



Mobile Computing &
Networking Laboratory

On Cooperative Caching in Wireless P2P Networks

Jing Zhao, Ping Zhang and Guohong Cao
MCN Laboratory
Pennsylvania State University
June 19th, 2008

PENNSTATE



Outline



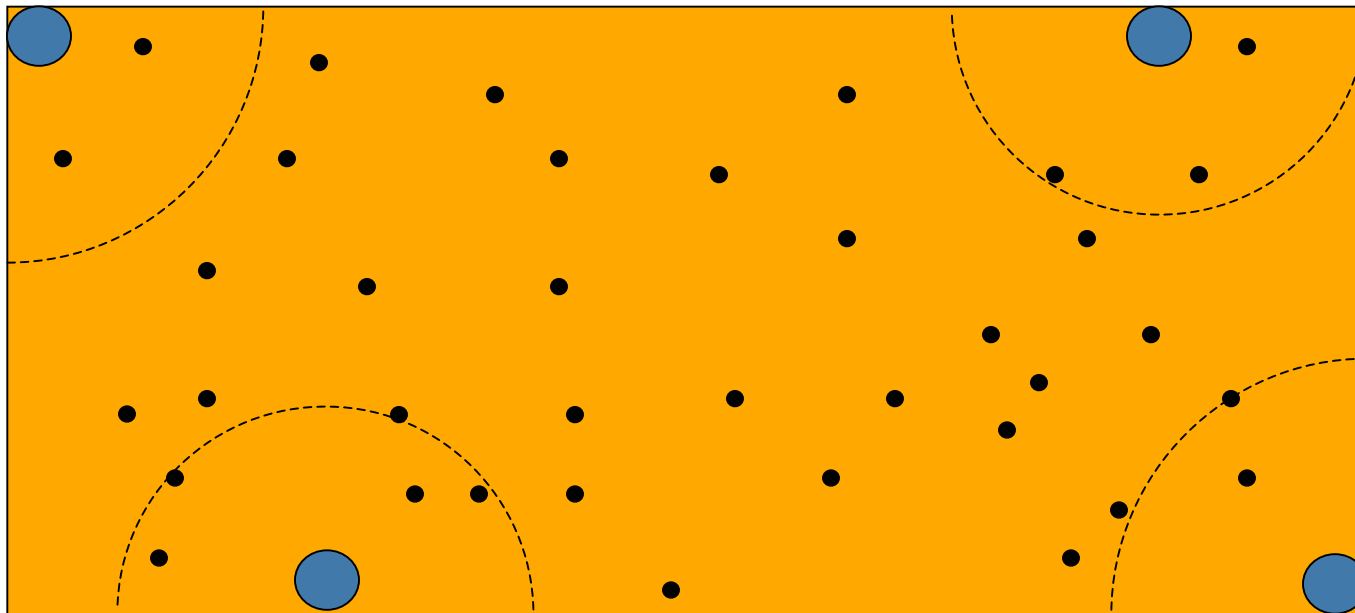
- Introduction
- Design issues
- System architecture
- Prototyping and experiments
- Performance evaluation
- Conclusions

Cooperative caching

- Cache:
 - Locally buffer data to serve future access.
 - Local decision.
- Cooperative cache:
 - Multiple nodes coordinate with one another to cache data and share the cached data.
 - Cached data not only replies local access, but also replies data request issued from other nodes.
 - Collaborative decision
- Reduce the number of hops that request/data need to travel in the network
 - Reduce the data access delay
 - Reduce the power & bandwidth consumption.
 - Reduce the workload of the gateway node or data center.

