flockfs, a moderated group authoring system for wireless workgroups

Surendar Chandra and Nathan Regola

- group authoring system for wireless users
- In lieu of maintaining single version, maintain multiple copies: each reconciled using moderation
Group authoring systems

Group members modify a shared document

- **Synchronous**: NFS, AFS, Google Docs, SubEthaEdit ..
- **Asynchronous**: Coda, Bayou, Apple iDisk, Windows Live (Mesh, SkyDrive, Sync) ..

**Centralized**
- good availability and control
- maintain cache consistency

**Distributed**
- ease of deployment
- propagate and reconcile updates

**Goal**: develop system for contemporary users
Contemporary users are wireless

WiFi users @ND (13.8K of 27.7K)

ubiquitous - ~1,300 AP
First, evaluate prior systems

Evaluation requires durations:

- when members can participate in group communications
  - e.g., long durations between sessions affect asynchronous propagation
  - collected empirical wireless availability traces at Notre Dame

- when groups modify shared contents
  - e.g., long simultaneous modification affects exclusive modification schemes
  - few deployed systems. Hence, when online synthesized update sessions:
    - group sizes: 5, 10, 20, 30
    - every so often (1, 2, 3 or 4), start randomly, last for up to 0.5, 1 or 2 hrs
critiques to random group, sessions..

- Group members are not random: “students schedule meetings: high simultaneous availability”
  - Poor system behavior when users are simultaneously available
  - Future work: empirical measurements from flockfs

- How about decompose document into sections that are modified by each group member
  - Consider complex documents where changes are global
Wireless availability traces

- Zeroconf _workstation records: Mac and Linux
- configured campus wireless to route packets to monitored VLAN
- 12/3/07-8/25/08, 2.7k users
- diurnal and week day variance
- session: median < 20 min, 95% < 75 min
- between session: median < 1.4 hr, 29% > 10 hr
- node churn throughout

more details in paper
Behavior of prior systems

1. centralized: exclusively lock object during entire session, others read prior document version
   * conflicting attempts to modify object: delay or fail

2. centralized: *last writer wins*
   * optimistic, allows concurrent updates, long session preempt
   * sessions that are preempted conflict

3. peer-to-peer: pair-wise anti-entropy - like bayou
   * out of order update delivery
   * roll-forwards and roll-backs
Analysis of prior systems

- Prior schemes attempt to maintain single version
- Poor performance when many users are available
  - Pessimistic schemes: consistent but poor performance
    - Allows few modifications
    - Users need to serialize
  - Optimistic schemes: inconsistent views

*Details in paper*
Design philosophy

- wireless availability characteristics requires maintaining multiple versions
  - need to reconcile changes among multiple versions - moderation

* Using reconciliation to share files between occasionally connected computers. John H. Howard. Workshop on Workstation Operating Systems '93
Flockfs: moderated collaboration system

- maintain ‘n’ authoritative copies
  - each group member exclusively modifies their version
  - each group member hoards read-only copies of others contents
  - author incorporates updates from others using moderation

R - read only
RW - read/write
**Abstract**: …. exhibits acceptable file system performance and update propagation latency...

**Experiments**: …. Iozone read performance within 5% of FUSE performance...

**Abstract**: …. Place holder. Alice is responsible for this ...

---

knol.google.com
Moderation

- automated moderation more complex than pairwise automated reconciliation (future work)
- log provenance to assist in convergence
  - unique version for each update session
  - log operations on read-only replica along with author copy
    - assumption: open() == changes incorporated into author version
  - others can query log to know if changes incorporated
  - more later


* Flexible and safe resolution of file conflicts. Puneet Kumar, M. Satyanarayanan. USENIX ‘95
Flockfs interface

- file system interface: application agnostic

% mkdir project1
% mkdir project1/sam/

~/flockfs/

project1

surendar    me    sam

- group membership not maintained by system
- not all users hoard others' contents
- hoarding helps overall system

instruct flockfs to hoard contents from sam
**publish** “Comment”

- ends an update session. Comment is available to others

**status** prints comments and provenance records

- file.c
  - Comment: Fixed the buffer overflow
  - Opened: sam:common.h Sat Apr 10 21:24:15 EDT 2009
  - Opened: bob: object.c Sat Apr 18 17:31:00 EDT 2009
### system architecture

**FUSE userspace file system**

**epidemic based middleware**
* P2P approach competitive
* pull with frequency adaptive to session length

**Git based file maintenance**
* Reuse optimized software, good compression

---

Yenta: *An asynchronous group communication middleware for wireless users*, MSWIM 2009
### IOzone file system benchmark

- similar to fuse filesystem on Mac

### asynchronous update propagation

<table>
<thead>
<tr>
<th>Operation</th>
<th>powerbook laptop (MB/s)</th>
<th>iMac desktop (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>native</td>
<td>fuse</td>
</tr>
<tr>
<td>write</td>
<td>22.74</td>
<td>16.65</td>
</tr>
<tr>
<td>read</td>
<td>22.46</td>
<td>16.12</td>
</tr>
</tbody>
</table>
Analysis of wireless user availability shows limitation of prior collaboration systems.

Propose to maintain ‘n’ definitive copies with provenance logging. Manual convergence.

Fuse and git helps build quick prototype.

Acceptable file system performance.

Future work: user studies.