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

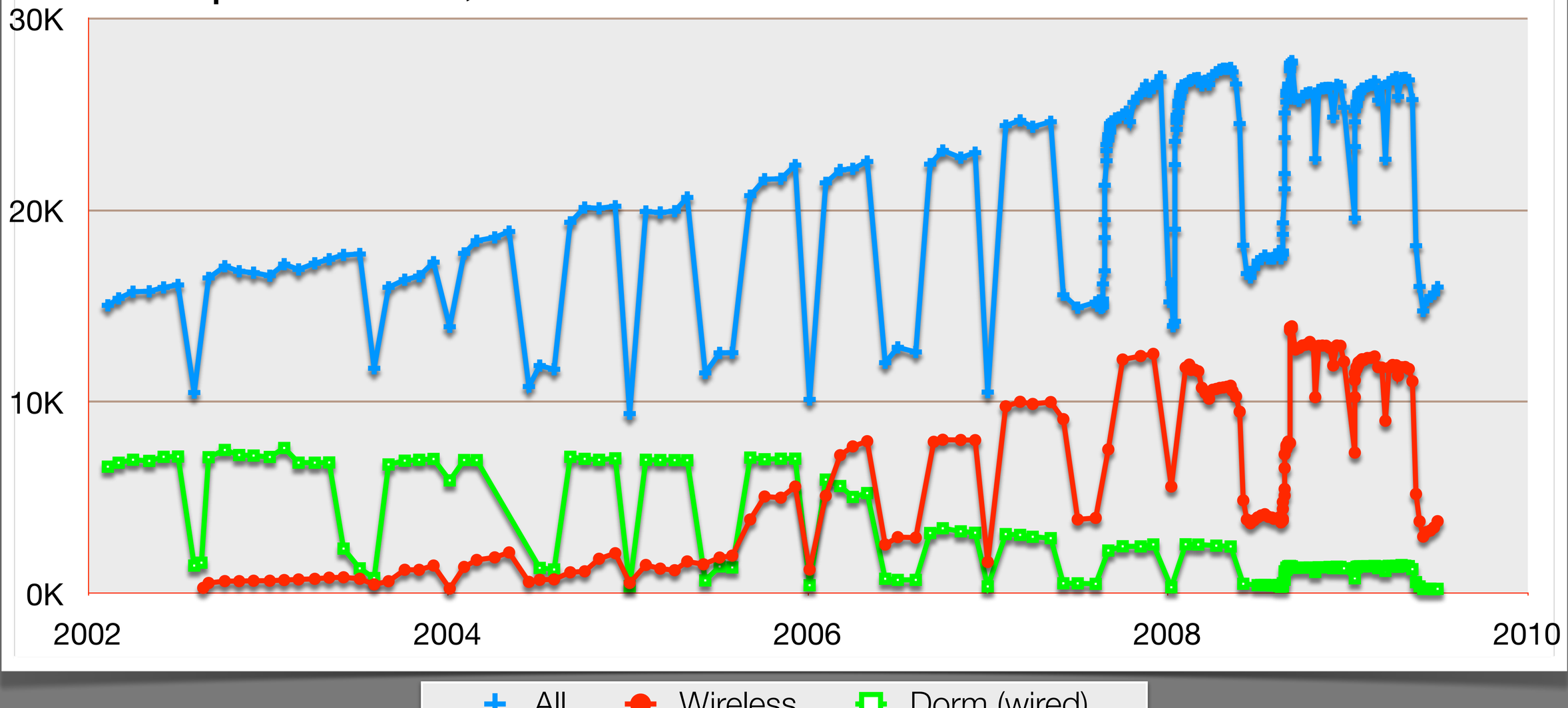
Distributed

- ease of deployment
- propagate and reconcile updates

- Goal: develop system for contemporary users

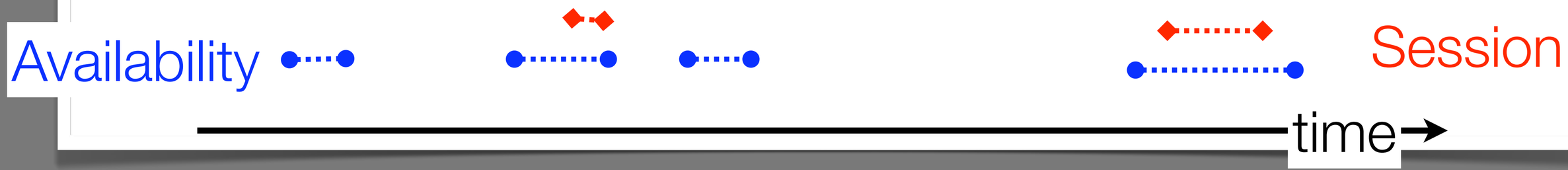
Contemporary users are wireless

- WiFi users @ND (13.8K of 27.7K)
  - ubiquitious - ~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 mi
