Self-organizing Wide-area Network Caches

Samrat Bhattacharjee **Ken Calvert** **Ellen Zegura**

Networking and Telecommunications Group
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, USA

`http://www.cc.gatech.edu/projects/canes`

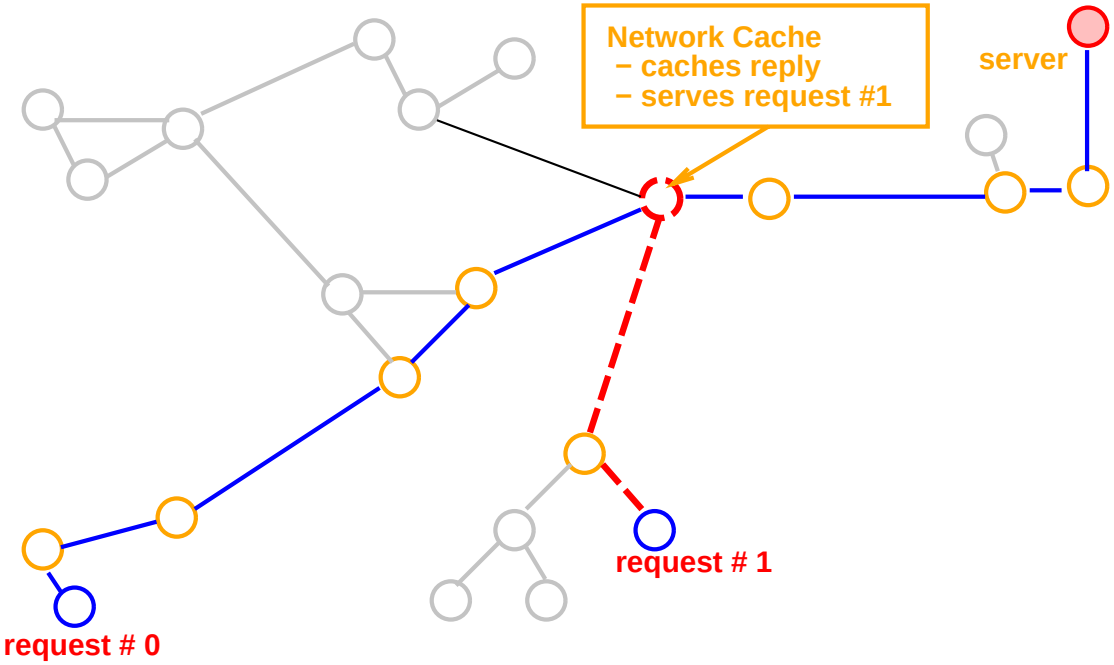
Sponsors: DARPA, NSF

Outline

- Problem statement
- Caching approaches
- Self-organizing caching schemes
- Results
- Conclusions

Problem Statement

Improve client access latencies by associating caches with routers within the network



