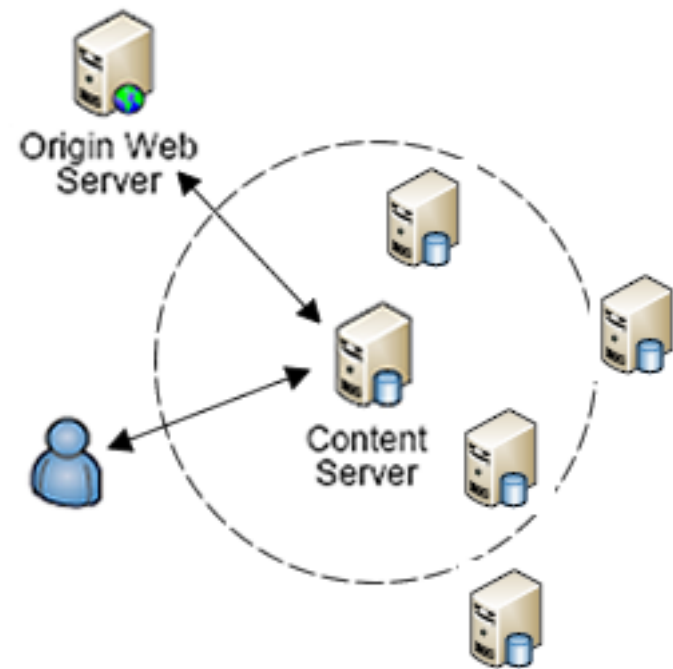
Increase in RTT samples and coordinate distances under network congestion

System Architecture



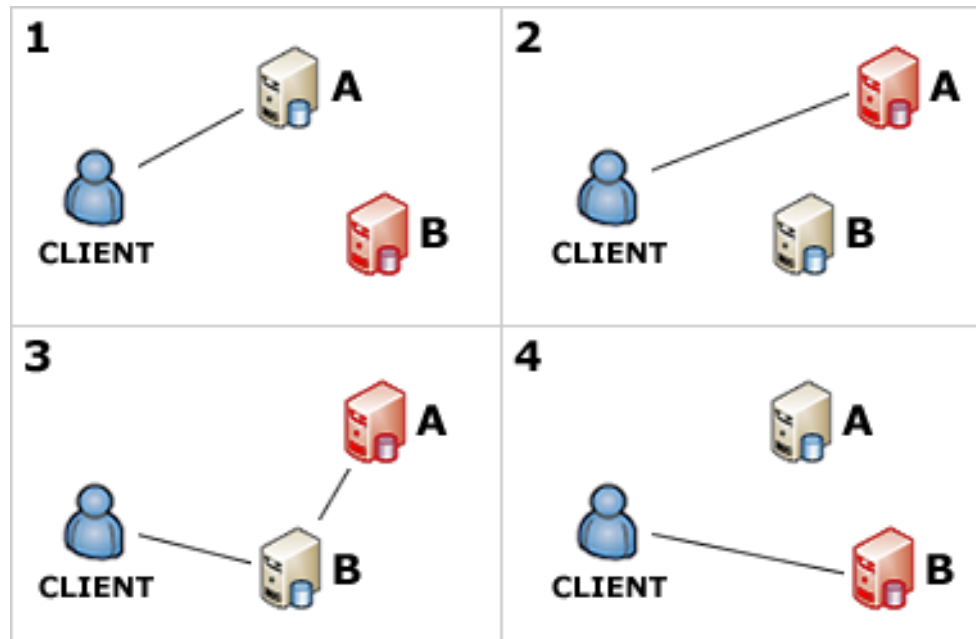
Server-Centric Strategy

- **Closest content server** (primary) asks all other **neighbouring servers** (secondary) within a radius
- Retrieves from origin web server if timeout or global cache miss
- Higher global cache hit rate



