Adaptive Web Caching

Sally Floyd
Lawrence Berkeley National Laboratory
http://www-nrg.ee.lbl.gov/floyd/

Joint work with Lixia Zhang (UCLA)
and Van Jacobson (LBNL)

Cache Workshop '97
June 10, 1997

Overview of talk:

- The vision - a foundation for a global data dissemination architecture.

- Two main components:

  - The underlying communication paths between neighboring web caches.

  - The flow of requests for data along these paths.

- Important but orthogonal issues: caching decisions by individual caches, data integrity as a property of the data, unique names for data.

- Is there a path for incremental deployment from reality to vision?

The vision:

- Popular pages `spread out' from the origin server to neighboring caches, following demand-driven diffusion.

- New caches are easily added to the caching infrastructure in a self-configuring manner.

- Data integrity is a property of the data itself, and does not rely on authentication of the caches themselves.

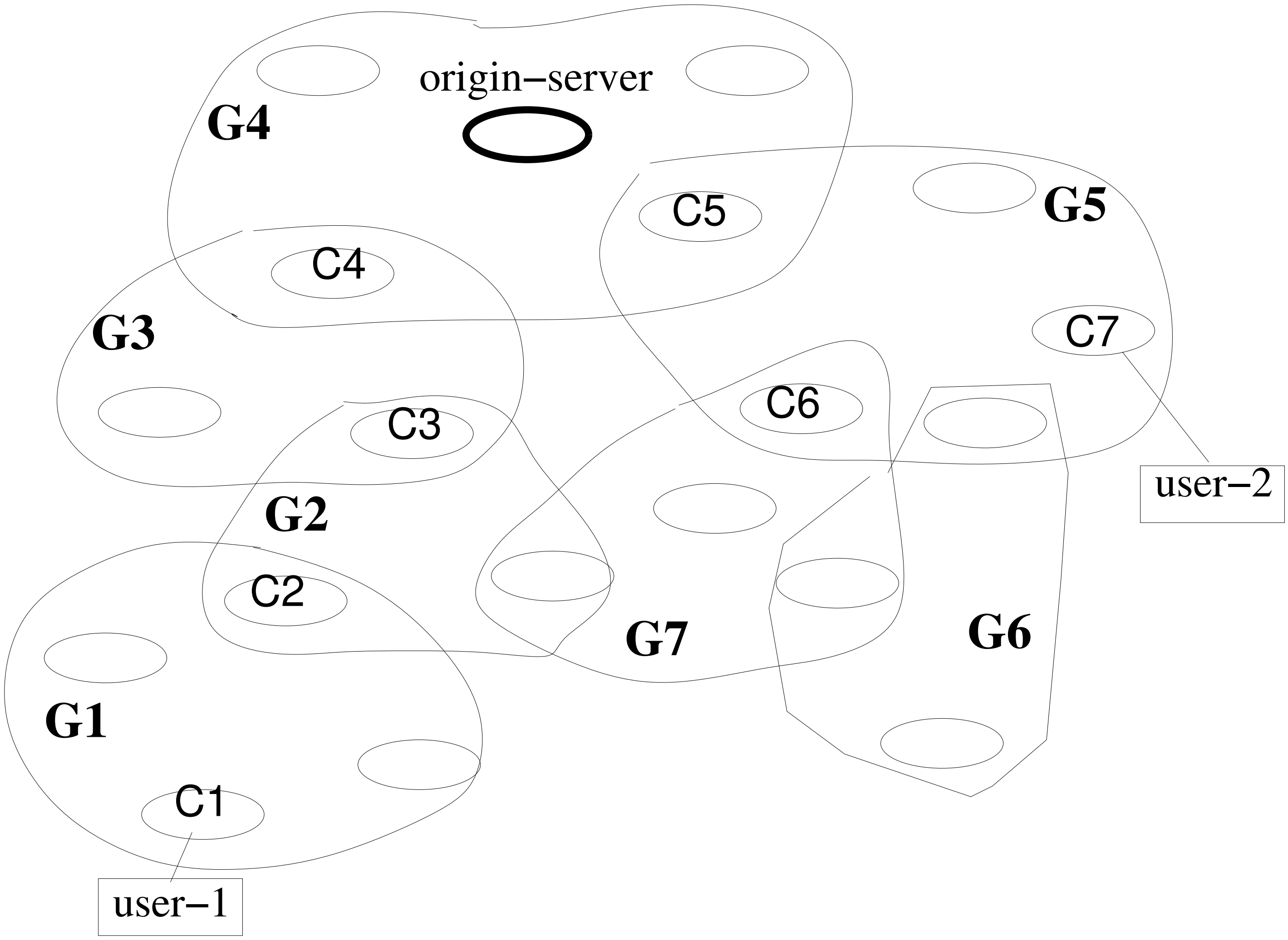
Self-organization of the data flow:

- Individual web caches are autonomous agents.
- The forwarding of requests for data among the caches is based on local information.
- The global behavior that results
  - avoids manual configuration,
  - is adaptive to changes in topology, load, etc.,
  - routes requests to the nearest cache likely to have the data.

