Last Class: Consistency Models

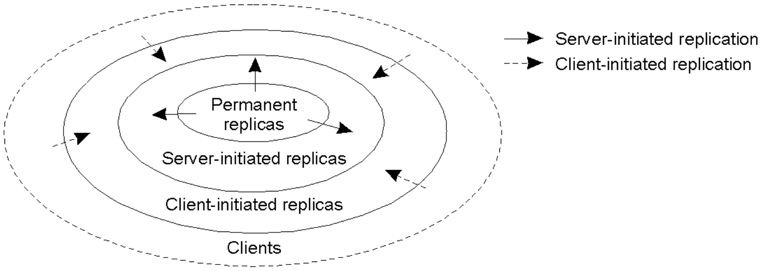
- Need for replication
- Data-centric consistency
  - Strict, linearizable, sequential, causal, FIFO

Today: Implementation Issues

- Replica placement
- Use web caching as an illustrative example
- Distribution protocols
  - Invalidate versus updates
  - Push versus Pull
  - Cooperation between replicas

Replica Placement

- Permanent replicas (mirroring)
- Server-initiated replicas (push caching)
- Client-initiated replicas (pull/client caching)



Server-initiated replication

Client-initiated replication

Permanent replicas

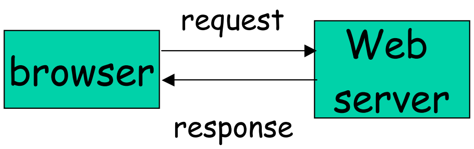
Server-initiated replicas

Client-initiated replicas

Clients

Web Caching

- Example of the web to illustrate caching and replication issues
  - Simpler model: clients are read-only, only server updates data

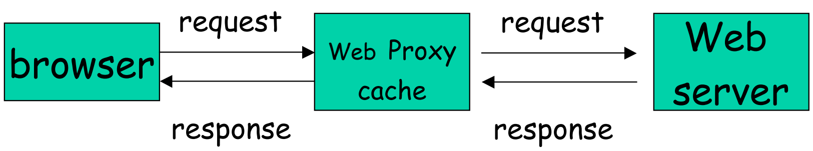


browser

request

response

Web server



browser

request

response

Web Proxy cache

request

response

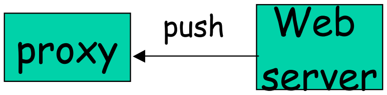
Web server

Consistency Issues

- Web pages tend to be updated over time
  - Some objects are static, others are dynamic
  - Different update frequencies (few minutes to few weeks)
- How can a proxy cache maintain consistency of cached data?
  - Send invalidate or update
  - Push versus pull

Push-based Approach

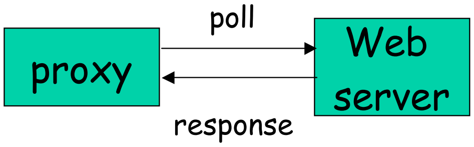
- Server tracks all proxies that have requested objects
- If a web page is modified, notify each proxy
- Notification types
  - Indicate object has changed [invalidate]
  - Send new version of object [update]
- How to decide between invalidate and updates?
  - Pros and cons?
  - One approach: send updates for more frequent objects, invalidate for rest



Push-based Approaches

- Advantages
  - Provide tight consistency [minimal stale data]
  - Proxies can be passive
- Disadvantages
  - Need to maintain state at the server
    - Recall that HTTP is stateless
    - Need mechanisms beyond HTTP
  - State may need to be maintained indefinitely
    - Not resilient to server crashes

Pull-based Approaches



- Proxy is entirely responsible for maintaining consistency
- Proxy periodically polls the server to see if object has changed
  - Use if-modified-since HTTP messages
- Key question: when should a proxy poll?
  - Server-assigned *Time-to-Live (TTL)* values
    - No guarantee if the object will change in the interim

Pull-based Approach: Intelligent Polling

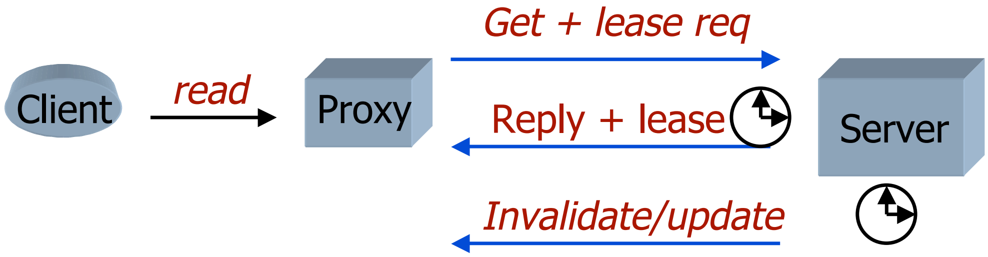
- Proxy can dynamically determine the refresh interval
  - Compute based on past observations
    - Start with a conservative refresh interval
    - Increase interval if object has not changed between two successive polls
    - Decrease interval if object is updated between two polls
    - Adaptive: No prior knowledge of object characteristics needed