Load Balancing & Dynamic Content Replication

- Achieved by the dynamic behaviour of **LANCs**
 - **Load Balancing**
 - **Dynamic Content Replication**



PlanetLab Deployment (1)

- **109 Content servers** world-wide and **16 Clients** at our University
- **5 different configurations:**
 - **LANC+SC** : Server-Centric strategy (2 sec timeout)
 - **LANC+LO** : Local-Only strategy
 - **Nearest** : Single closest content server
 - **Random** : Random content servers world-wide
 - **Direct** : From the origin web server

PlanetLab Deployment (2)

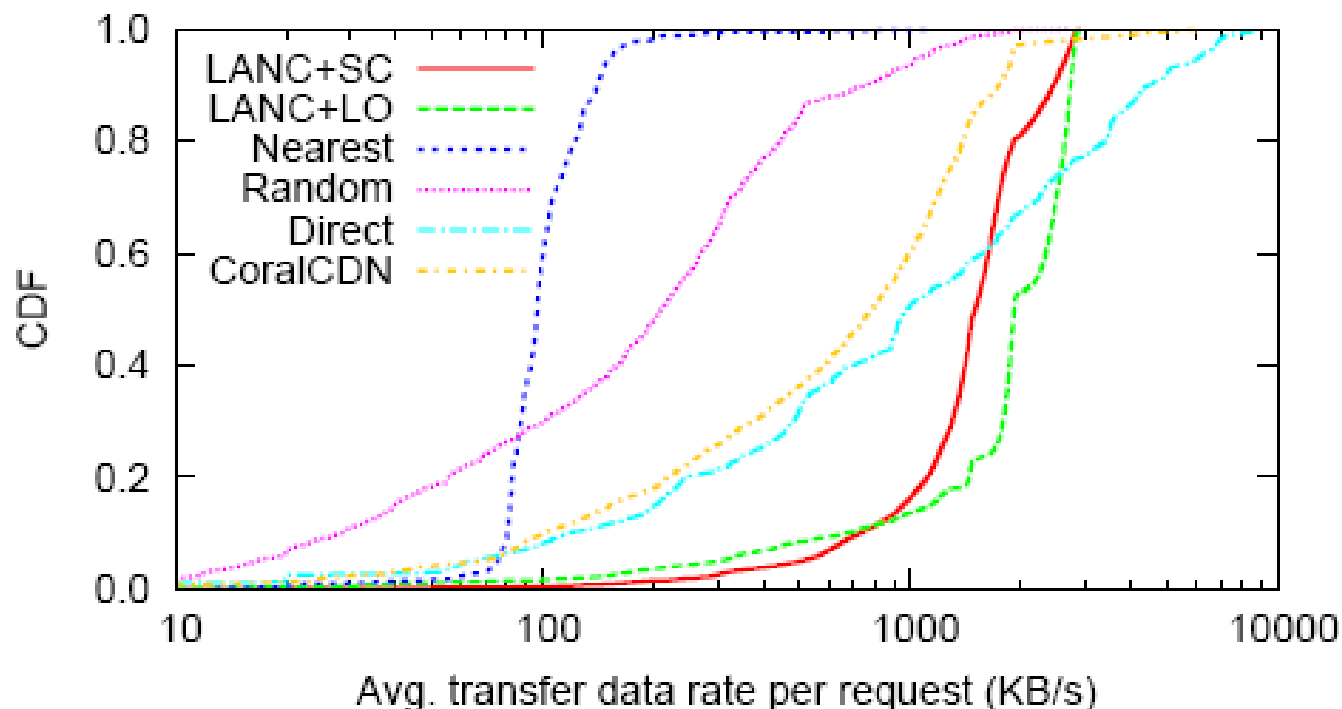
- First content server is the **bootstrap node** for all others
- Stabilizing period of **1 hour**
- RTT Measurement taken with **4KB** every **minute**
- Content servers start with **cold caches**
- Synthetic request workload - Gentoo Linux
Distribution of **3.28 MB** from a list of **100 web servers** world-wide
- Requests are generated for **1 hour**

