Peer-to-Peer based Version Control

QuaP2P – DFG Research Group: Improving the Quality of P2P Systems - www.quap2p.de

ES Real-Time Systems Lab
Dept. of Electrical Engineering and Information Technology
Department of Computer Science (Adjunct Member)
TUD – Technische Universität Darmstadt

Patrick Mukherjee, Christof Leng, Wesley W. Terpstra, Andy Schürr

Patrick.Mukherjee@ES.tu-darmstadt.de
Tel.+49 6151 164839

© author(s) of these slides 2008 including research results of the research network ES and TU Darmstadt otherwise as specified at the respective slide
Motivation / Application Area: GSD (PIPE)

Facts:

- New trend in software development → global distribution
- No tools supporting new trend

- Usage of tools for on-site development
  - Centralized communication
  - Communication bottleneck
  - Additional latency
  - Single point of failure
  - High set-up and maintenance costs

→ We need adequate support for global distributed software development

Most important tool: Version Control
Outline

- Motivation
  - State of the art version control systems
  - Problem with eventual consistency
  - Our approach to distributed version control
    - Repository distribution
    - Operations
    - Failure handling
- Demo
- Conclusion/future work
State of the Art

Approaches to Version Control (VCS)

- Centralized VCS
  - CVS, SubVersion, ...
  - Not distributed (see slide 2)

- Distributed VCS
  - GIT, Mercurial, ...
  - No head brunch
  - Artifacts are not always available
  - Only eventual coherency

- P2P VCS
  - Pastwatch, Code Co-Op
  - Only eventual consistency
  - High load of system messages
Problem with Eventual Consistency (Conflicting Concurrent Changes)

Update CVS 1

Update CVS 2
Outline

- Motivation
- State of the art version control systems
- Problem with eventual consistency
- Our approach to distributed version control
  - Repository distribution
  - Operations
  - Failure handling
- Demo
- Conclusion/future work
Peer-to-Peer Based Version Control: Repository Distribution

Based on structured p2p overlays:

- Any two peers route to same peer
- Each repository Item is controlled by one specific peer (the PrimaryOwner (PO))
- DHT-liked mapped (using replicating peers (RP))

Repository distribution:

- A version is immutable (changes create new versions)
- Each file has a companion metadata file
  - Stores information about the file (e.g. author, version no., …, # words)
- All versions (revisions & variants) are stored at the PO & RP
Check-Out

- **Check-Out latest Version:**
  - Ask for latest Version No.
    - Lookup(Repository Item ID)
    - → can only be answered by Primary Owner or Replicating Peers
  - Ask for specific Version No.
    - Lookup(Repository Item ID, Version No.)
    - → can be answered by any peer
Commit

1. Send commit Request: store(Repository Item ID, patch, based on Version No)
2. PO replicates to RP (if valid = based on latest version) (with Handshake)
3. PO sends acknowledgment (ACK)
   - User doesn’t receive ack → commit request is repeated
Concurrent Commit

PO synchronises concurrent changes:
- Not valid, if based on outdated Version:
  - changes were already applied (ACK was lost)?
    - (based on pre-latest version && latest author == User)
  - Other peer just commited changes (concurrent commit was serialized)
    - Apply commit of *first* peer
    - Reject commit of second peer → send him latest version (automerged on client)
    - Conflict solved on client
Failing Peers: Replica Peer (RP)

If committing peer fails:
- Commit request gets served (if valid applied)
- Peer is notified by repeated commit

If RP fails
- PO notice (keep-alive) and replaces with next neighbor
- Note: also happens if closer peer joins
  - Data from old RP will not be removed
  - New RP requests only missing versions
Failing Peers: Primary Owner (PO)

If PO fails

- Requests gets routed to RP who is closest to Data ID
- Note: this happens if closer peer enters network as well!
  - Data from failed PO will not be removed
- New PO updates his RPs
Germany, officially known as the Federal Republic of Germany, is a country in Central Europe. It is bordered to the north by the North Sea, Denmark, and the Baltic Sea; to the east by Poland and the Czech Republic; to the south by Austria and Switzerland; and to the west by France, Luxembourg, Belgium, and the Netherlands. The territory of Germany covers 357,021 km² and is influenced by a temperate climate, seasonal climate. With over 82 million inhabitants, it has the largest population of any member state of the European Union and is home to the List of countries by immigrant population, third-largest number of international migrants worldwide.

Contents

1. History
   1. Germanic tribes
   2. Holy Roman Empire (962 - 1806)
   3. Restoration and revolution (1814 - 1871)

Version 5, Cliff, Sun Sep 07 02:39:46 CEST 2008
Conclusions/Future Work

! Software development is distributed
→ development artifacts should be distributed as well
✓ Data correctness on distributed copies is preserved
   ✓ Updates of distributed copies are consistent
   ✓ Relation between artifacts are maintained (correctness)
   ✓ Changes which could influence linked artifacts are propagated

Future Work
- Support for folders
- Offline version control

- 3-Phase Commit for atomic commit of multiple files
- Evaluation
Thank You for Your Attention.

Peer-to-Peer based Version Control

QuaP2P –
DFG Research Group: Improving the Quality of P2P Systems - www.quap2p.de

ES Real-Time Systems Lab

Patrick Mukherjee, Christof Leng, Wesley W. Terpstra, Andy Schürr

Patrick.Mukherjee@ES.tu-darmstadt.de
Tel. +49 6151 164839

© author(s) of these slides 2008 including research results of the research network ES and TU Darmstadt otherwise as specified at the respective slide
Decentralized Version Control (DVCS)

No consistency is archived!

- Local VCS-repository on each peer
- Updates get *pushed or pulled on demand* into local repository
- Merge conflicts gets decided local (decision of peers may differ)
- Changes are not propagated (danger of ‘double work’)
- All changes could be pushed to a *Head Peer*
- Head = Server

**Implementations:**

§ Mercurial, git, …
Transaction-logic (aka atomic commit (3PC))
Peer-to-Peer Based Version Control System (2)

Data structure:

- File has companion metadata file
  - (contains information about artifact, like author, links to other documents, …)
- Folders: metadata without artifact (lists content)

```
<link source="1" v=1.4 /><link target="2" v=1.2/>
<direction="to"/>
<status="active"/>
</link>

<links validFor ="1.2">
  (...)
  <link partner="a"/>
  <link partner="c"/>
  (…)
</links>
```
Why 2PC is sufficient

1. commit
2. prepare to replicate
2. ready to replicate
2. replicate!
2. ready to replicate
2. apply

Peer

PO

PO

RP