Forwarding URLs: metrics for evaluating neighboring caches or cache groups:

- Distance in hops from the origin server.
- Distance in Cache Group hops from the origin server.
- In the general direction of the origin server (using information from the routing tables of the nearby router?).
- Past success rate/declared willingness in resolving requests of this type (e.g., replicated servers, the Japan cache in the U.S.).
- Past success rate/declared willingness in resolving requests in general (e.g., the large regional cache in the other direction from the transoceanic link).
- PICS (Platform for Internet Content Selection) as a tool in selecting the metric?

Overlapping multicast groups of web caches:



Forwarding URLs: the multicast scenario

- URLs are forwarded to members of the multicast group.
- If a hit, members use a randomized algorithm to inform the group.
- If no one reports a hit, members of the cache group have to each decide if they are a good candidate for forwarding the request to the origin server or to a neighboring cache group.
- Members use a randomized algorithm to multicast to the group their willingness to forward the request.
- If no one volunteers, the responsible cache forwards the request to the origin server.

Answering the request:

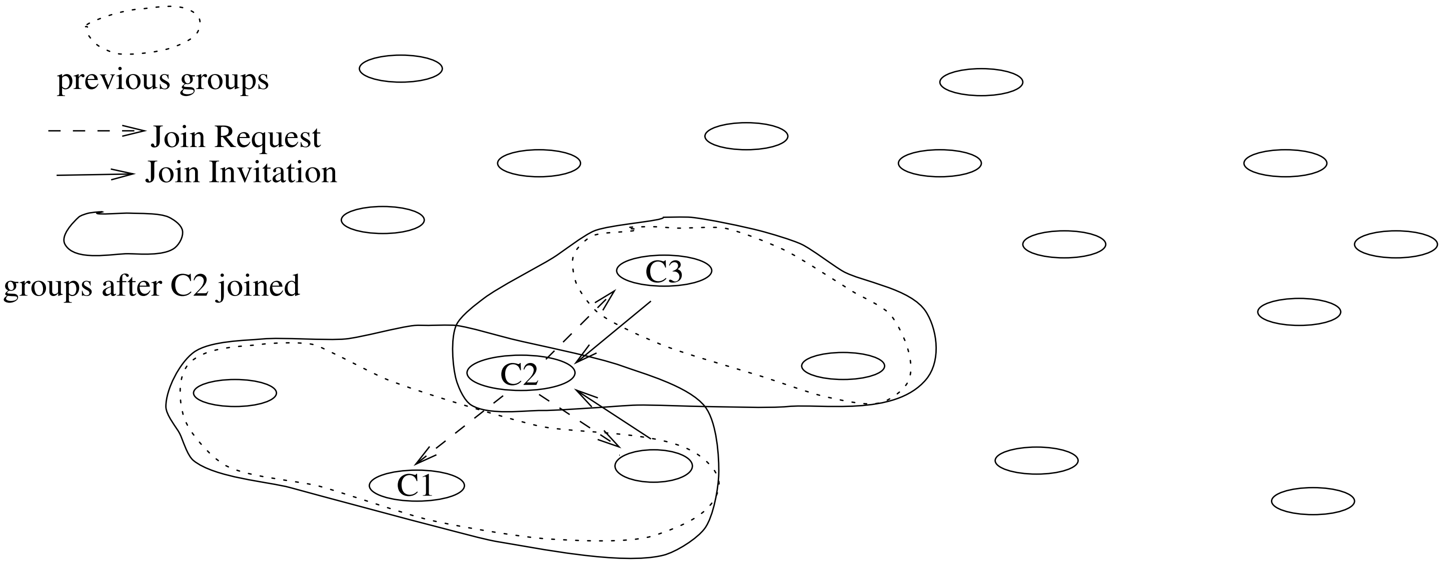
- Cache with a hit multicasts the data to the multicast group.

- Caches in the multicast group independently decide whether to cache the data.

- For the next steps, the cache that forwarded the request relays it by unicast to the requesting cache.

- Thus, for each request answered, the data gets dissiminated to one additional multicast group of caches along the path.

Self-organization of web caches into multicast groups:



Incremental deployment:

- Start with new algorithms for forwarding requests in the unicast-based cache hierarchy.

Forwarding URLs: the unicast scenario

- The local cache asks neighboring caches if they have the data.

- If no neighboring cache has the data, the local cache has to determine which of the neighboring caches is the best candidate for forwarding the request (either to the origin server or to another set of neighboring caches).

- If none of the neighboring caches are good candidates, then the responsible cache forwards the request to the origin server.

Protecting against rogue/broken caches:

- Learn from history which caches in the multicast group are reliable forwarders of requests.
- Learn from history which caches present invalid/unresolvable requests.
- Load-balancing when one cache is flooding the cache group with valid requests.

Orthogonal issues:

- Individual caches deciding whether or not to cache data.

- Individual caches deciding whether simply to forward the request to the origin server.

End-to-end issues:

- Data integrity.

- Access control.