Version Control Systems

Robert M. Dondero, Ph.D.
Princeton University

Objectives

- You will learn/review:
  - Version control systems (VCSs)
    - As mechanisms for…
    - Maintaining file versions
    - Safely sharing files with teammates

Motivation

- **Problem**
  - Programming involves discovery
  - Discovery involves making mistakes
  - Programmers sometimes must revert to a previous **version**
  - Programmers require **versioning**
  - Programmers require a ***version control system (VCS)***

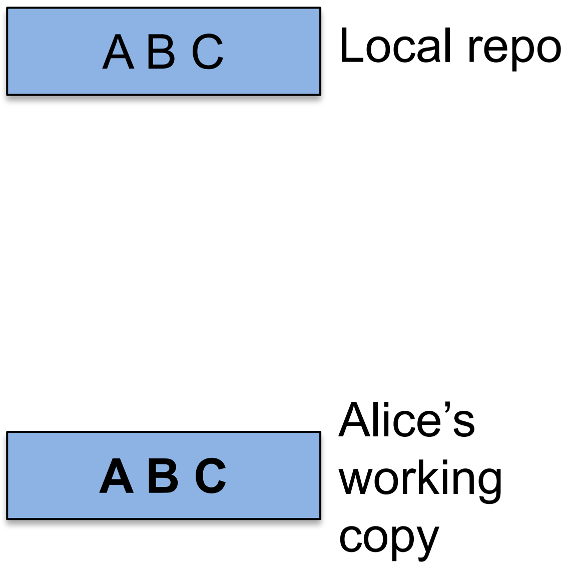
Agenda

- **Personal VCSs**
- Centralized VCSs
- Locking VCSs
- Merging VCSs
- Distributed VCSs

Personal VCSs

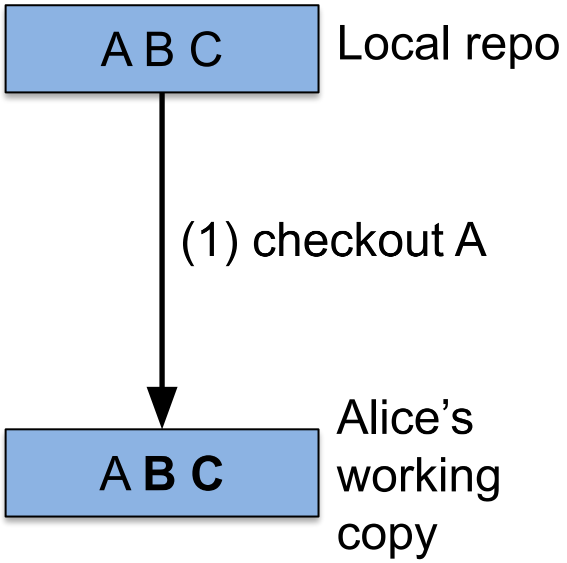
- **Solution**: personal VCS
  - Provides versioning
  - Popular system: ***Revision Control System (RCS)***
    - Still bundled with Linux
    - Readily available for Mac and MS Windows
- **Example**:

Personal VCSs

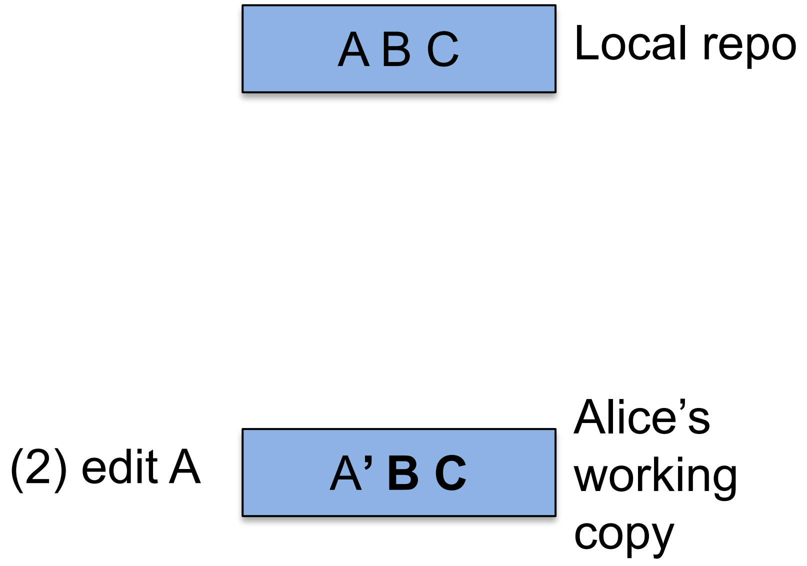


boldface =>
file is read-only

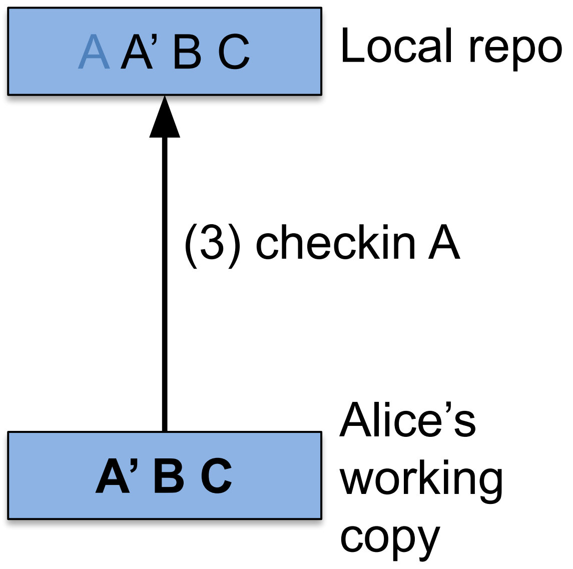
Personal VCSs



Personal VCSs



Personal VCSs



Old version of file A is retained in local repository
Old version can be retrieved to working copy if necessary

