

Tra, a file system synchronizer

Russ Cox

William Josephson

rsc@mit.edu

wkj@eecs.harvard.edu

November 26, 2001

PDOS Group Meeting

Outline

Problem

Solution (vector time)

Glimpse of algorithm

Building a good tool

The Problem

Want to use lots of computer systems (mostly) interchangeably.

Edit anything anywhere, have changes propagate properly.

For now, think “same home directory everywhere.”

Non-solutions

Avoid the problem: use one machine.

- remote login
- network file systems

Assumes connected operation.

File systems or tools for disconnected operation.

- AFS
- Coda
- CVS

Assumes central server (perhaps not possible; one more machine to admin).

What makes a solution?

Correct propagation of updates, creates, and deletes.

- never lose an update

Asymmetric synchronization.

- cvs update vs. cvs checkin

Relaxed communication requirements.

- perhaps some pairs of machines never talk

Partial replicas.

- maybe I don't want *all* of frenulum's /usr.

Portability.

- Windows. Unix. Plan 9. Macintosh? (User-level.)

Simplicity.

- this thing controls your files.
- obviously no bugs; not no obvious bugs

Other synchronizers

Rsync

- too much work left to the user
- bad packaging of a good file transfer algorithm
- knows *how* to copy; doesn't know *what* to copy

Ficus, Rumor

- almost perfect, doesn't run anywhere

Unison

- only works for a pair of hosts

Discuss these more after we know about Tra.

Why is this hard?

sync A's F to B change F on A	sync A's F to B change F on B	sync A's F to B change F on A change F on B
sync from B to A: <i>nop</i>	sync from B to A: <i>copy B's F to A</i>	sync either way: <i>report conflict</i>

sync A's F to B remove F on A sync from A to B: <i>remove F on B</i>	sync A's F to B remove F on A sync from A to B: <i>remove F on B</i> create new F on B sync from B to A: <i>copy B's F to A</i>	sync A's F to B remove F on A remove F on B create new F on B sync from B to A: <i>???</i>
---	---	---

One-writer synchronization

Suppose only A makes changes to file F .

Define F 's *modification time* to be the time on A that F was last changed.

If we copy F 's modification time when we copy F , we can always tell which of two copies is newer.

Suppose there are lots of files.

If we compare modification times on every file, we'll get correct results and be very slow.

On system X , store $t_X =$ “when our copy of the file system existed on A .”

$B \rightarrow C$: I know about A as of time t_B .

$C \rightarrow B$: I know about A as of time t_C .

$C \rightarrow B$: Here are all the files I have that you don't know about.

B incorporates new files, sets $t_B = \max(t_B, t_C)$.

t_X is a *synchronization time*.

Vector time

From theoretical distributed systems.

An array specifying local time on a collection of systems.

Modification time of $(A:5 \ B:100)$ means last change on A was at A -time 5, last change on B was at B -time 100.

Only partially ordered:

- $(A:5 \ B:100) \leq (A:6 \ B:102)$
- $(A:5 \ B:100) \leq (A:5 \ B:102)$
- $(A:5 \ B:100) \parallel (A:6 \ B:99)$

General synchronization

Replace scalar time with vector time in the one-writer algorithm and everything works out.

Incomparable times mark conflicts.

$B \rightarrow C$: I know about A as of time t_B .

$C \rightarrow B$: I know about A as of time t_C .

$C \rightarrow B$: Here are all the files I have that you don't know about.

B incorporates new files, sets $t_B = \max(t_B, t_C)$.

To handle partial replicas, use per-file sync time instead of per-replica.

Algorithm

Five states for a file:

File	path is an extant plain file (non-directory)
Dir	path is an extant directory
Ghost	path is a record of a ghost
Unknown	there is no record whatsoever of path
NotHere	this replica is configured not to store path

Twenty-five cases for each (from-state, to-state) pair.

We'll go through two.

The (File, File) decision

```
case (File, File):  
  if from.mtime(path) ≤ to.synctime(path)  
    // to knows about from's version  
    return complete  
  else if to.mtime(path) ≤ from.synctime(path)  
    // from knows about to's version: safe to copy  
    copy path  
    to.mtime(path) = from.mtime(path)  
    to.synctime(path) max= from.synctime(path)  
    return complete  
  else  
    // to and from have incomparable versions  
    report update/update conflict  
    return incomplete
```

The (Dir, Dir) decision

```
case (Dir, Dir):  
  if from.mtime(path) ≤ to.synctime(path)  
    // to knows about from's version  
    return complete  
  else  
    // there are updates on from that we need to consider  
    status = complete  
    for each child in path on either replica  
      if sync(path/kid) == incomplete  
        status = incomplete  
    if status == complete  
      to.synctime(path) max= from.synctime(path)
```

Ficus and Rumor

Vector modification times (version vectors) but no sync times.

Only handles full replica syncs. (Ugly attempts to fix this in Rumor.)

Almost invented vector sync times.

Instead, they need distributed garbage collection to handle deletions.

Moral: ideas from file systems don't translate directly to user-level tools.

Unison

Proved correct, for some definition of correct.

Only considers pair of replicas.

copy A's F to B
change F on B
sync B's F to C
sync from B to A:
 copy B's F to A
sync from C to A:

Unison: *conflict!*

Tra: *nop*

Tool building

I use Tra every day. No one else uses it at all.

It's not usable unless you understand the algorithm.

Rewrite in progress addresses:

latency — add parallelism

bandwidth — SHA1 hashes to avoid dumb copies

undo, redo — encourage experimentation

ease of use — explanations must be understandable

```
/sys/src/cmd/tra/tra.c: update/update conflict
```

VS.

```
/sys/src/cmd/tra/tra.c: update/update conflict
```

```
Sun Nov 11 17:33:01 EST 2001
```

```
modified on lusitania by rsc
```

```
Mon Nov 12 09:12:31 EST 2001
```

```
modified on emelie by rob
```

Tool building, II

Interface

want simple, easy-to-intuit gui

not clear what dumb text version should look like

Partial Replicas

not clear how to specify them:

```

386
  +
mail
  lib
    *
usr
  rsc
    mp3
    tmp
    +

```

or

```

-*.o
/386
/mail/lib/*
-/usr/rsc/mp3
-/usr/rsc/tmp
/usr/rsc

```

or something else entirely?

Future work

Completeness

- sound: never incorrectly discards an update
 - complete: never raises a spurious conflict
- obviously sound, not obviously complete

Software distributions

Replace sup?

The next Plan 9 release will use Tra in some form.

Get users

Has to be easy to configure, easy to use

Release some time in January?