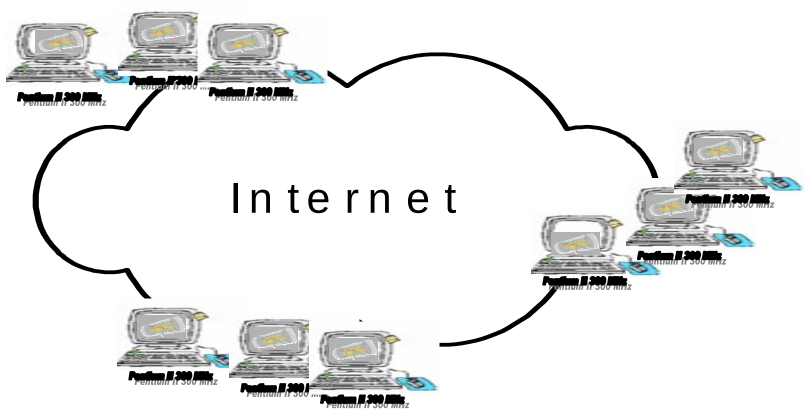
Locating a Resource Set with Desired Network Connections in a Large Resource Pool

Chuang Liu
Department of Computer Science
University of Chicago

Problem

- A resource pool: A set of internet-connected resources accessible for users to run their applications.



PLANETLAB

Grid3

U.S. ATLAS

- Find a set of R resources such that the network latency between any pair of those resources is **less than** (**more than**) L milliseconds

Challenges

- Resource pools such as Gnutella, PPlive, etc.
  - Large number of resources
  - Resources join the resource pool **incrementally**

- Challenges to locate a resource set
  - Space (Store latency between all resources)
  - Traffic (Measure latency between all resources)
  - Dynamic
  - NP hard (Find a resource set)

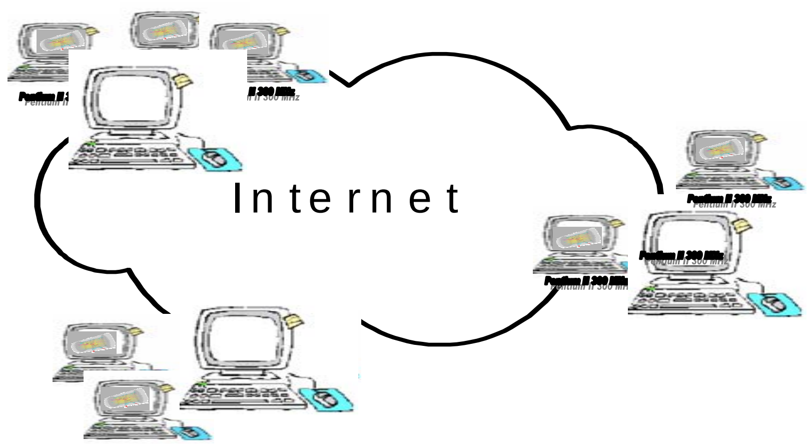
Outline

- **How to represent and store latency information?**
  - $O(N^2)$ --> $O(N)$
- How to measure latency information?
  - $O(N^2)$ --> $O(N * \log N)$
- How to update the latency information?
  - Incremental
- How to locate a resource set based on latency information?
  - $O(N^R)$ --> $O(N)$