Develop *algorithms* to co-ordinate network caches in order to reduce latency

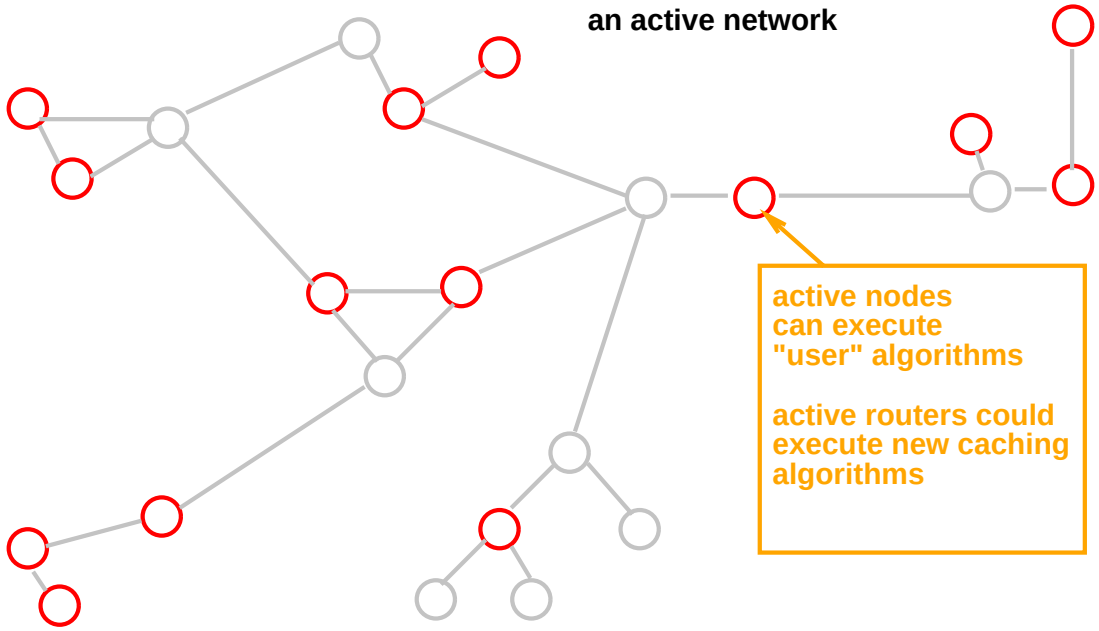
Wide-area Caching

- Assumptions
  - Request-response paradigm
  - Requests can be served by caches
- Components
  - *Location of caches*
  - *What to cache and where*
  - *Rules for forwarding requests*
  - Protocols to communicate between caches
  - Cache consistency
- Examples: Harvest, ICP, Squid, Netcache

Develop *algorithms* to coordinate caches to reduce latency

Active Networks and Network Caching

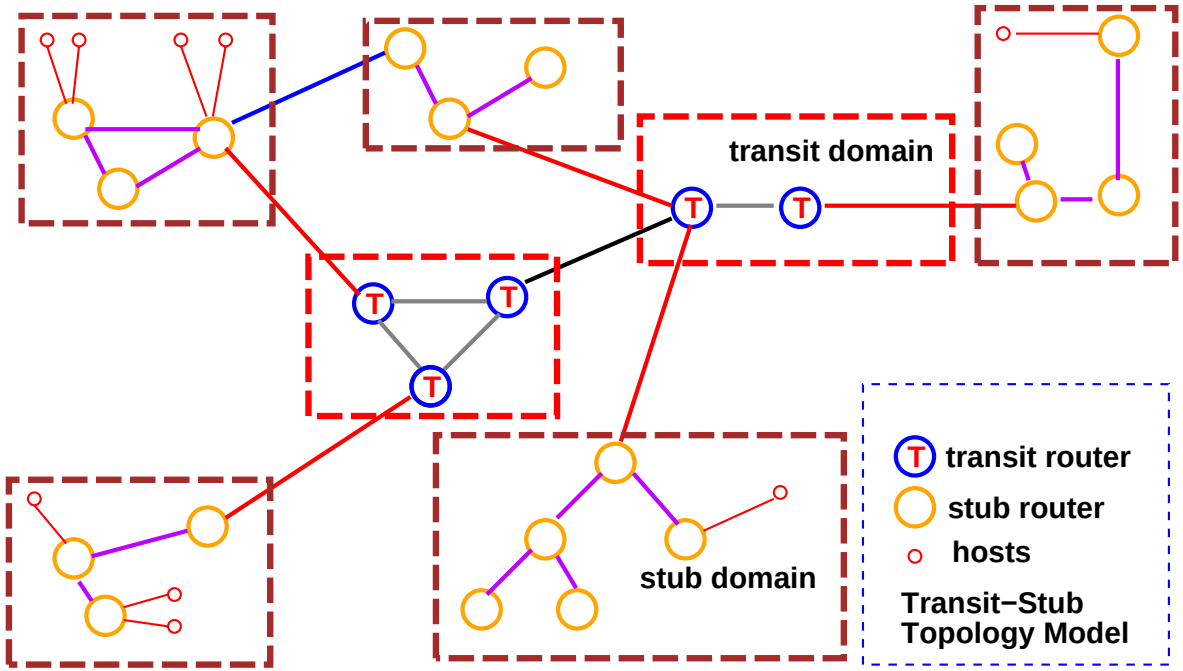
Active Networks provide a programmable network platform