An Example

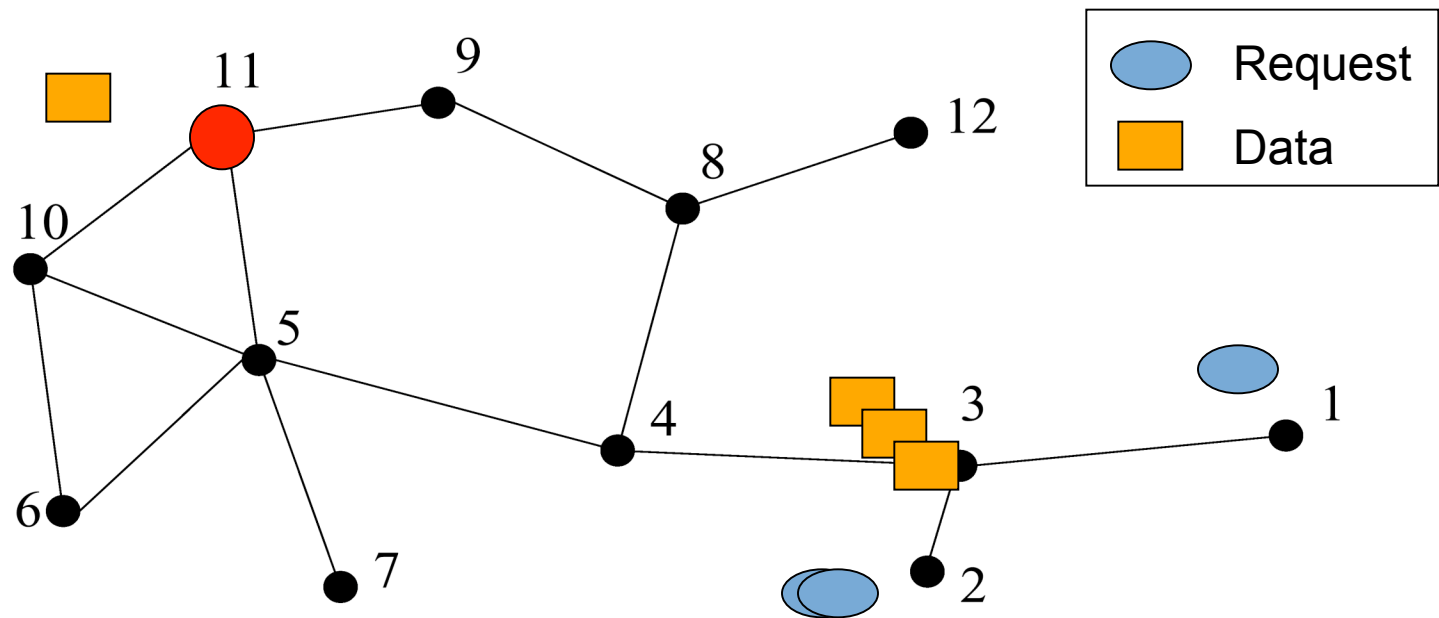
- Infostations are deployed to provide info, such as maps, attractive sites, or restaurant info to mobile users. Users may relay for each other to serve those not directly covered by the infostations.
- Users also share data got from the infostations.



Contributions

- First work to present the design and implementation of cooperative caching in wireless P2P networks.
 - Cooperative caching in wireless network has been studied at a very high level and the evaluation are limited by pure simulation.
 - Implementation of cooperative caching has only be seen in Internet and Web environment.
- Propose an asymmetric caching approach to reduce the overhead of caching.
- Study the effects of single channel 802.11 network and multi-interface multi-channel mesh network on the performance of cooperative caching.

Cooperative Caching Scheme [Infocom'04]



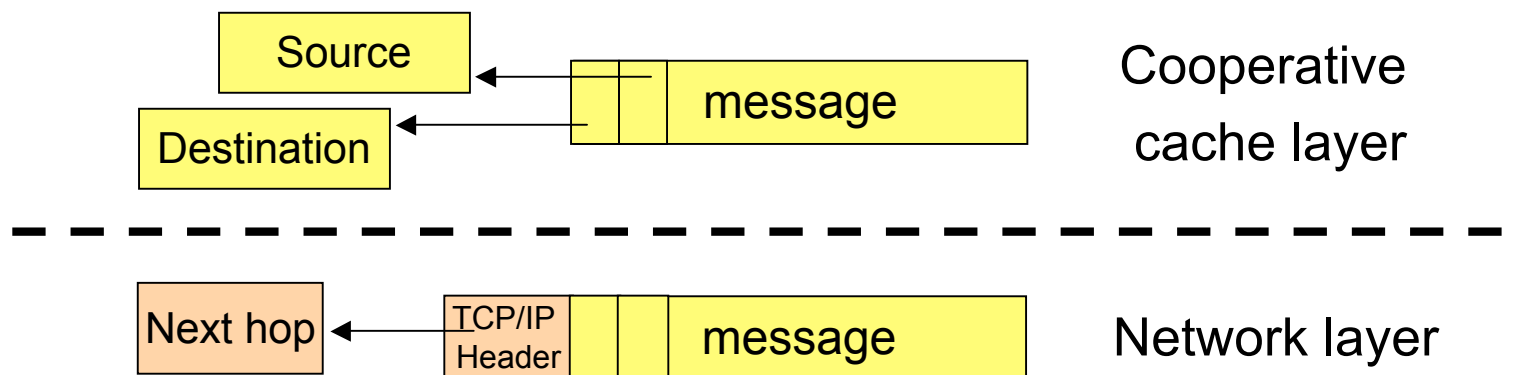
- Passing-by data are cached to serve future requests.

Design Cooperative Cache

- Basic operation
 - Intermediate forwarding nodes check every passing-by packet to make caching decision
 - Not provided by existing ad hoc routing protocols
- First Approach: Integrated Design
 - Cache functionalities are integrated into the network layer.
 - Kernel space implementation
 - Pros
 - Close to routing daemon
 - Straight forward
 - Cons
 - Make the kernel complex
 - Increase kernel memory usage
 - Increase software maintenance work

