
monotone

- Monotone is:
  - A version control system
  - in C++
  - ~20,000 executable LOC
  - fully supported on Windows, OS X, Unix
  - fully internationalized
- Monotone has:
  - Atomic commits
  - Rename support (including directories)
  - Full merge support

What makes using a VCS frustrating?

What makes using a VCS frustrating?

- Unpredictability
- Getting bossed around by the system
- Data loss

What makes using a VCS frustrating?

- Unpredictability
- Getting bossed around by the system
- Data loss
- Not knowing what's going on around you
- Getting blocked on someone else's build breakage
- Not being able to find things

So we want...

- Understandable model of how the VCS views the world
- Freedom of movement in that world
- Reliability

How the VCS views the world

How the VCS views the world

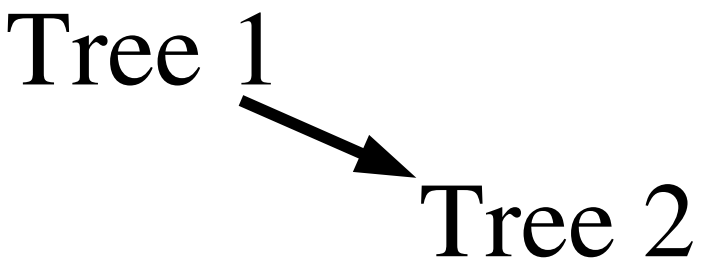
Tree 1

How the VCS views the world

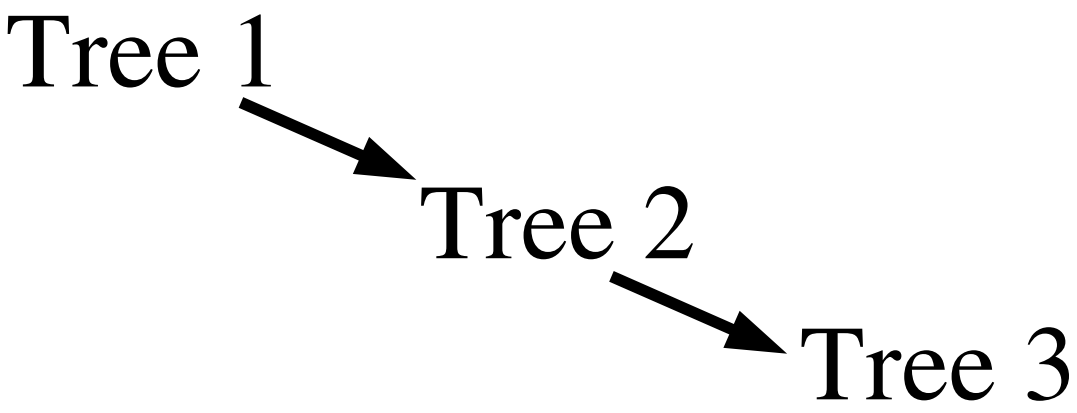
Tree 1

Tree 2

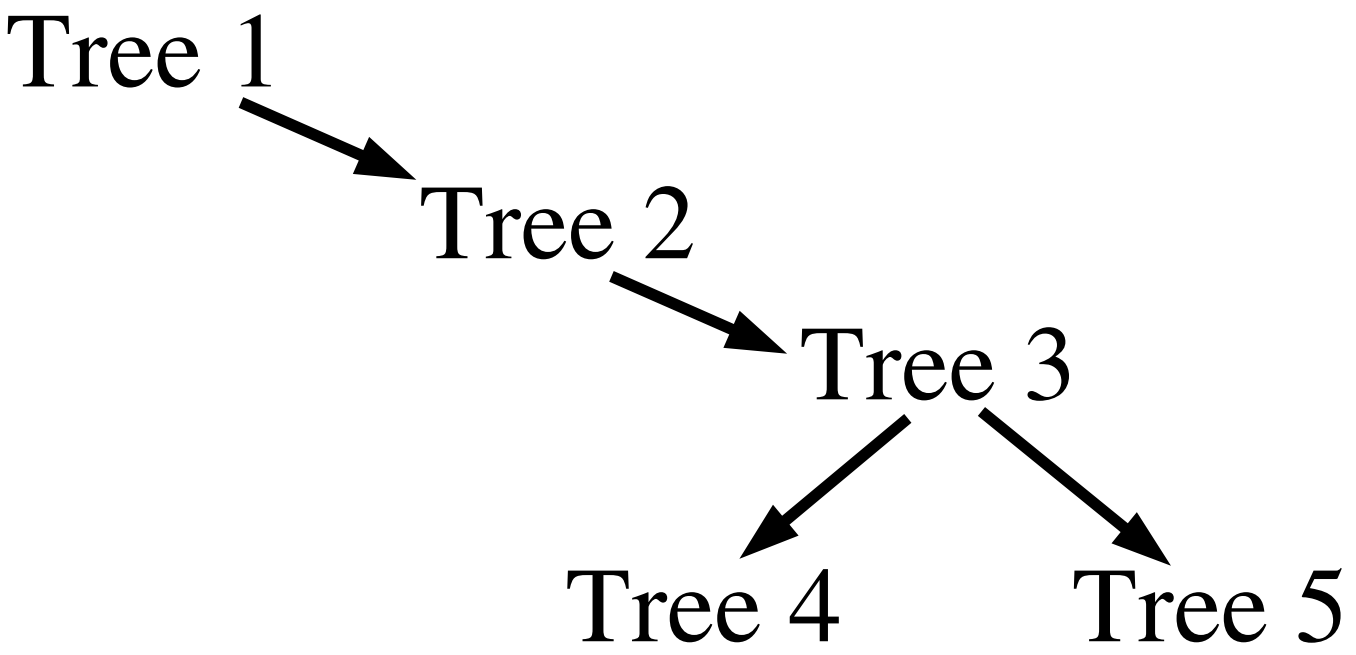
How the VCS views the world



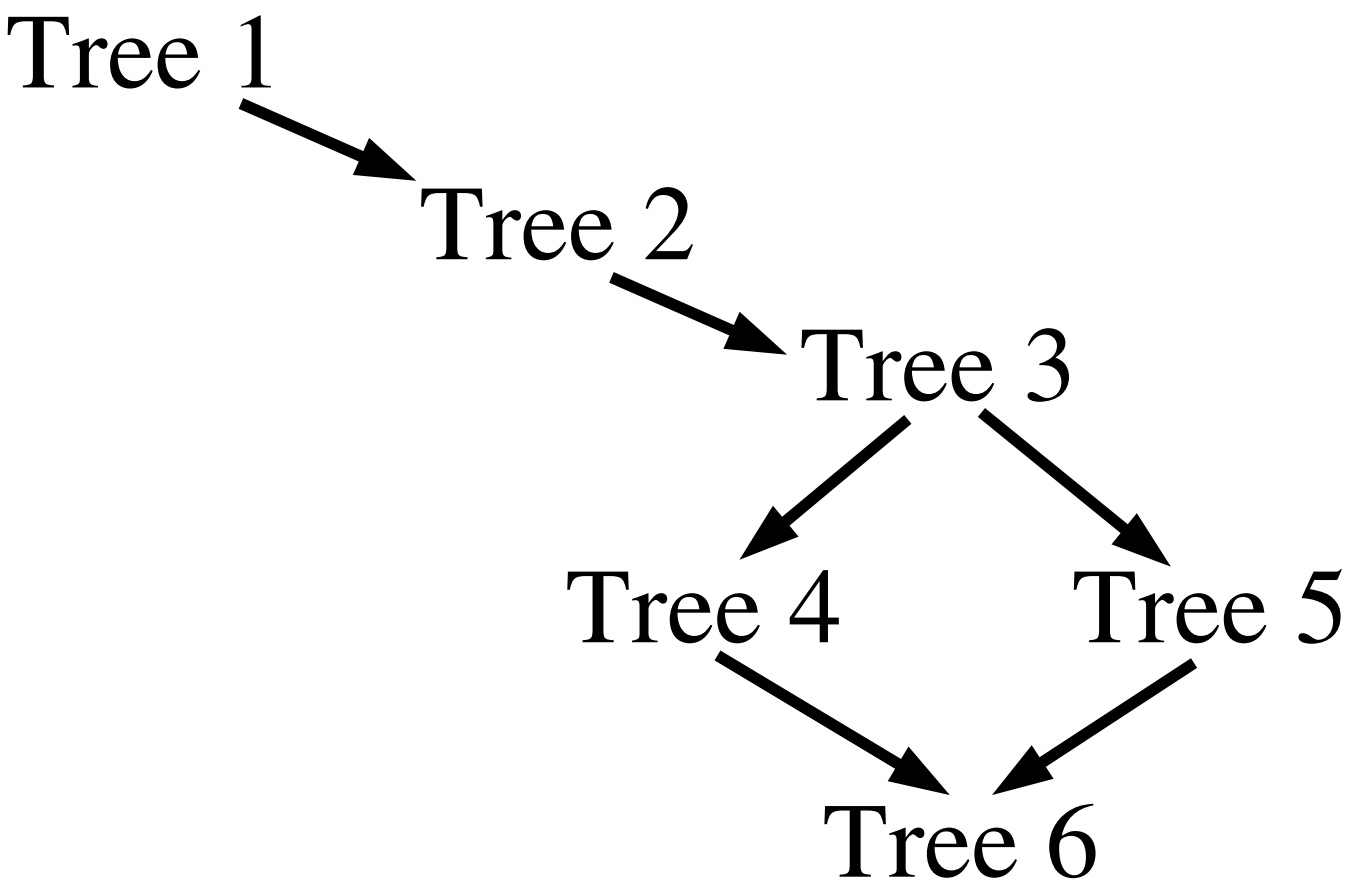
How the VCS views the world



How the VCS views the world



How the VCS views the world



Robustness

Robustness

- Redundancy

Robustness

- Redundancy
- Redundancy

Robustness

- Redundancy
- Redundancy
- Redundancy

But distributed systems are hard

- Stable public formats?
- Distributed data structures?
- Unique branch heads?
- Tracking synchronization state?

But distributed systems are hard

- Stable public formats?
- Distributed data structures?
- Unique branch heads?
- Tracking synchronization state?

Stable public formats

- Network formats *never* want to change

Stable public formats

- Network formats *never* want to change
- Storage formats *always* want to change

Stable public formats

- Network formats *never* want to change
- Storage formats *always* want to change

Decouple them

Stable public formats

- Side-effects:
  - Public formats optimized entirely for accurately representing domain
  - Anything monotone knows can be pulled out in a simple, documented, stable textual formats