Evaluation (1)

	LANC+SC	LANC+LO	Nearest	Random	Direct
Total number of requests	12,922	8,524	1,204	1,098	3,331
Cache hit ratio	97.3%	89.2%	90.2%	4.6%	N/A
Avg. request time (secs)	3.20	4.83	35.84	50.33	20.40
Avg. transfer rate (KB/s)	1,533	1,935	106	305	1,778
Median transfer rate (KB/s)	1,500	1,912	96	210	973
90 th perc. transfer rate (KB/s)	2,448	2,726	142	756	4,448

- **LANC+SC** - Highest number of requests handled, cache hit ratio and avg. request time
- **Random** - Lowest number of requests handled, cache hit ratio and avg. request time due to no locality or load awareness
- **LANC+LO** - Highest avg. and median transfer rates due to not having overhead of relaying content from neighbours

Evaluation (2)



CDF plot of the distribution of transfer data rates for six configurations. (Faster is better)

Conclusion

- **Load-Aware Network Coordinates** are:
 - **Low cost** - have load and locality awareness under one simple unified mechanism
 - **Highly Dynamic** - adapt to change in both network paths, network congestion and computational load
 - **Fairly Accurate** - not aiming for perfect accuracy but instead provide consistently good answers at low cost
 - **Scalable** - completely decentralized solution
 - **Versatile** - can be used in a wide range of applications - Game server selection, Security patch distribution etc...
- **LANCs highly competitive** and very useful in a **CDN** where **locality** and **load** awareness is essential

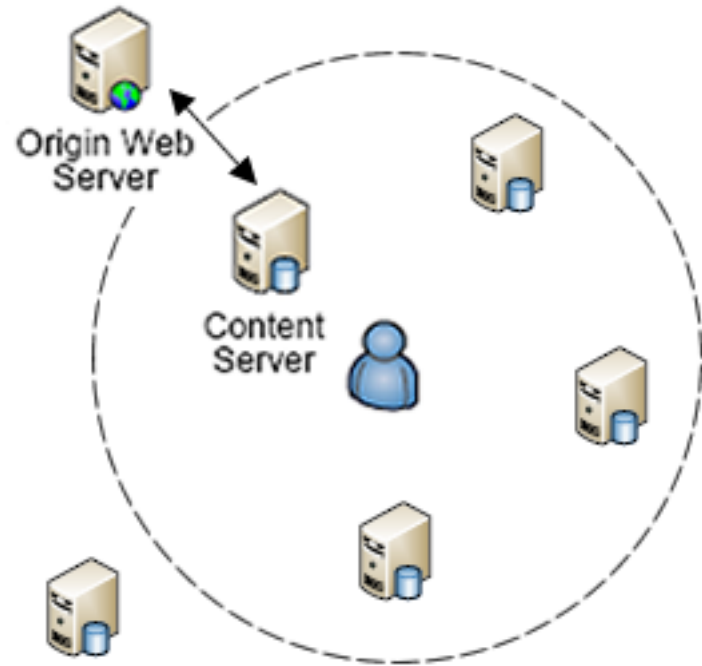
Future Work

- Study the stability and load-awareness of LANCs under dynamic workloads
- Thorough comparison with CoralCDN
- More testing with a variety of redirection strategies and parameters
- Investigate how we can relieve the client of having an application
- **Thank you for listening. Any Questions?**

Extra Slides

Client-Centric Strategy

- **Client** asks **all content servers** within a distance
- Closest content server retrieves from origin web server if timeout or global cache miss
- **No content replication!**
- Slower Response?