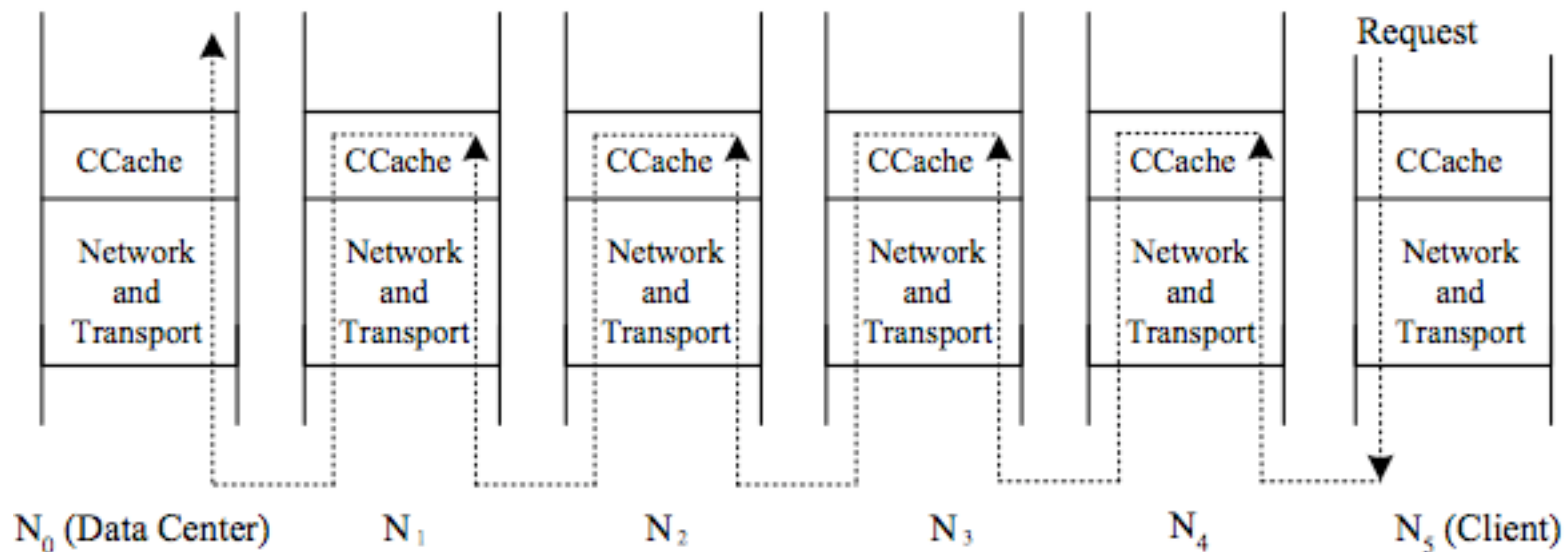
Layered Design

- Dedicated cooperative cache layer as a middleware between application layer and network (TCP/IP) layer.
- User space implementation
- Transmit message hop-by-hop in user space level
 - Fetch next hop address from routing table
 - The destination address to transmit a message is the next-hop address instead of the destination of message



