DECADE Requirements

draft-gu-decade-reqs-05

Yingjie Gu, **David A. Bryan,** Y. Richard Yang, Richard Alimi

IETF-78 Maastricht, DECADE Session

Changes Since -04

- Author list changed
- Added one new major requirement (cross-platform access)
- Clarification on storage management
  - May be a logical function, resides in the service provider space/provided by provider
- Clarification on use of optimizations for authorization
  - May use optimizations to prevent checking credentials every call (i.e., authenticated connections) as long as permissions are preserved

Overview

- Draft presents the requirements, as well as the rationale behind them
- Focus is on chartered work, keeping scope as narrow as possible, while trying to allow for reuse
  - Avoid non-charter application requirements that will widen the scope of work in the group
- Broken down by category

General Principles

- Core data storage operations: read/write/delete
  - Explicit control over in-network storage (contrast to P2P caching)
  - Network element likely operated by service provider
- Low-latency access
  - P2P applications may have constraints on delivery time
- Efficient transfer among multiple storage servers
  - Data transfer between storage servers avoids last-mile upload
- Low management costs for providers
- Application-independent API
  - Allow many types of applications to take advantage (so long as doesn't increase complexity for base P2P use case)
- Client control over resource allocation
  - Bandwidth (e.g., rate/proportion/priority), storage quota, connections
- Allow for small object size
  - Some P2P apps deliver data in small chunks (e.g., 16KB)

Data Access

- User can read/write from own storage
  - May also allow negotiation of data transport protocol
- Define and enforce access control policies for remote peers
  - Note that remote peers may be in different admin/security domains
- Allow server-to-server transfers
  - Improve efficiency, data portability, maintenance reasons

Data Management

- Protocol agnostic with respect to storage service
  - Be able to offer different storage service levels (e.g., multiple copies, longevity) using same protocol/API
- User can get current resource usage and limits (including list of stored objects)
  - Make local resource allocation decisions; application restarts
- Current proposal: Simple set of operations
  - Write model: allow append, but no update of existing data. Single writer for an object
  - Delete model: explicit delete, or TTL based
  - Read model: Multiple readers, read before completely written, parallel or pipe-lined read

Simple Operations: Rationale

- Major considerations
  - Semantics under multiple writers and read/write conflicts vastly increases complexity
  - Updating data in-place leads to consistency issues
  - Erred on side of design that is simple but perhaps slower
- Current requirements
  - Allow multiple, concurrent readers
    - P2P client uploads to multiple peers concurrently
  - Allow readers to access data before fully-written
    - Avoid store-and-forward delays to reduce latency, in particular for large object
  - Avoid update operation for already-written data (immutability)
    - However, allow appending
- Possibly could have performance optimization through relaxing consistency requirements
- WG needs to consider and discuss these carefully. Looking for input.

Resource Control

- Allow user to define resource control policies between concurrently-running applications
  - Apps may be on different machines, or may not directly communicate
- Allow per-peer, per-data resource control
  - e.g., per-peer BW control or certain blocks with higher priority
- For discussion in WG
  - Requirements on mechanism to define resource control for remote peers' requests
    - Decision has impact on latency and load on server

Authorization

- Per-peer (user), per-data read access
  - Authorize particular peers to retrieve particular content
- Per-peer (user) write access
  - Authorize particular set of peers to store content
- For discussion in WG / Future work
  - Requirements on mechanism to define access control for remote peers' requests
    - Again, decision has impact on latency and load on server

Data Availability

- Allow (authorized) offline-access to user's storage
  - Handle intermittent connectivity, or when no app actively running

Error Conditions

- Indicate error if insufficient resources
  - Requested resources (e.g., storage) not available
- Indicate error if content unavailable or deleted
  - Provider may need reject, delete or quarantine data
  - DECADE does NOT indicate how such data identified
  - … but should not cause applications to break
- Allow server to reject requests/connections if overloaded
  - Server should not be forced to undertake new work if overloaded

Protocol Requirements

- Support for peers behind NATs/Firewalls
- No unsolicited inbound messages to clients
  - For simplicity, clients don't have to listen/receive requests
- Platform agnostic encoding
  - Information such as metadata should be stored in an OS and architecture independent format

Other Requirements

- Other requirements for discussion in WG
  - Data naming
  - Reliability/persistence
- Security requirements are essential
  - Not yet specified: some will be dependent on decision about architecture/approach taken, some are independent

Comments and questions?