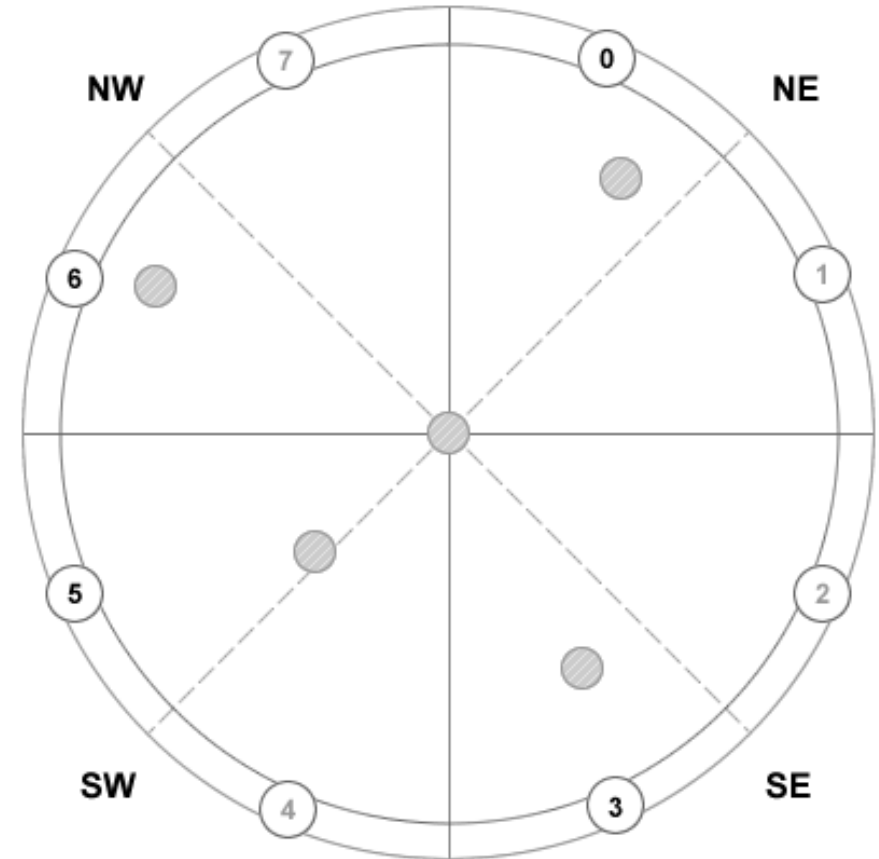
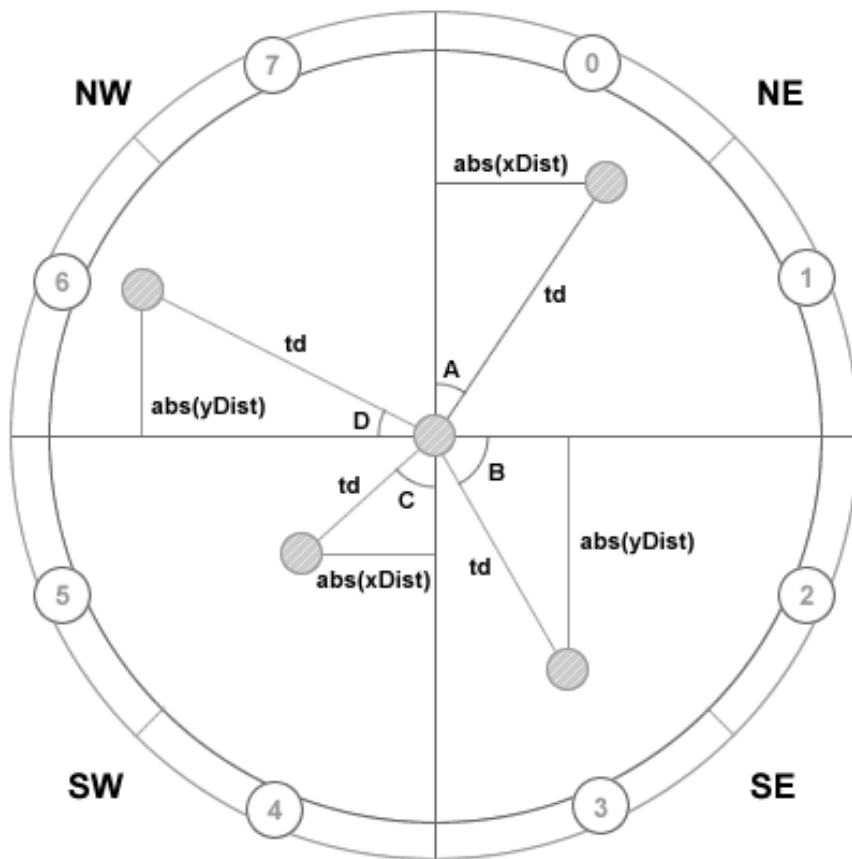
Existing Systems