Packet Flow in Layered Design

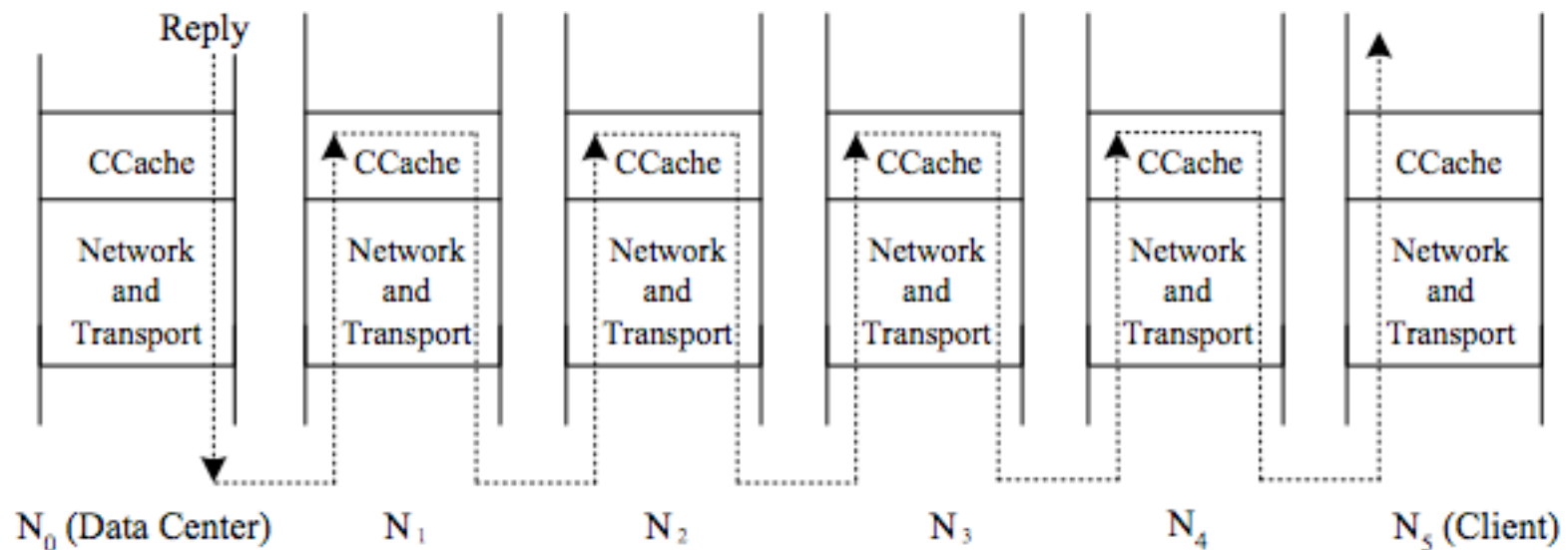
- Request packet



(a) The request packet flow

Flow in Layered Design

- Reply data packet



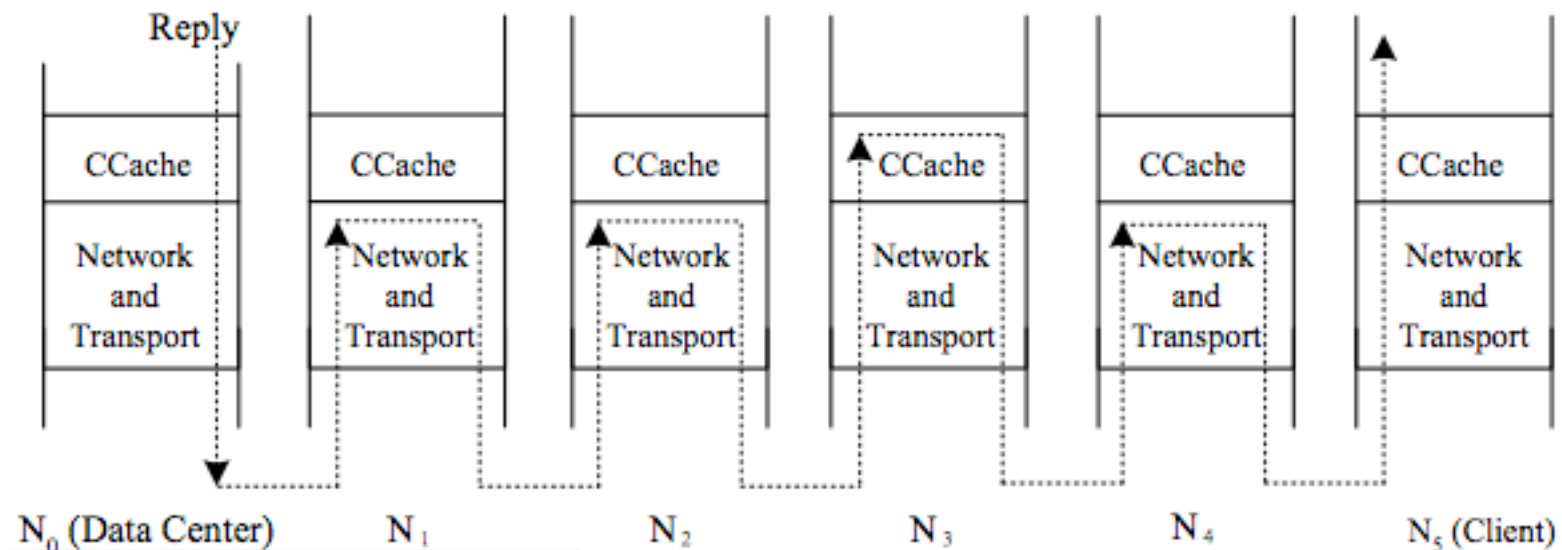
(b) The reply packet flow

Pros and Cons of Strict Layered Design

- **Benefit**
 - Higher hit rate, better performance
 - Flexible option
 - Independent of routing protocols, less maintenance work
- **Problems of Strict Layered Design**
 - Use system calls to transfer data between userspace and kernel space
 - Packet goes to the cache layer hop-by-hop
 - Fragmented large data are re-assembled at each hop, eliminates any chance of pipeline.

A Modified Layered Caching Approach

- Asymmetric Caching Strategy
 - Request message goes to the cache layer hop-by-hop
 - Reply message only goes to the cache layer of the nodes which need to cache data.
- Benefit
 - Minimize the cache layer processing overhead
 - Allow data packet pipeline between two caching nodes