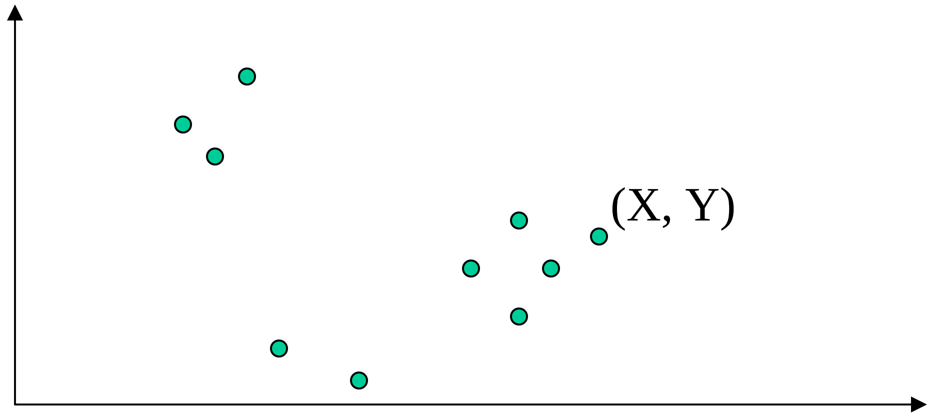
- Conclusion

Existing Methods

- Landmarks

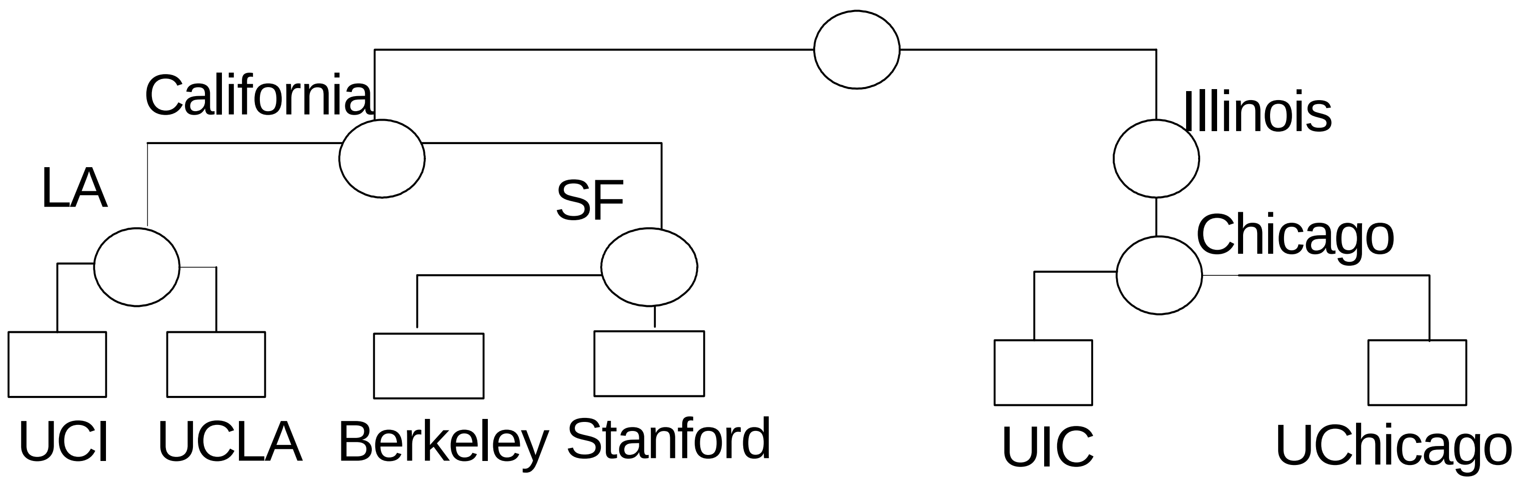


- Coordinate-based methods [Infocom'02, IMC'03]



Hierarchical Cluster Structure

- Hierarchical cluster structure



- Members in a cluster have similar latency among them
  - Instead of storing pair-wise latency, we store average latency for each cluster. O(N^2) → O(N)

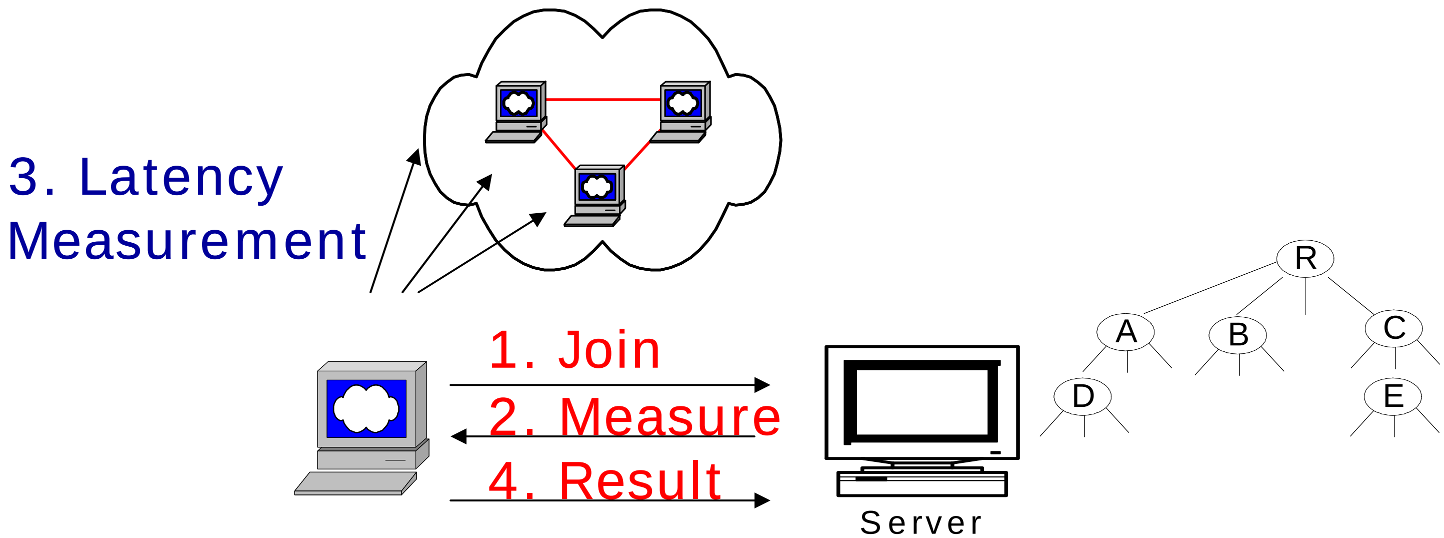
How to build a hierarchical structure on the fly?