- **CoralCDN** - DSHT (distributed sloppy hash table)
 - Both locality and load aware but not under one simple mechanism
- **OASIS** - (Overlay-based Anycast Service InfraStructure)
 - Locality aware prioritized, load awareness not considered important
- **Meridian** - On-demand probing
 - Locality aware, extremely accurate, high overhead costs, no load-awareness

Scaled Theta Routing

- Uses a '*sense of direction*'
- 360 degree circle divided into **sectors**
- **Rings** of increasing exponential size
- Implemented using a 2D array with each position as a **zone**
- Divided up into two processes: *whichSector* and *whichRing*

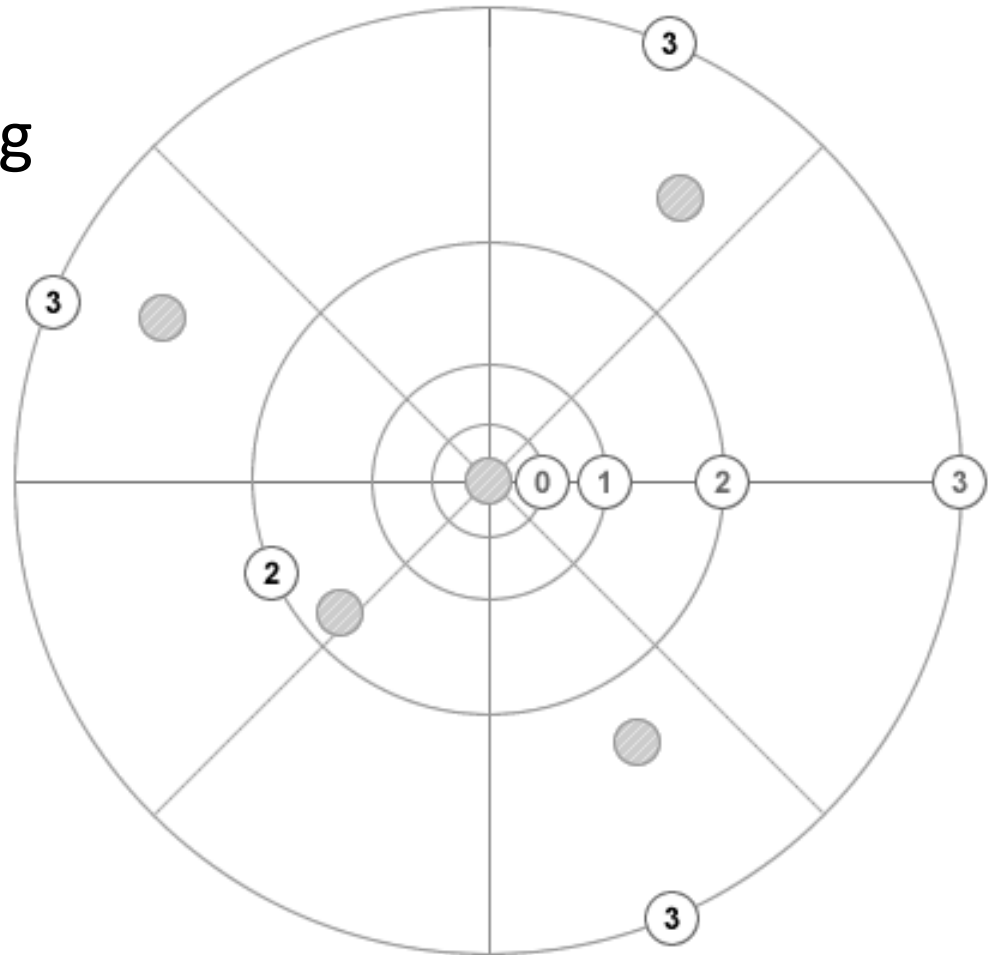
Routing - Which Sector



Number of sectors = 8

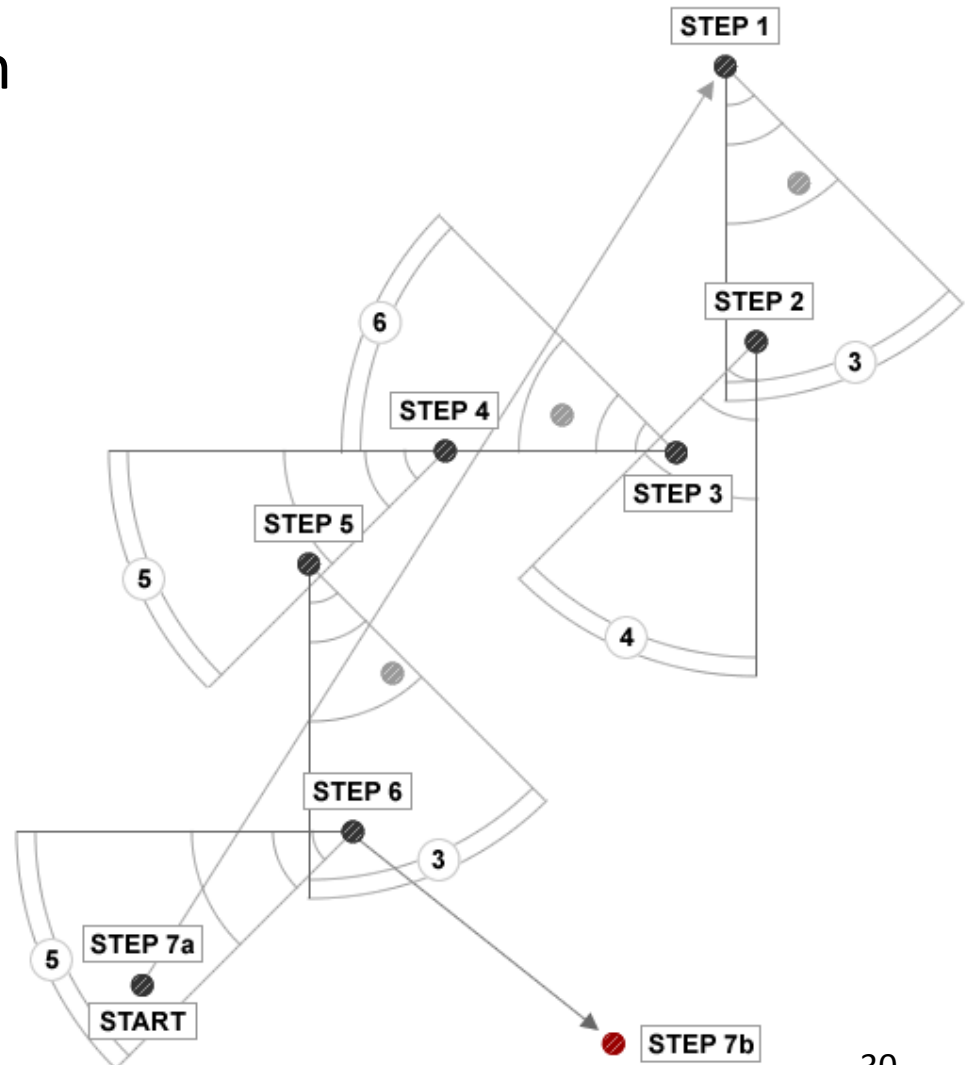
Routing - Which Ring

- Exponentially increasing rings
- Helps keep long distance shortcuts and minimize hop count
- Each ring is an index in routing array

