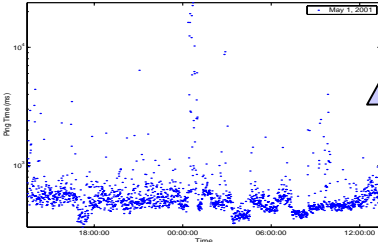


Enabling Efficient Content Location and Retrieval in Peer-to-Peer Systems by Exploiting Locality in Interests

Kunwadee Sripanidkulchai, Bruce Maggs, Hui Zhang, Carnegie Mellon University

Challenges posed by peer-to-peer

- Want scalable and high performance content location and peer selection. Existing solutions provide scalable location, but have not addressed peer selection.
- Retrieval performance between end-hosts is highly variable and dynamic.

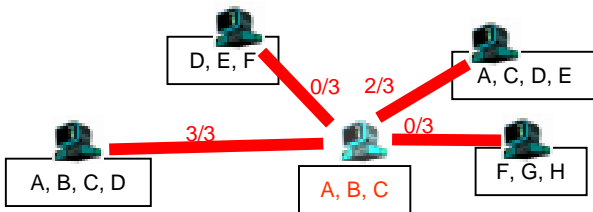


High variability ($\sigma \approx 1$ sec) in ping times over a 24-hour period to a random end-host on the Internet (typical for 1/3 of 2400 end-hosts pinged in our experiments)

- Need to use up-to-date performance state to select a peer
- For scalability, **cannot** maintain up-to-date state for all peers
- **Which** peers should we maintain state for?
 - **Peers that have locality in interests**

Locality in interests

Observation: people share common interests. Can we exploit this to improve content location and retrieval?



Proposed solution

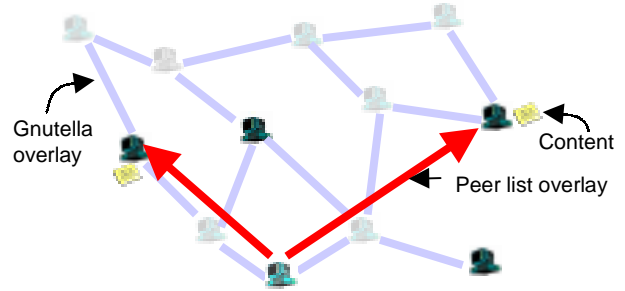
A distributed algorithm for peers to self-organize into clusters based on interests (peer list)

Why is it easier to incorporate dynamic performance state when using locality in interests to locate and retrieve content?

- Only need to keep performance state for peers that are likely to provide the content one is looking for.

Peer list

- 1) Each peer maintains a list of peers who share the **same interests**
- 2) Peer lists are initially bootstrapped using existing protocols, such as Gnutella, Tapestry, Pastry, CAN, or Chord. We use the following **heuristic**: peers that have the content you are looking for have the same interests.



- 3) The list evolves as more content is retrieved and more peers are discovered

Content location

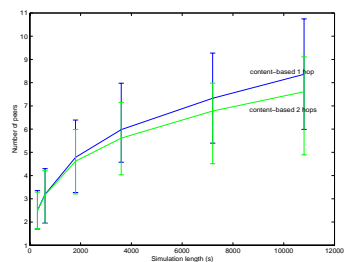
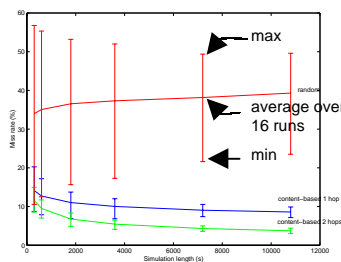
Queries for content are sent to peers in the list

Peer selection and content retrieval

Fine-grained dynamic performance state can be maintained for peers on the list

Potential benefits and overhead

- Use Boeing corporate web proxy traces to drive the request stream for the simulations
- Treat a request for a new document (a compulsory cache miss for a web cache) as a publish in peer-to-peer system
- Ran simulations over a period of 5 minutes to 3 hours
- Content location algorithms are based on
 - Asking random peers
 - Asking peers with same interests (1 hop)
 - Asking peers of peers with same interests (2 hops)



Locating content amongst peers with locality in interests results in low miss rates

Maintaining a small list of peers who share the same interests provides good hit rate

Implementation status

- Refining algorithm by ranking peers in one's list to select peers that are more likely to have content
- Exploring alternative mechanisms to bootstrap peer lists
- Developing techniques for incorporating dynamic performance state into algorithm
- Implementing our solution using Gnutella to bootstrap peer lists