Use active networking to instantiate self-organizing cache management algorithms within the network

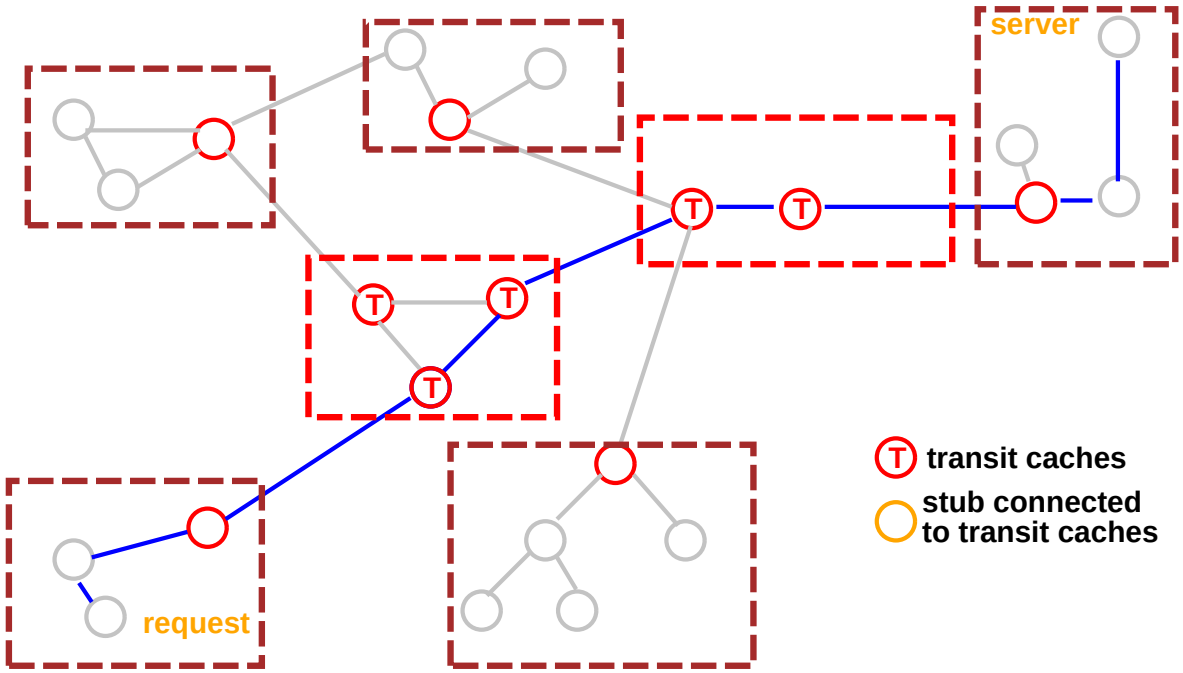
Caching Model and Topology Model

- Clients, Servers, Popular Items
- Transit-Stub Network Topologies



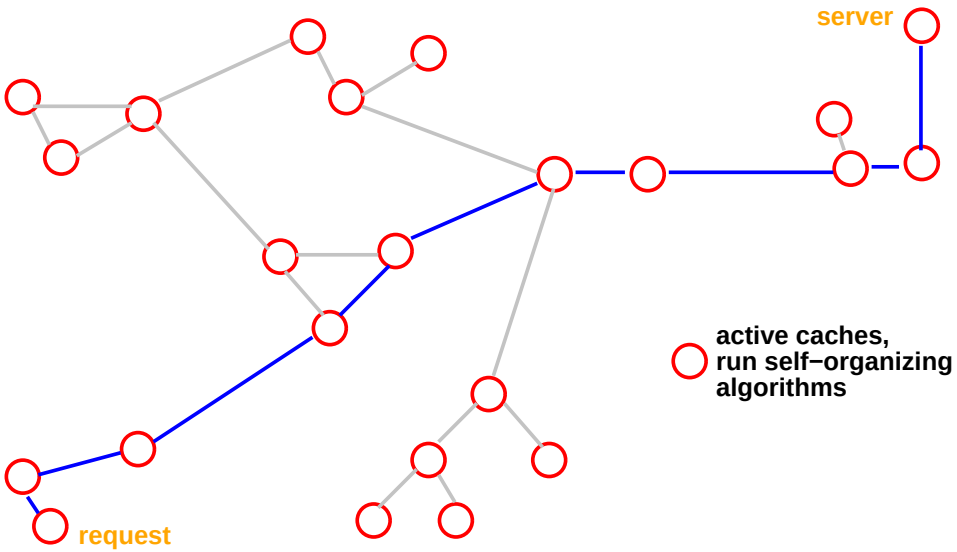
Approaches towards Caching

- Caching by location