But distributed systems are hard

- Stable public formats?
- Distributed data structures?
- Unique branch heads?
- Tracking synchronization state?

Distributed data structures

- “Pointers” require globally unique ids, which might turn out not to be so unique
- State may become inconsistent between clones
- Corrupt state can spread and poison the network

Distributed data structures

- “Pointers” require globally unique ids, which might turn out not to be so unique
- State may become inconsistent between clones
- Corrupt state can spread and poison the network
- Malicious users

Distributed data structures

- "Pointers" require globally unique ids, which might turn out not to be so unique → Use hashes
- State may become inconsistent between clones
- Corrupt state can spread and poison the network
- Malicious users

Distributed data structures

- "Pointers" require globally unique ids, which might turn out not to be so unique → Use hashes
- State may become inconsistent between clones → Use hashes
- Corrupt state can spread and poison the network
- Malicious users

Data formats: manifest

```
format_version “1”

dir “”

   file “AUTHORS”
content [53ba0a32a8d3d257787d12f2e37b8c1329c664b4]

   file “configure”
content [c7f70f0b3353a12bcf84a27f491e9caca5499c5f]
   attr “mtn:execute” “true”

dir “src”

   file “src/main.c”
content [cbd89a231305a1e00895e53403656c77c6337bfe]
```

Data formats: revision

```
format_version “1”

new_manifest [8bcb3f8761a67ac3a59abf89bd78801ff08a4d05]

old_revision [32cc8671c7ab40b7152d865dba589952635918d1]

add_dir “src/awesomeness”

patch "AUTHORS"
 from [071db1a513c3d68bc0c2b025399688549f0af6b7]
   to [43e4feb8f3375938b4382e1356af36b5e0ed59e6]
```

Hashes as pointers

- Side-effects:
  - Every revision id is a complete history checksum

Distributed data structures

- "Pointers" require globally unique ids, which might turn out not to be so unique → Use hashes
- State may become inconsistent between clones → Use hashes
- Corrupt state can spread and poison the network