- 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

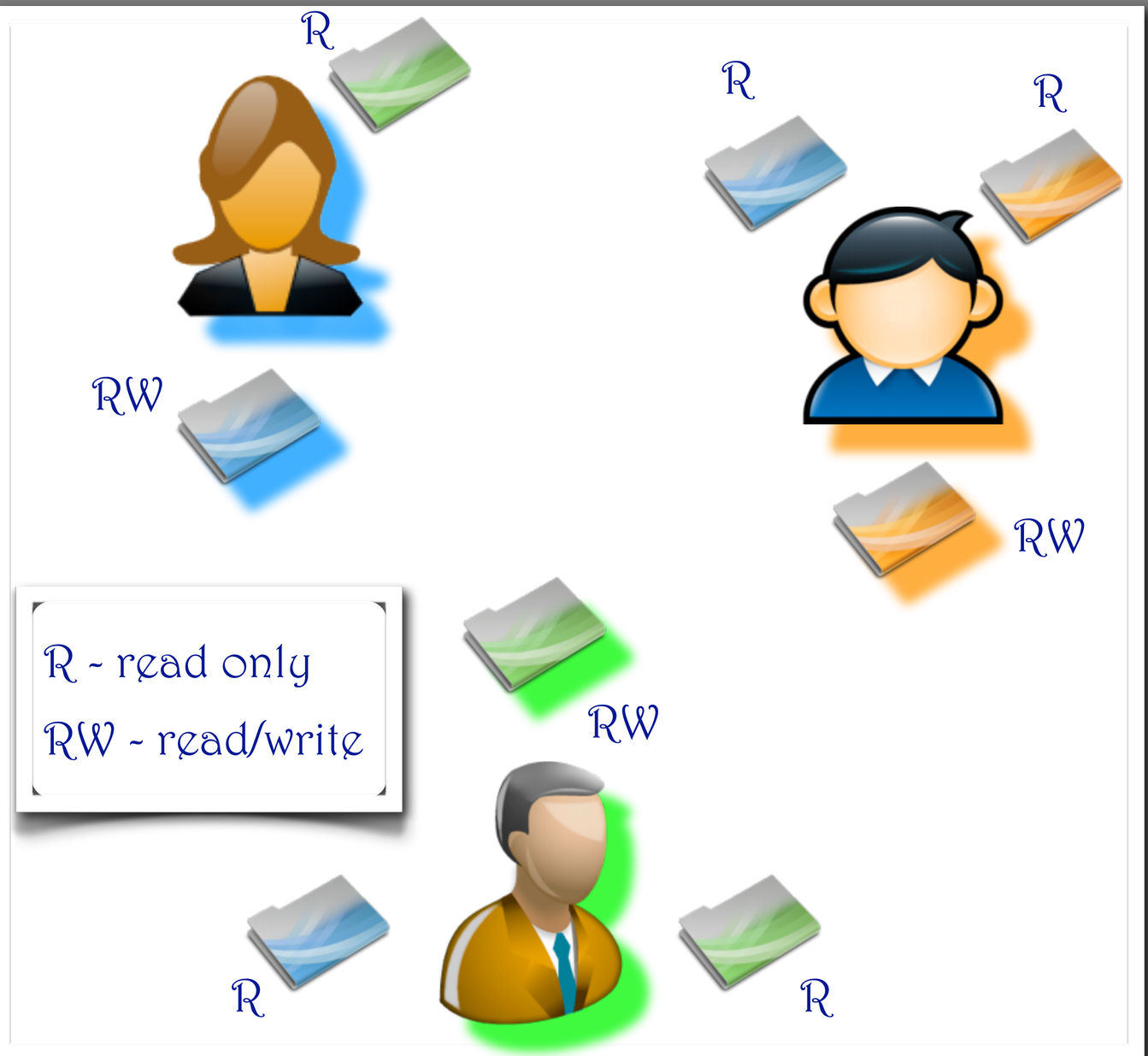
Design philosophy

* 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



Moderation

* manual reconciliation of updates from (n-1) others

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

Abstract: …. ??? …

Moderation

- automated moderation more complex than pair-wise 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