  - At transit nodes — *Backbone* routers
  - At border stub nodes — *Access* routers



Self-organizing Network Caches

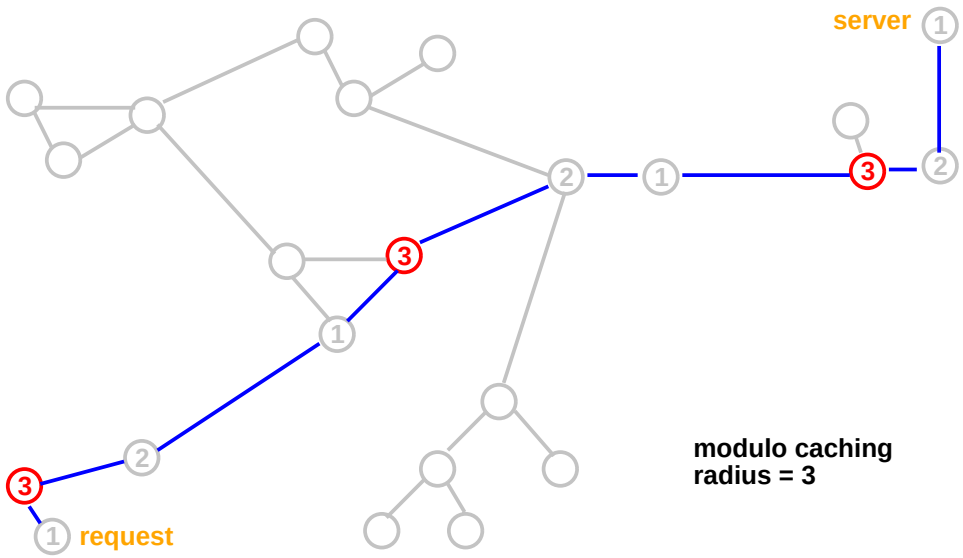
- Small caches without regard to location
- Effective use of widely distributed caches is difficult



Challenges: Where to cache, how to forward requests?