Pull-based Approach

- Advantages
  - Implementation using HTTP (If-modified-Since)
  - Server remains stateless
  - Resilient to both server and proxy failures
- Disadvantages
  - Weaker consistency guarantees (objects can change between two polls and proxy will contain stale data until next poll)
    - Strong consistency only if poll before every HTTP response
  - More sophisticated proxies required
  - High message overhead

A Hybrid Approach: Leases

- Lease: duration of time for which server agrees to notify proxy of modification
- Issue lease on first request, send notification until expiry
  - Need to renew lease upon expiry
- Smooth tradeoff between state and messages exchanged
  - Zero duration => polling, Infinite leases => server-push
- Efficiency depends on the *lease duration*



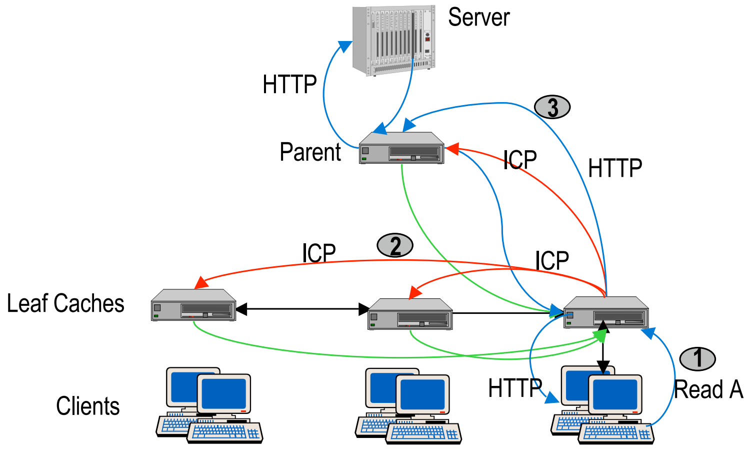
Policies for Leases Duration

- Age-based lease
  - Based on bi-modal nature of object lifetimes
  - Larger the expected lifetime longer the lease
- Renewal-frequency based
  - Based on skewed popularity
  - Proxy at which objects is popular gets longer lease
- Server load based
  - Based on adaptively controlling the state space
  - Shorter leases during heavy load

Cooperative Caching

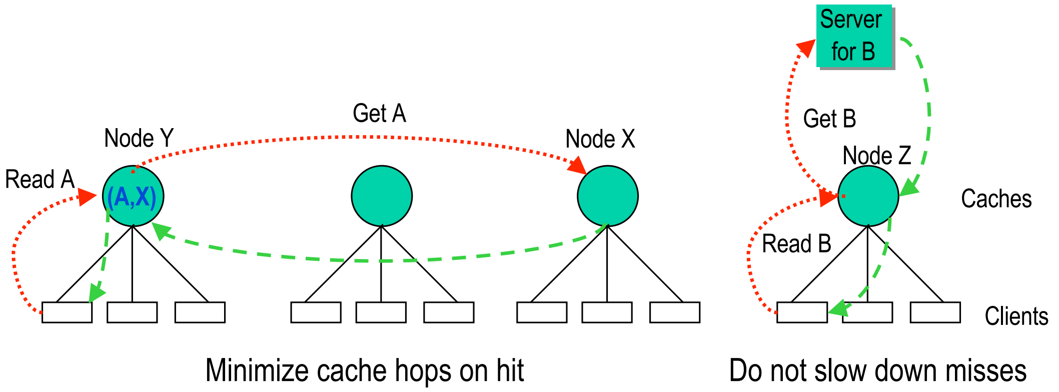
- Caching infrastructure can have multiple web proxies
  - Proxies can be arranged in a hierarchy or other structures
    - Overlay network of proxies: content distribution network
  - Proxies can cooperate with one another
    - Answer client requests
    - Propagate server notifications

Hierarchical Proxy Caching



Examples: Squid, Harvest

Locating and Accessing Data



Properties

- Lookup is local
- Hit at most 2 hops
- Miss at most 2 hops (1 extra on wrong hint)

CDN Issues

- Which proxy answers a client request?
  - Ideally the "closest" proxy
  - Akamai uses a DNS-based approach
- Propagating notifications
  - Can use multicast or application level multicast to reduce overheads (in push-based approaches)
- Active area of research
  - Numerous research papers available