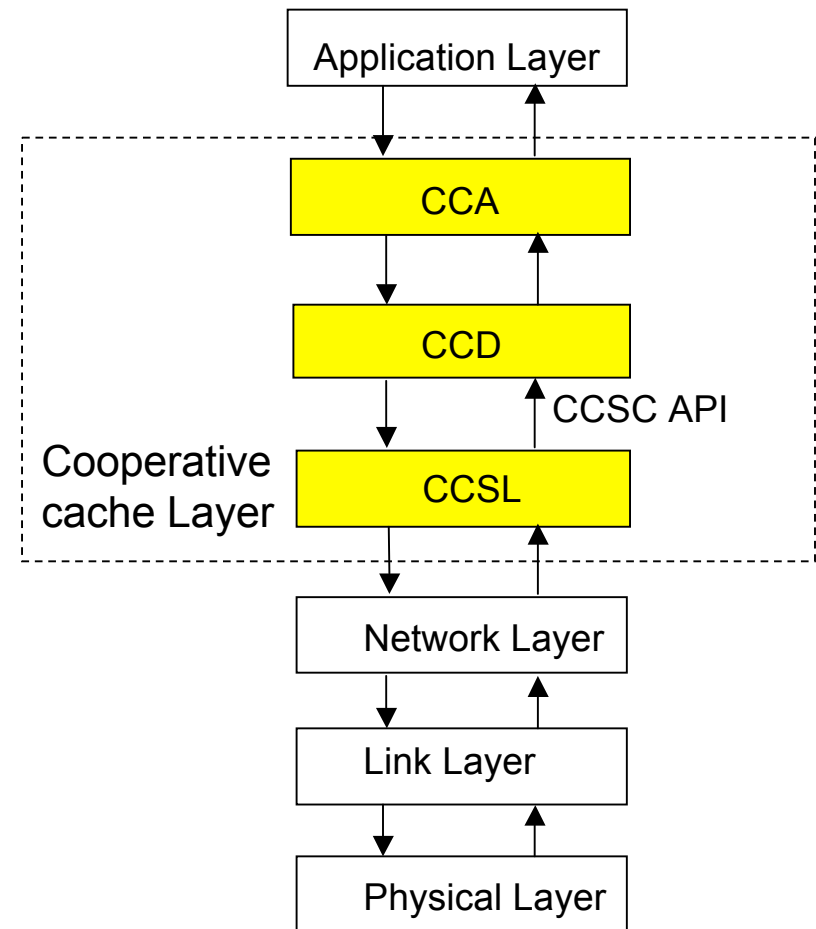


Asymmetric Caching

- Request forwarding
 - Each forwarding node checks whether it is a local cache hit
 - Each forwarding node decides whether or not to cache
 - Add its ID in *Cache_List* if it wants to cache
 - Wraps the message, using the next hop id as the destination address.
 - The data source node examines *Cache_List*
 - Copy *Cache_List* to the header of the reply message.
- Forwarding the Data Reply
 - Tunneling data reply between the nodes in the *Cache_List*
 - Each caching node wraps the reply packet, uses the next caching node id as the destination address.

Cooperative Cache Implementation

- System Architecture
 - Cooperative Cache Agent (CCA)
 - Cooperative Cache Daemon (CCD)
 - Cooperative Cache Supporting Library (CCSL)
 -
- Cooperative Cache Agent (CCA)
 - Wrap the application message and generate the standard cache layer message.



System Components

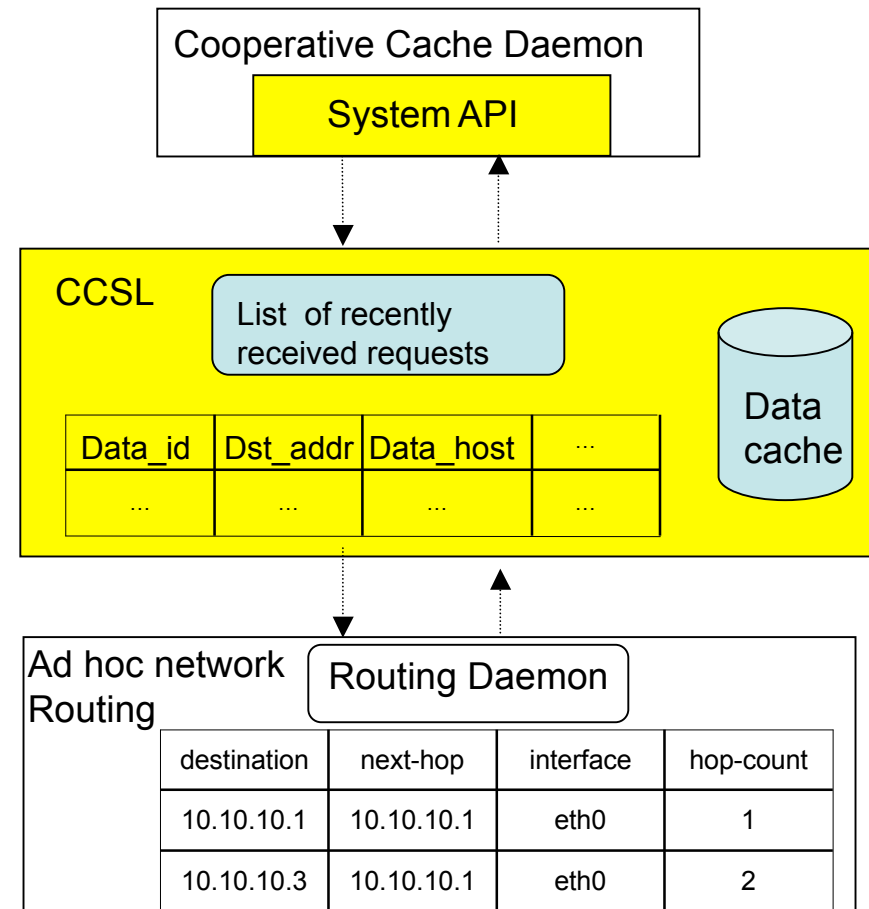
- Cooperative Cache Supporting Library (CCSL)

- Functions

- Check passing by packets
 - Make data access record
 - Cache popular data
 - Invalidate cached data
 - Retrieve routing information