Outline

- How to represent and store latency information?
- **How to measure latency information?**
- **How to update the latency information?**
- How to locate a resource set based on latency information?

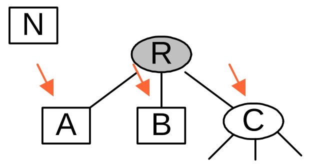
- Conclusion

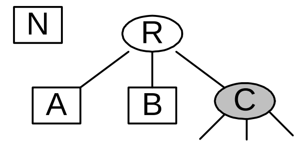
Incremental Cluster Algorithm – System Structure



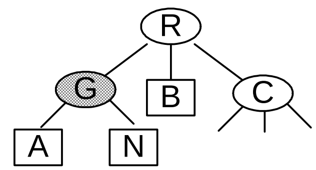
- **Latency measurements** between nodes
- **Messages** between server and nodes

Incremental Cluster Algorithm



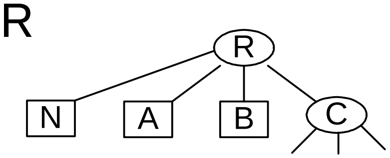


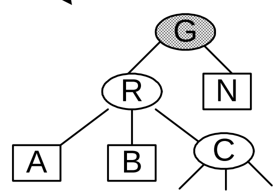
N is **g-times** closer to C than any member of R



N is **g-times** closer to A than any member of R

The distance between N and members of R is similar to average distance in R





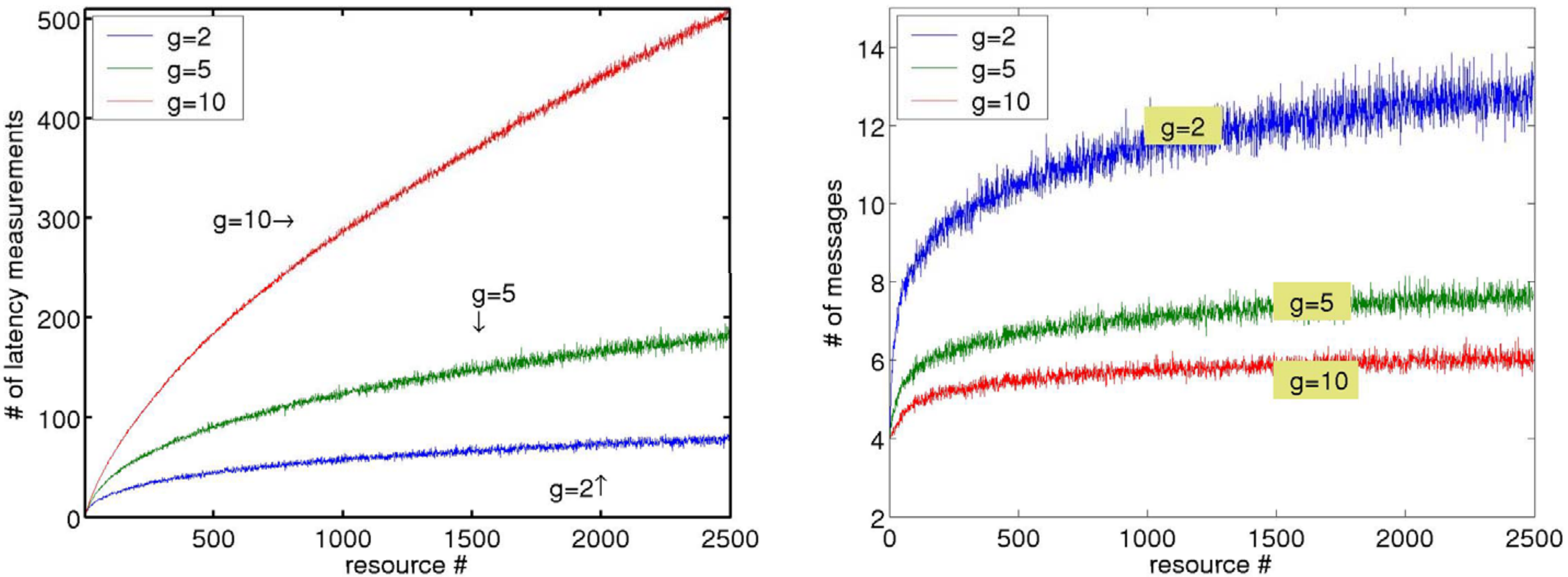
Distance between between N and any members of R is **g-times** bigger than average distance in R