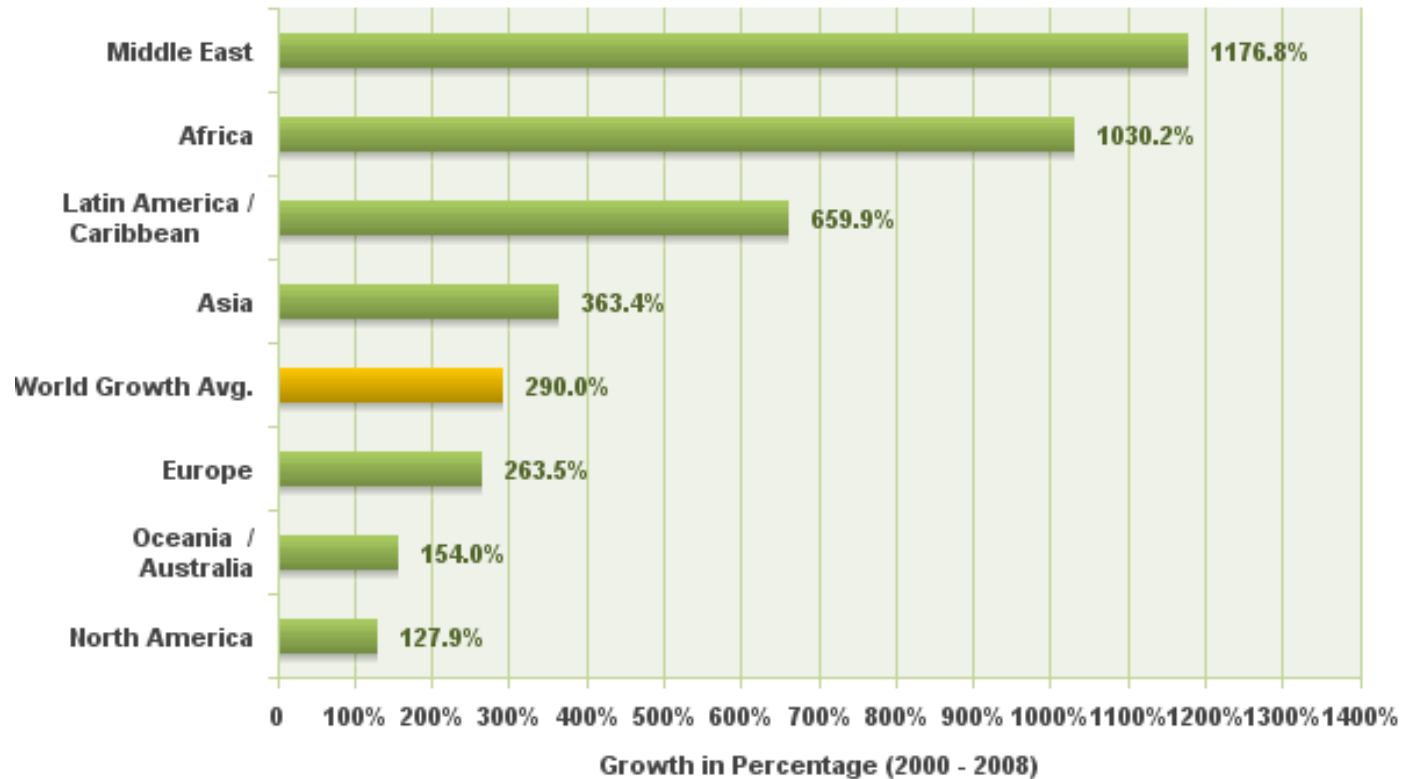


Route Action

- Send a **RoutingMsg** which has locator coordinate (local coordinate)
- Message routed back to **origin** with routing tables appended
- New node then creates links with remote hosts learnt about in message
- Also done when node enters system through a **bootstrap node**