Modulo Caching

- Spread item over client-server path
- *Cache radius* — modulo caching

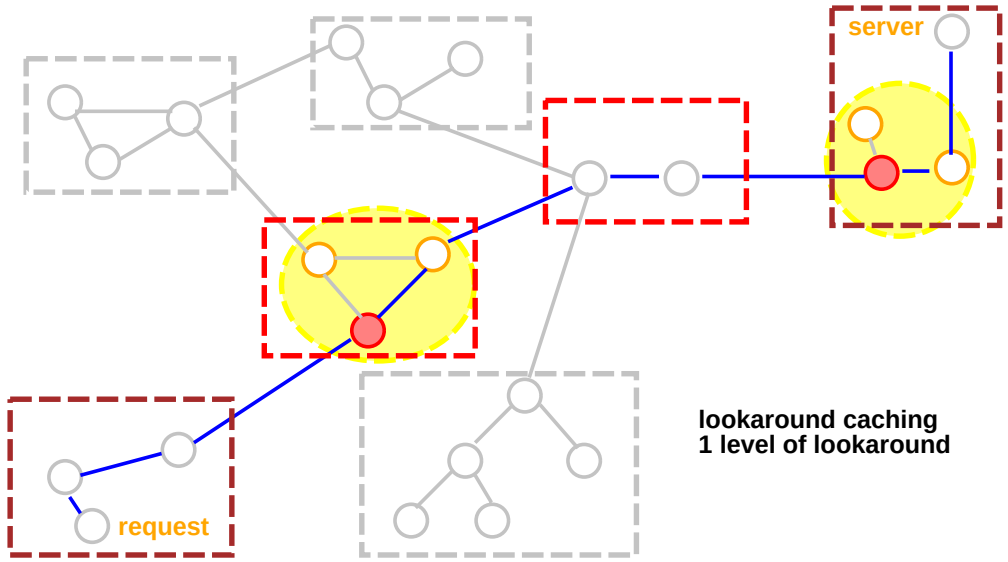


- Item is cached once on server-client path every *radius* hops

Cache Lookaround

Observation: *Item location size* $\ll$ *average item size*

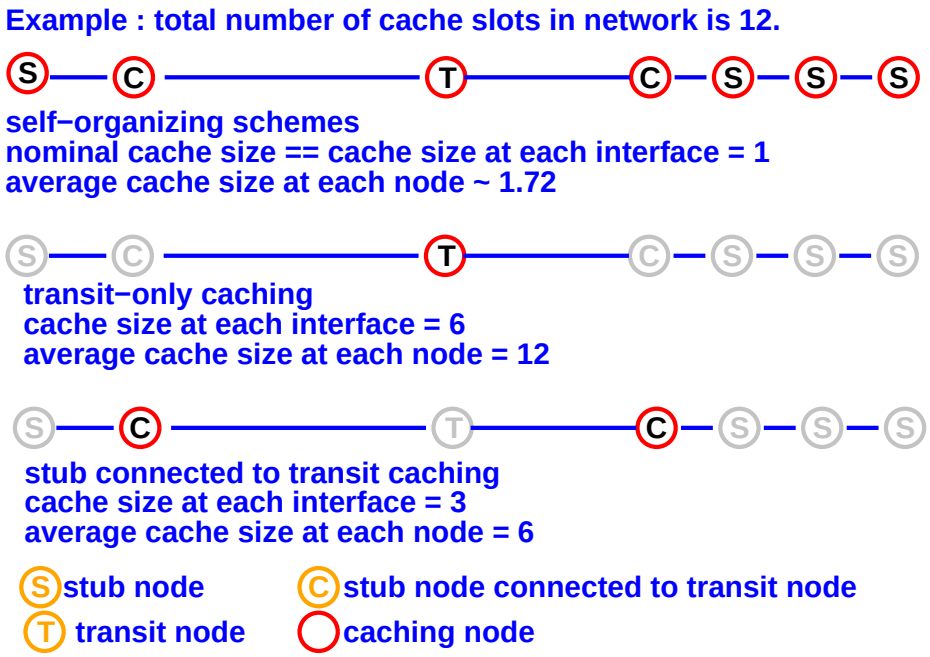
- Use some item memory to store location of *nearby items*
- Trade cache memory for location information



- Request may be *deflected* (even off the shortest path to server) to caches within lookaround radius

Simulation Methodology

- Total cache size constant across methods



- Base Topology — 1500 nodes, average degree 3.71
  - Average transit-only cache 25 *times* larger than average active cache, average SCT cache 4.25 times larger