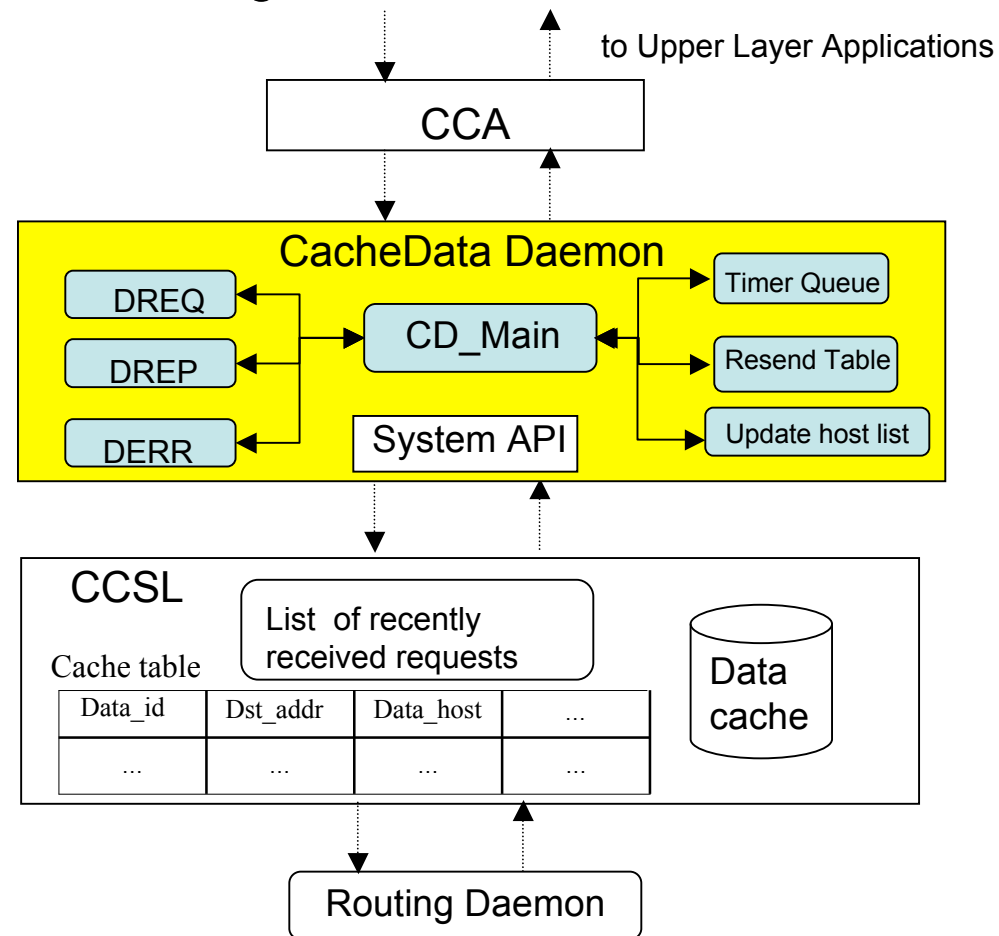
- API

- Cache operation (insert, search, update, remove)
 - Data access record
 - Packet transmission



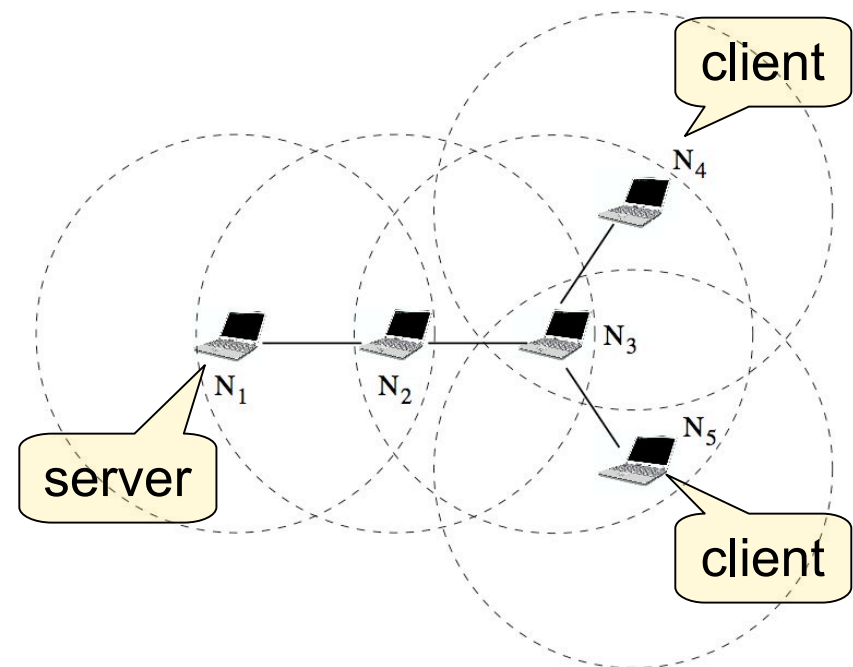
System Components

- Cooperative Cache Daemon (CCD)
 - main cache control logic



Implementation Setup

- Nodes:
 - Dell laptops 1.6 GHz CPU, 512 MB memory
 - Redhat Linux, kernel v2.4.5
 - Dell True-Mobile 802.11b wireless card (in ad-hoc mode)
- Experiment testbed
 - N_1 stores 100 test files of sizes 0.9KB, 1.3KB, 3.2KB and 6.4KB
 - N_4 and N_5 randomly choose files to issue request