Need to decide the value of **g**

Traffic – Experimental Settings

- Create a cluster structure for 2500 randomly chosen DNS servers on the Internet [SIGCOMM'05]
- For each DNS server joining the cluster structure, we count
  - Number of latency measurements
  - Number of messages between the server and nodes
- Repeat the experiment 100 times. Each time, we randomize the joining sequence of DNS servers
- Calculate the average number of messages and latency measurements in 100 runs.

Scalability



- Number of Latency measurements and messages increase logarithmically
- We choose g=2 for our algorithm

Outline

- How to represent and store latency information?
- How to measure latency information?
- How to update the latency information?
- **How to locate a resource set based on latency information?**

- Conclusion

Existing Methods

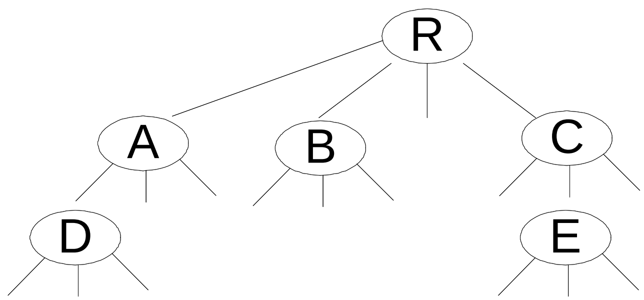
- Tree search algorithm [HPDC'05]
  - Start with an empty set
  - Repeatedly pick from available resources one resource that has required connections with current members in the set, and adds it to the set
  - Roll back the addition in previous step if no such resource exists
  - Finish when the set contains all required resources
- Modified tree search algorithms [CLUSTER'05][SIGCOMM'05]