Result: Varying Cache Size

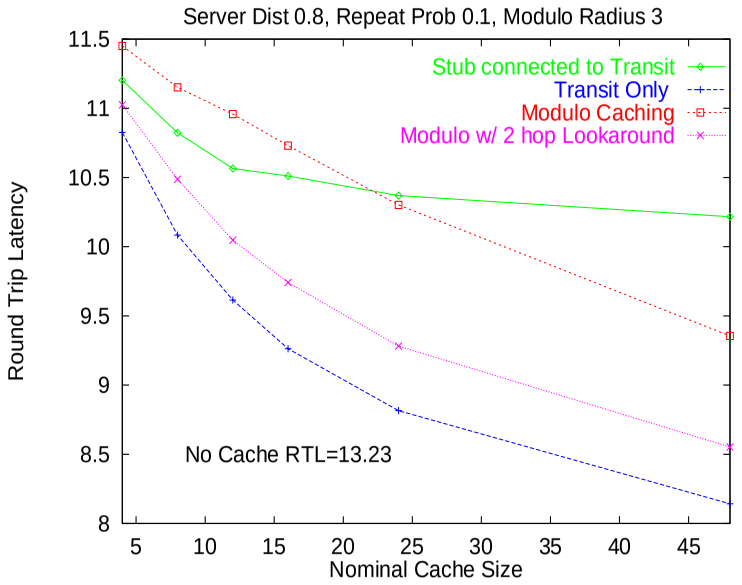


Figure 1: Low Access Correlation

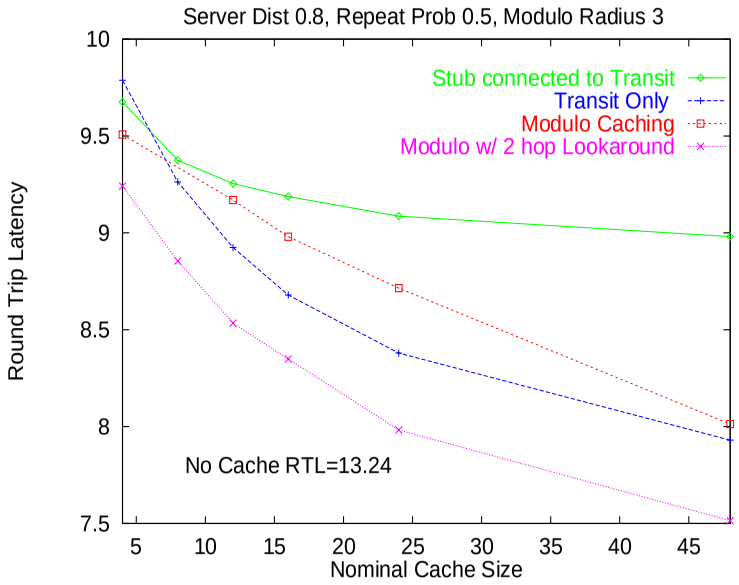
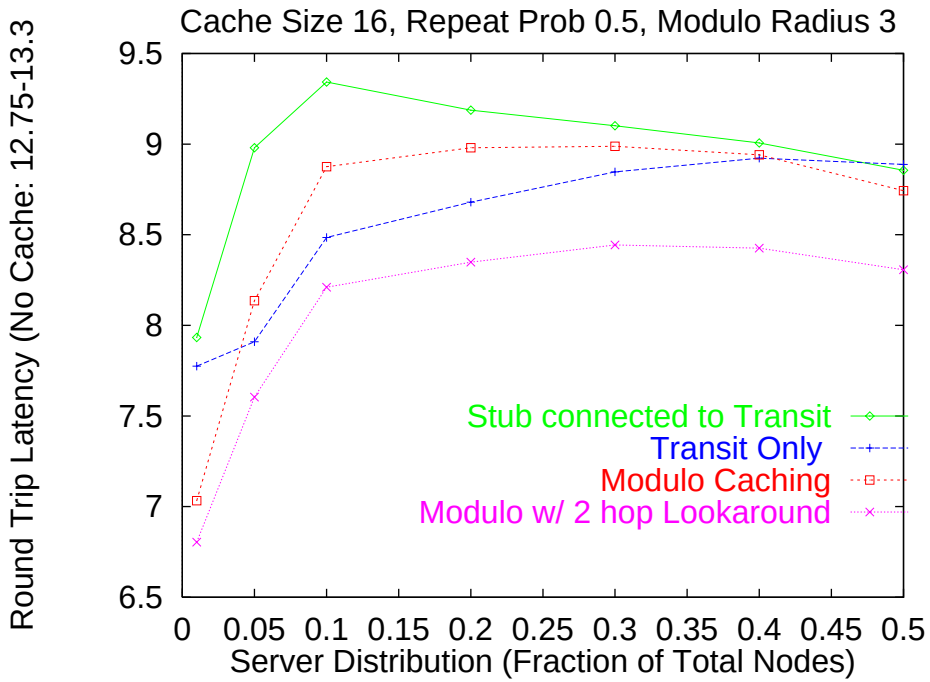