Motivation



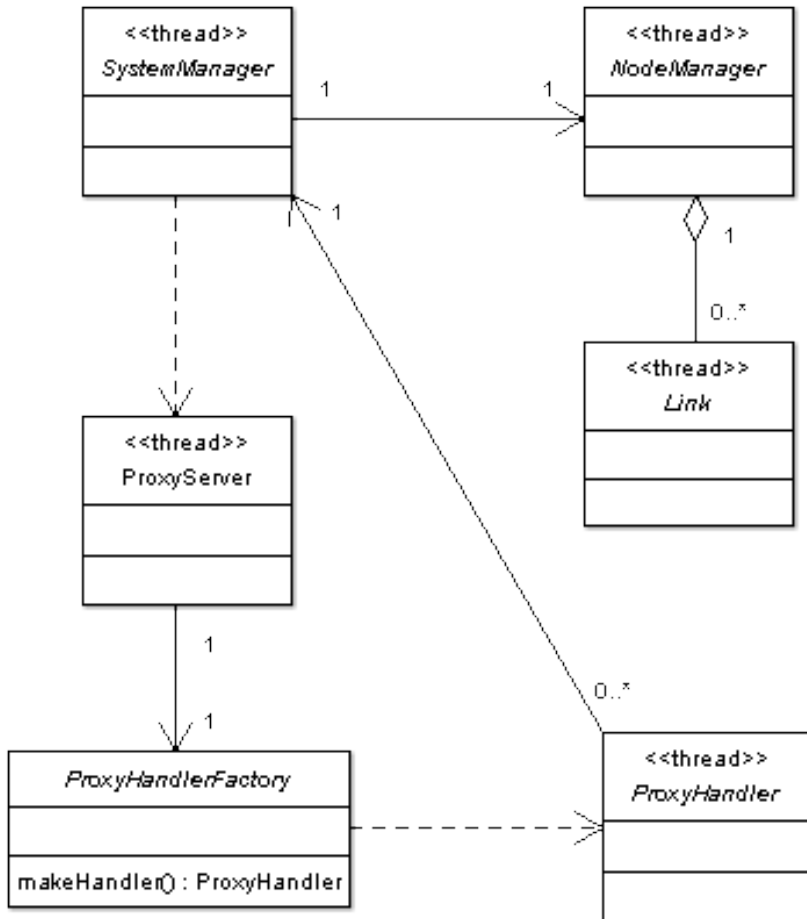
- 1. Huge internet usage growth** - web servers worldwide becoming increasingly overloaded
- 2. Increased dispersion of usage** - network congestion

CoralCDN

Evaluation

	LANC+SC	LANC+LO	Nearest	Random	Direct	CoralCDN
Tot. number of req.	12,922	8,524	1,204	1,098	3,331	4,074
Cache hit ratio	97.3%	89.2%	90.2%	4.6%	N/A	Unknown
Avg. request time (secs)	3.20	4.83	35.84	50.33	20.40	18.41
Median request time (secs)	2.24	1.76	34.84	15.97	3.45	4.01
90 th perc. request time (secs)	4.57	4.97	41.81	117.58	28.26	19.55
Avg. transfer rate (KB/s)	1,533	1,935	106	305	1,778	876
Median transfer rate (KB/s)	1,500	1,912	96	210	973	741
90 th perc. transfer rate (KB/s)	2,448	2,726	142	756	4,448	1,739

Design



Simplified Abstract Class UML View

- Abstract Factory Design Pattern for Handlers
- NodeManager with multiple Links
- Links are ReceiverWorkers or Senders
- ProxyHandler Threads come from a Pool