Complexity: $O(N^R)$

Needs to know pair-wise latency

An Approximate Search Algorithm

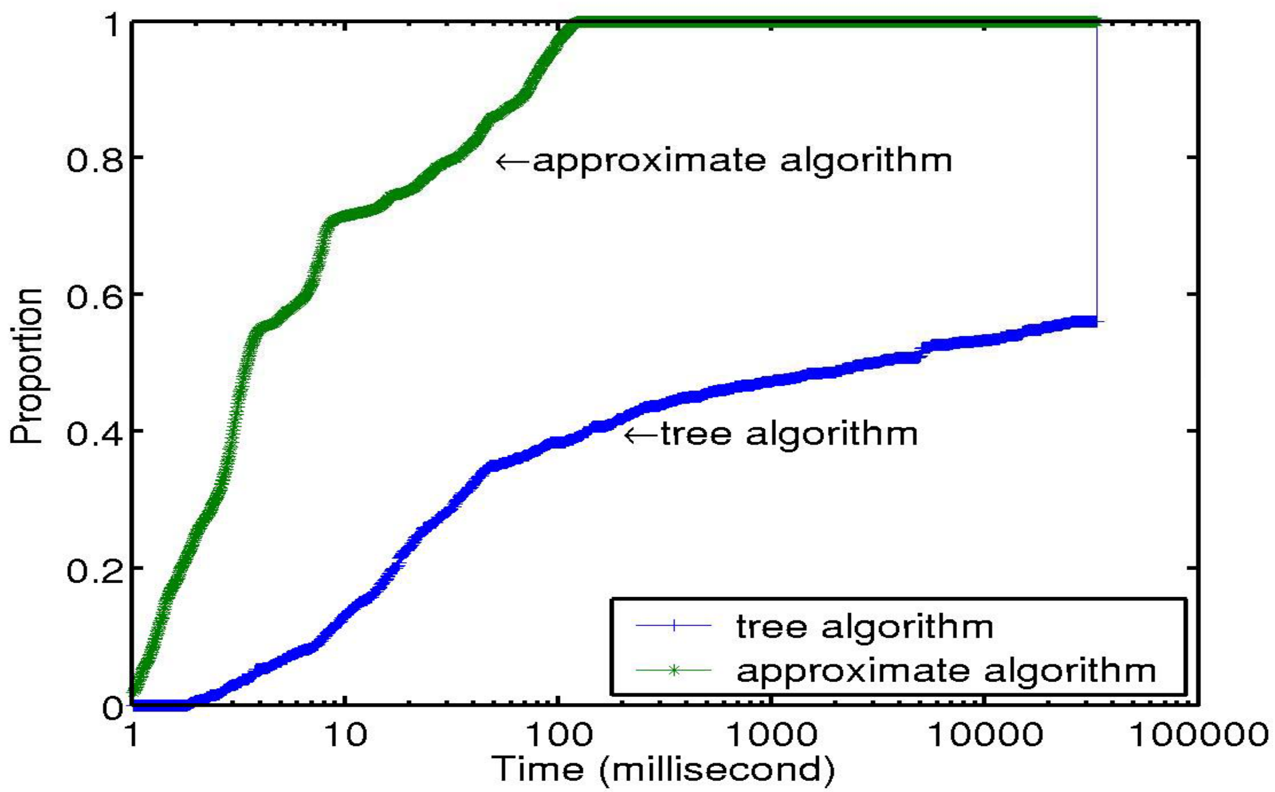
- Algorithm
  - Start from the root
  - Checks each cluster. If a **cluster's average latency satisfies the constraint** in the query, returns a result.
  - Finishes when all clusters have been checked or results have been found



Complexity: O(N)

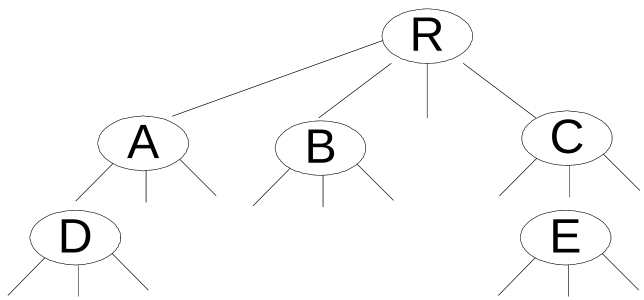
Response Time

- Q1 searches for R resources with latency between any two of them ***smaller*** than L
- Build 1000 different Q1s by randomly choosing values for R and L
- Apply to 2500 DNS servers



An Approximate Search Algorithm

- Algorithm
  - Start from the root
  - Checks each cluster. If a **cluster's average latency satisfies the constraint** in the query, finds a result.
  - Finishes when all clusters have been checked



Wrong answers

Accuracy

- Secure factor S (S= 1, 2, 3)
  - If a cluster's average latency is **S time better** than the requirements in a query, finds a result

- Accuracy Metrics
  - Return Ratio (RR): Percentage of queries whose results are found
  - False Positive Ratio (FPR): Ratio of inaccurate results.

Experimental Results

- Apply benchmark queries on 2500 DNS servers

Algorithm	S=1	S=2	S=3	tree	rank
RR	77%	62%	44%	58%	77%
FPR	28%	15%	2%	0%	2%

- Rank-based search algorithm
  - Checks if a **cluster's average latency** is **1, 2, or 3 times better** than the requirements in a query
  - Return one result with the highest secure factor

Rank-based algorithm achieves RR similar to the algorithm with S=1, and FPR similar to the algorithm with S=3

Summary

- A **incremental cluster algorithm** to create the cluster structure when resources join the resource pool
  - Small storage space O(N^2) → O(N)
  - Little network traffic Log(N)

- **An approximate search algorithms** to search for a resource set with desired aggregation properties and network connections
  - Order-of-magnitude(s) faster than current methods
  - Return many more queries with small error rate

- **Questions?**

- **Thank you**