Figure 2: High Access Correlation

- SCT caches are not able to reduce latency as well
- Self-organizing schemes perform better as cache size increases, and as access correlations increase

Result: Varying Server Distribution



- Self-organizing schemes are robust against server distribution

Result: Varying Topology and Access Patterns

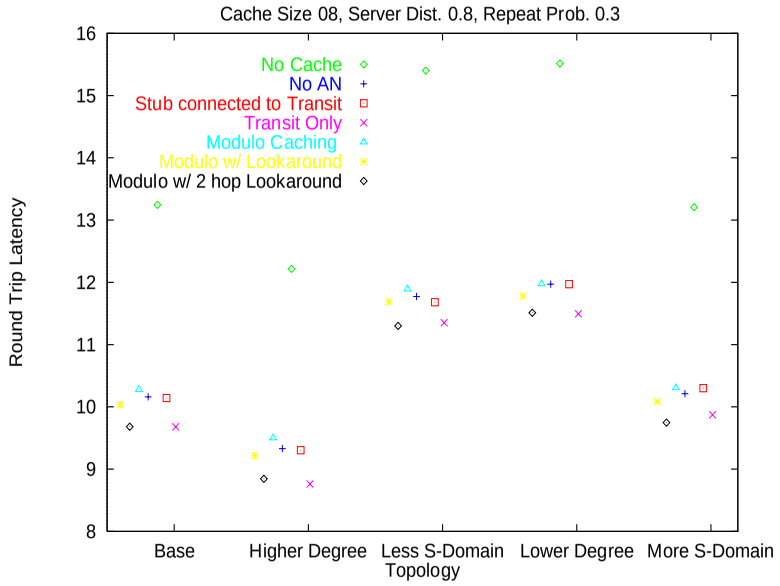


Figure 3: Different Topologies

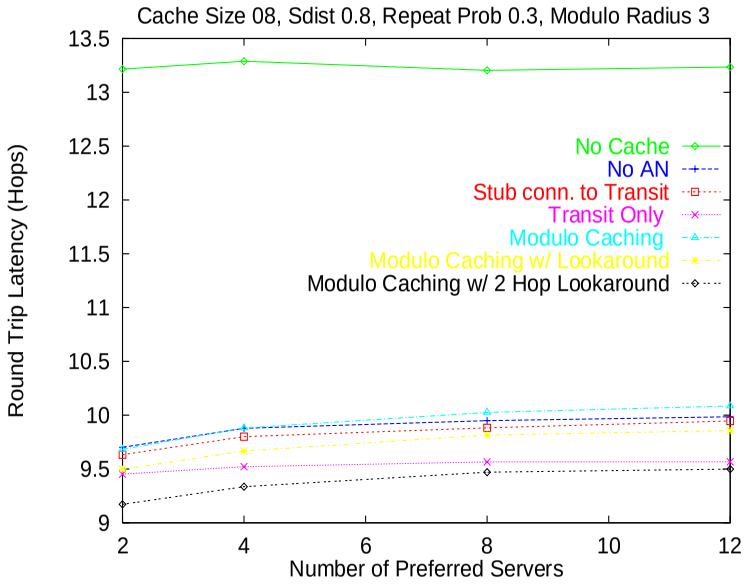
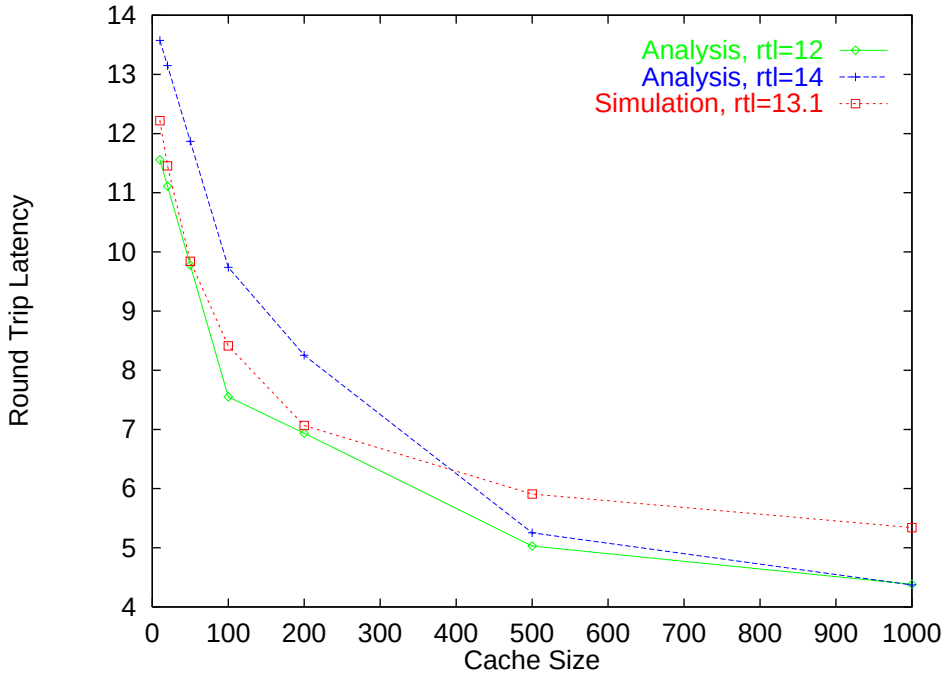