Experiment Results

- Data access delay (in milliseconds)

	0.9KB	1.3KB	1.9KB	3.2KB	6.4KB
SimpleCache	28.56	31.19	42.12	60.64	118.26
Symmetric	24.87	27.13	36.97	49.30	102.38
Asymmetric	22.56	24.21	32.36	46.09	93.70

- The small scale prototype precludes the thorough evaluation.
- Collect the real data, and measure the parameter from the prototype to tune the simulation testbed

Packet type	Packet processing time (ms)				
	0.9KB	1.3KB	1.9KB	3.2KB	6.4KB
Request	0.217	0.218	0.215	0.217	0.219
Reply	1.483	1.514	1.836	2.294	3.132

Packet processing delay at the cache layer

Simulation Setup

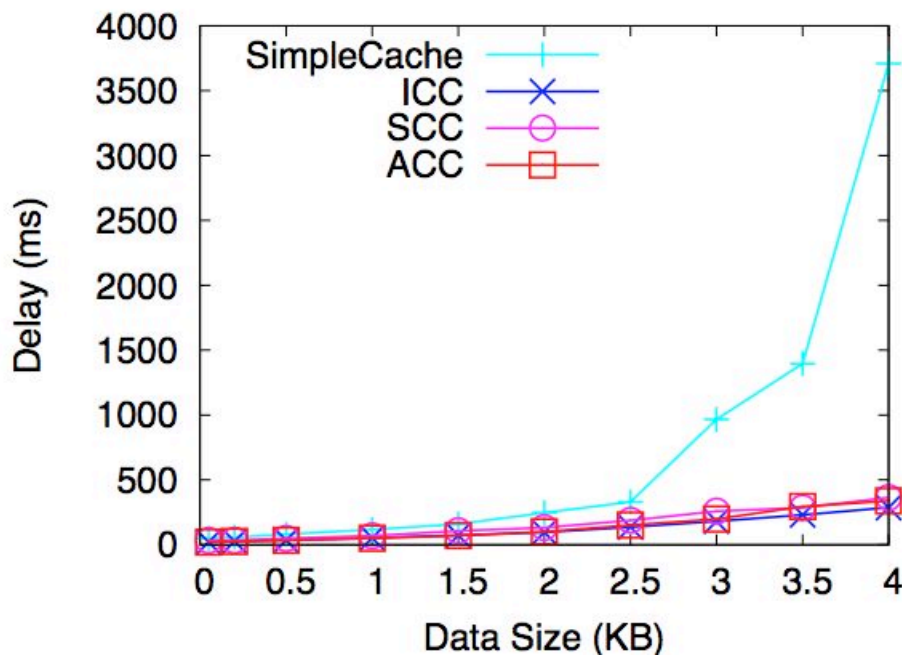
- Wireless Interface Setup

Wireless Interface	Channel Bandwidth	
single-interface single-channel	2 Mbps	5Mbps
multi-interface multi-channel	2M bps	5Mbps

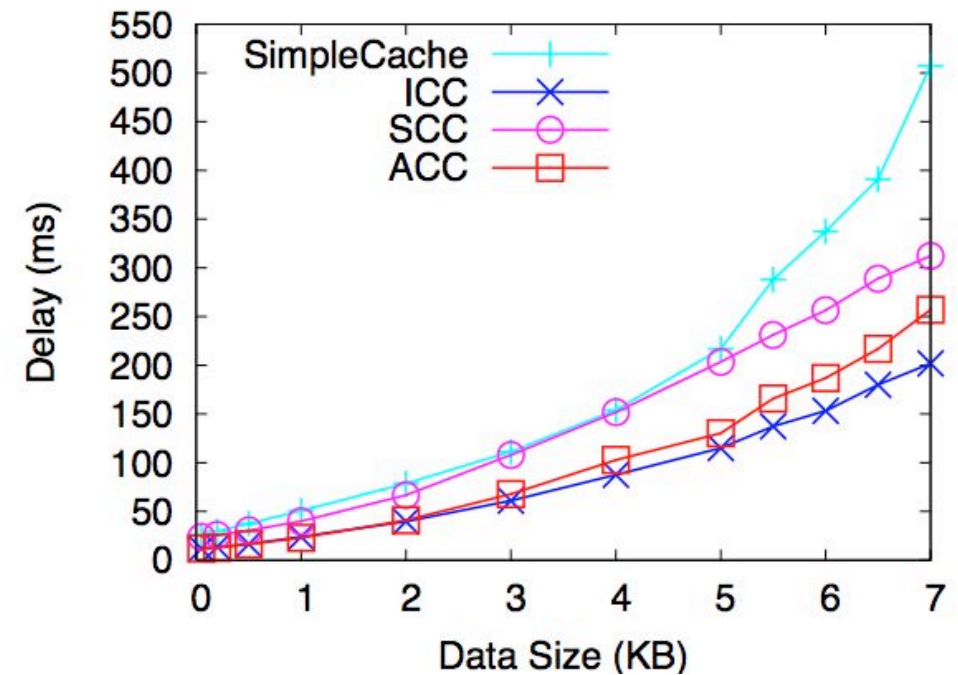
- Verify the simulation testbed by configuring it with the five-node experimental topology
- Increase the scale of the testbed
 - 100 nodes, 2 data servers.