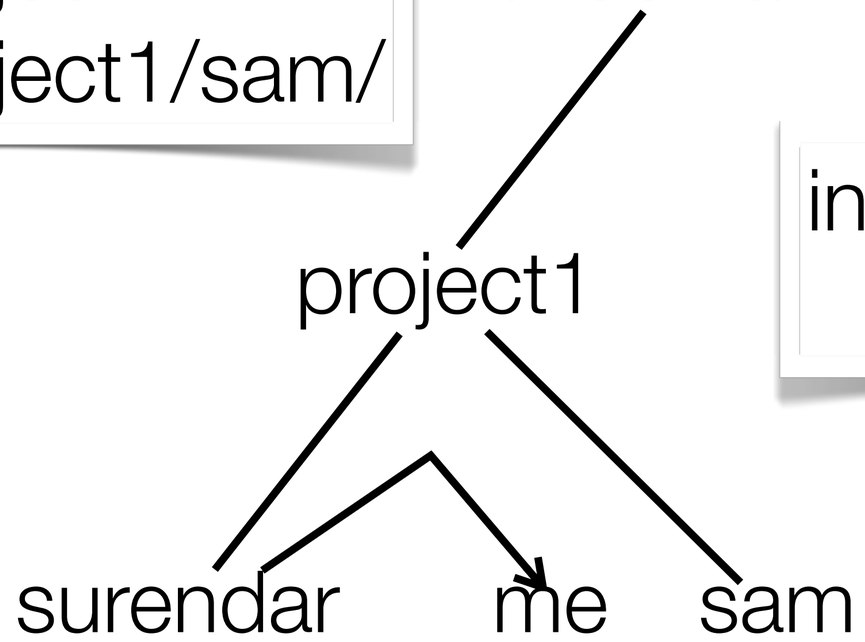
* *Resolving File Conflicts in the {Ficus} File System*. Peter Reiher, John S. Heidemann, David Ratner, Gregory Skinner and Gerald J. Popek. USENIX ‘94

* *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/
```




```
instruct flockfs to hoard
contents from sam
```

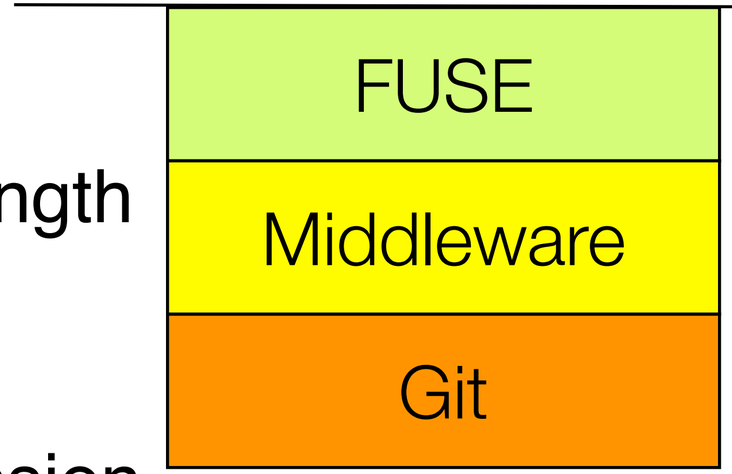
- group membership not maintained by system
- not all users hoard others contents
  - hoarding helps overall system

system support programs

* *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

performance

* IOzone file system benchmark
  * similar to fuse filesystem on Mac
* asynchronous update propagation

Operation	powerbook laptop (MB/s)			iMac desktop (MB/s)		
	native	fuse	flockfs	native	fuse	flockfs
write	22.74	16.65	16.28	57.63	41.35	40.27
read	22.46	16.12	15.67	55.06	52.31	56.30

summary

- 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
  - Available: http://flockfs.sourceforge.net/