Figure 4: Spatial Access Pattern

Analysis

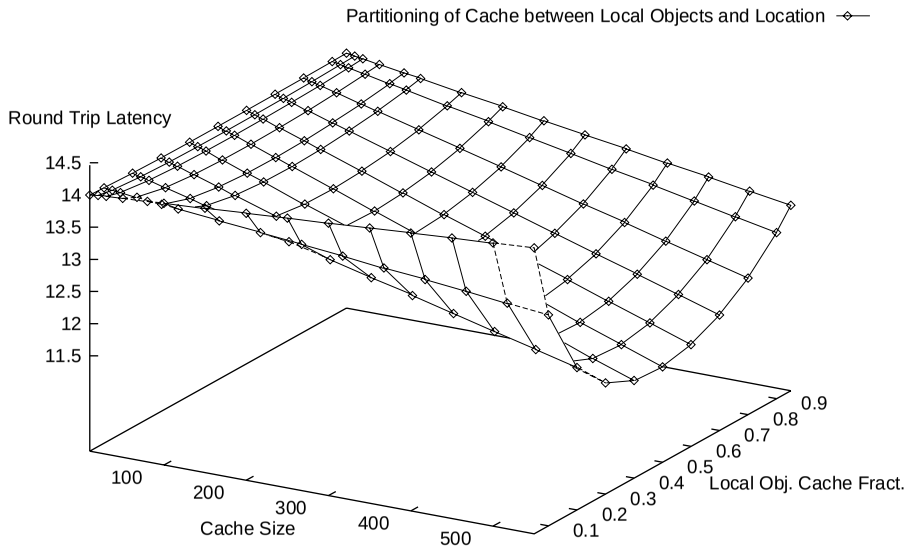
- Probabilistic analysis of cache performance
- Expressions for latencies for various methods



Comparison of analytic and simulation results — Lookaround Caches: Base Topology, 50 location pointers per item, 2 levels of lookaround

Analysis of Lookaround Benefit

- Caches space is partitioned into data and location pointers
- Question: What is the *optimal* amount to devote location pointers?



Valley in plot due to benefits of lookaround caching

Conclusions and Future Work

- Self-organizing schemes reduce latency using far smaller individual caches
- Self-organizing schemes are robust against varying access schemes, topologies, server distributions
- Self-organizing algorithms are an active network application that is independent of individual users

- Increase *activity* of in-network processing:
  - Utilize per-application, per-user semantics
  - Increase per-item state in the network
- Active Network implementation using ANTS