- Malicious users

Handling corrupt state

Larry McVoy:

> “BK is a complicated system, there are >10,000 replicas of the BK database holding Linux floating around. If a problem starts moving through those there is no way to fix them all by hand. This happened once before, a user tweaked the ChangeSet file, and it costs $35,000 plus a custom release to fix it.”

Detecting corrupt state

- Side-effects:
  - Every monotone operation is exhaustively self-checking
    - (And of course we optimize this too)
  - If monotone reports success, you can count on that

But distributed systems are hard

- Stable public formats?
- Distributed data structures?
- Unique branch heads?
- Tracking synchronization state?

Unique branch heads

- Two options:
  - Make branches a local concept

Unique branch heads

- Two options:
  - Make branches a local concept
    - Hard to know what branches exist
    - Hard to find branches that do exist
    - Extra friction to collaboration

Unique branch heads

- Two options:
  - Make branches a local concept
    - Hard to know what branches exist
    - Hard to find branches that do exist
    - Extra friction to collaboration
  - Allow divergence in branches

Unique branch heads

- Two options:
  - Make branches a local concept
    - Hard to know what branches exist
    - Hard to find branches that do exist
    - Extra friction to collaboration
  - Allow divergence in branches

Branches

- New concept: “certs”
  - Signed key/value pairs attached to commits
  - A branch is “all commits that have a certain branch=<something> cert”

Branches

- Side-effects:
  - This actually makes “branch” a more meaningful concept
  - Crypto makes everything auditable
  - commit always succeeds
  - push/pull/sync always succeed

Branches

- Side-effects:
  - You have an arbitrary extension mechanism for interesting workflow management
  - Possible certs:
    - tags
    - “this revision is ready for review”
    - “this revision has passed review”
    - “this revision passes automated tests”
    - you tell me...