Data Access Delay in Traditional 802.11 Networks

- Asymmetric approach effectively reduces the processing overhead at the cache layer.



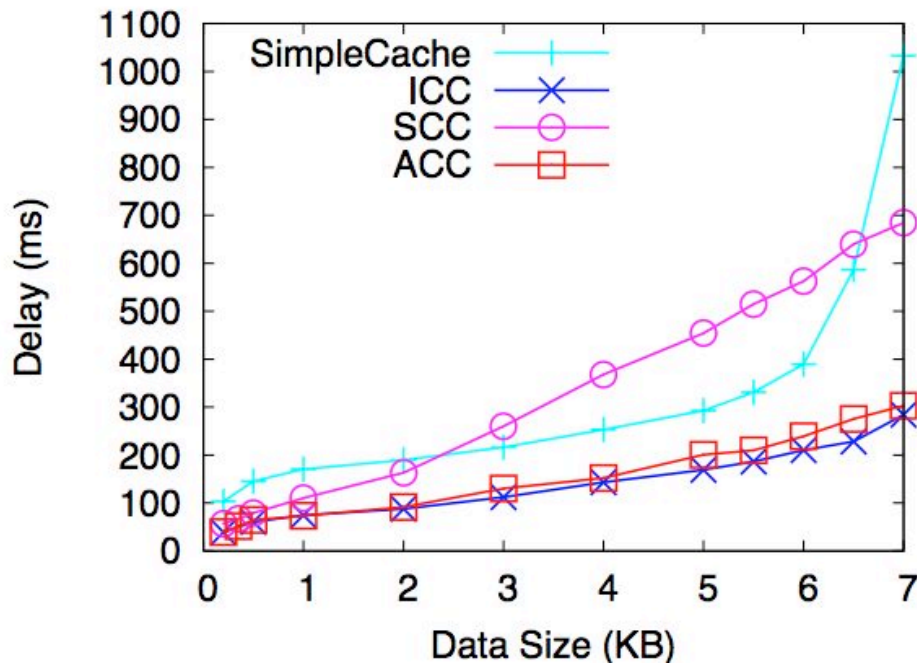
2M bandwidth



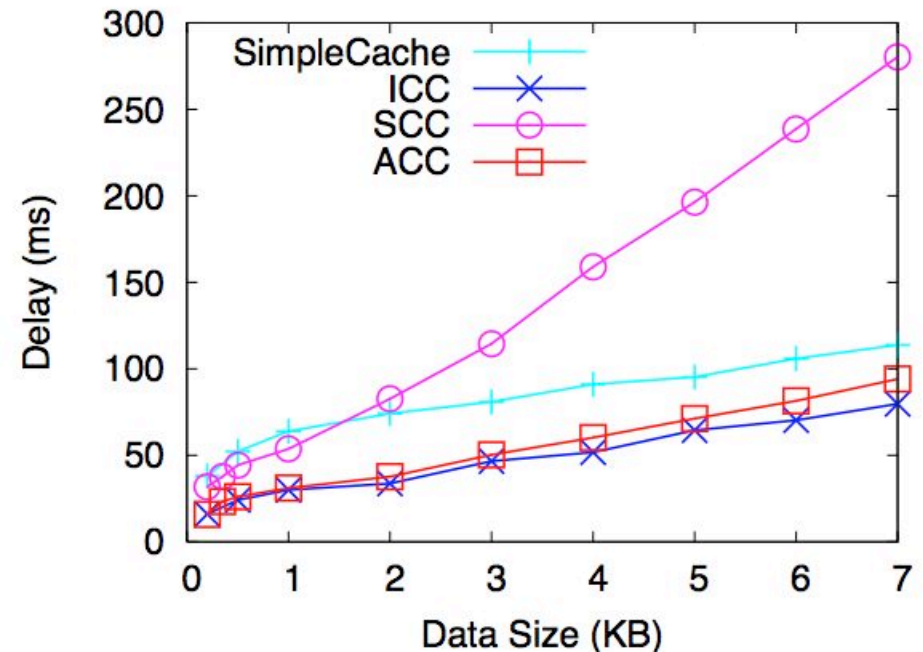
5M bandwidth

Data Access Delay in Multi-interface-multi-channel Mesh Networks

- Asymmetric approach enable data packet pipelining along the forwarding path.



2M bandwidth



5M bandwidth

Conclusions

- Designed and implemented cooperative cache in wireless P2P networks
- Designed asymmetric caching approach to reduce the cache layer overhead
- Studied the effects of data pipeline and MAC layer interference on cache management.



Mobile Computing &
Networking Laboratory



Thank You

Jing Zhao

jizhao@cse.psu.edu

<http://www.cse.psu.edu/~jizhao>

PENNSTATE