But distributed systems are hard

- Stable public formats?
- Distributed data structures?
- Unique branch heads?
- Tracking synchronization state?

Tracking synchronization state

- Need to be able to send only local changes
- Which requires keeping track of what those local changes are...
- Which is *more* state that can become corrupt

Tracking synchronization state

- Need to be able to send only local changes
- Which requires keeping track of what those local changes are...
- Which is *more* state that can become corrupt

Tracking synchronization state

- Need to be able to send only local changes
- Determine what changed on the fly, from scratch, at every sync

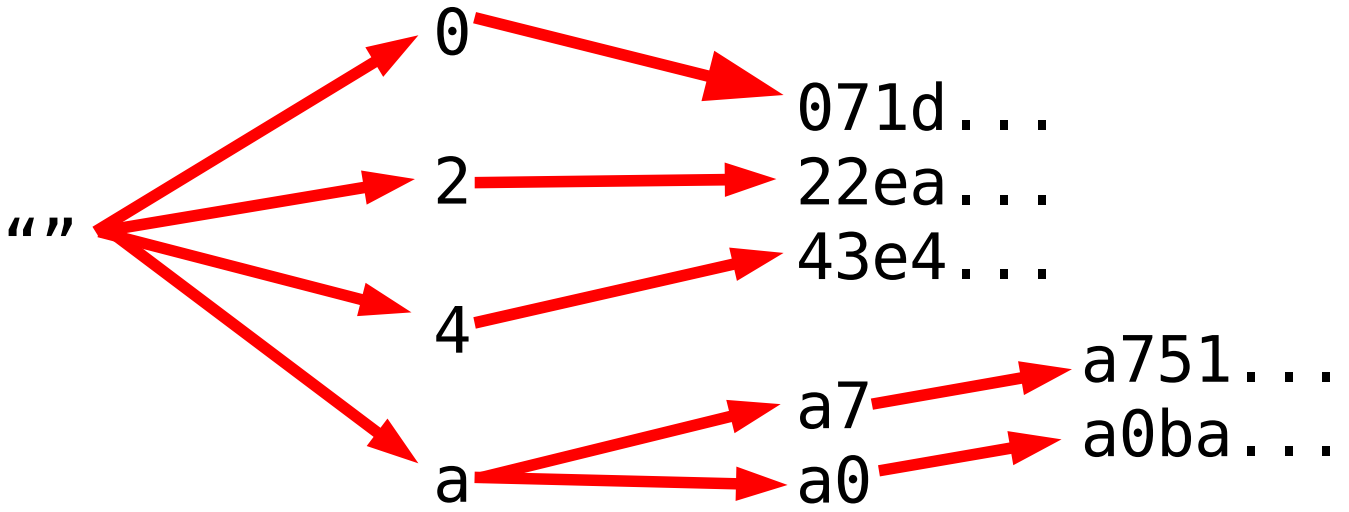
Tracking synchronization state

```
071db1a513c3d68bc0c2b02539968549f0af6b7
22ea2b7a70f2e65d3bc70b5f78785f38d052a994
43e4feb8f3375938b4382e1356af36b5e0ed59e6
a7516319a3afd6bb8c1719beca72290acad44554
a0ba013fbf8fcdf7ba399d671d1c28c1827d00af
```

Tracking synchronization state

```
071d...
22ea...
43e4...
        a751...
        a0ba...
```

Tracking synchronization state



Tracking synchronization state

- Arbitrary set synchronization
- Pipelining friendly
- O(d log n) bytes
- log(n)/2 round trips
  - where d = size of the difference
  - n = total size of the sets

Tracking synchronization state

- Side-effects
  - Synchronization is always fast, and always accurate
  - You never have to remember what needs pushing; monotone will notice if you forget

Distributed systems are hard

- But if you design a VCS with:
  - Simple representations
  - Reliability
  - Security
  - Comfortable free workflow

- Then distributed-ness is free!
  - And necessary for reliability

Teaser slide

- There's more I don't have time for:
  - Our merge model is provably correct, efficient, and also understandable by users
  - UI for “peripheral vision”
  - We're working on improving our trust delegation – granting and revoking permissions, also without dependence on a central server

http://venge.net/monotone