Personal VCSs

- **Problem**:
  - Programmers often must work in teams
  - Programmers need both version control and **file sharing**

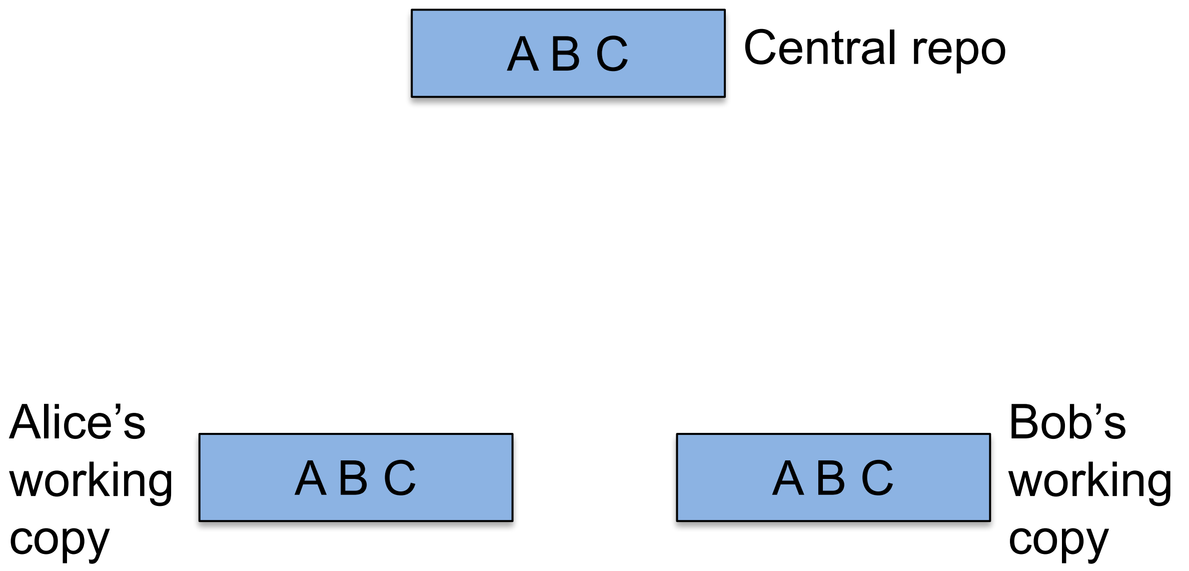
Agenda

- Personal VCSs
- **Centralized VCSs**
- Locking VCSs
- Merging VCSs
- Distributed VCSs

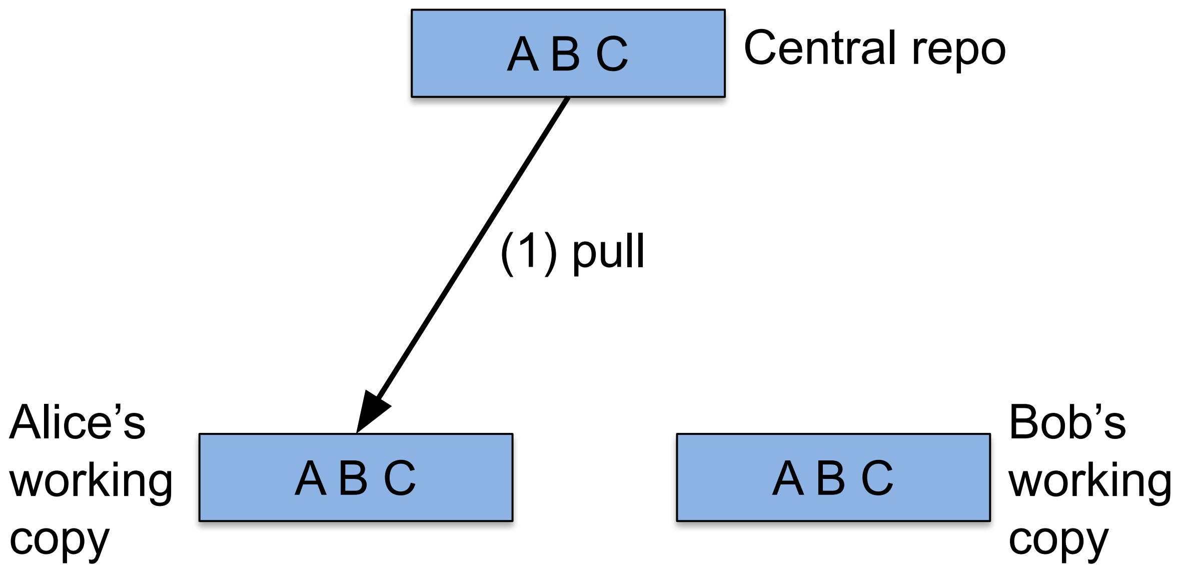
Centralized VCSs

- **Solution**: centralized VCS
  - Version control + ***central repository***

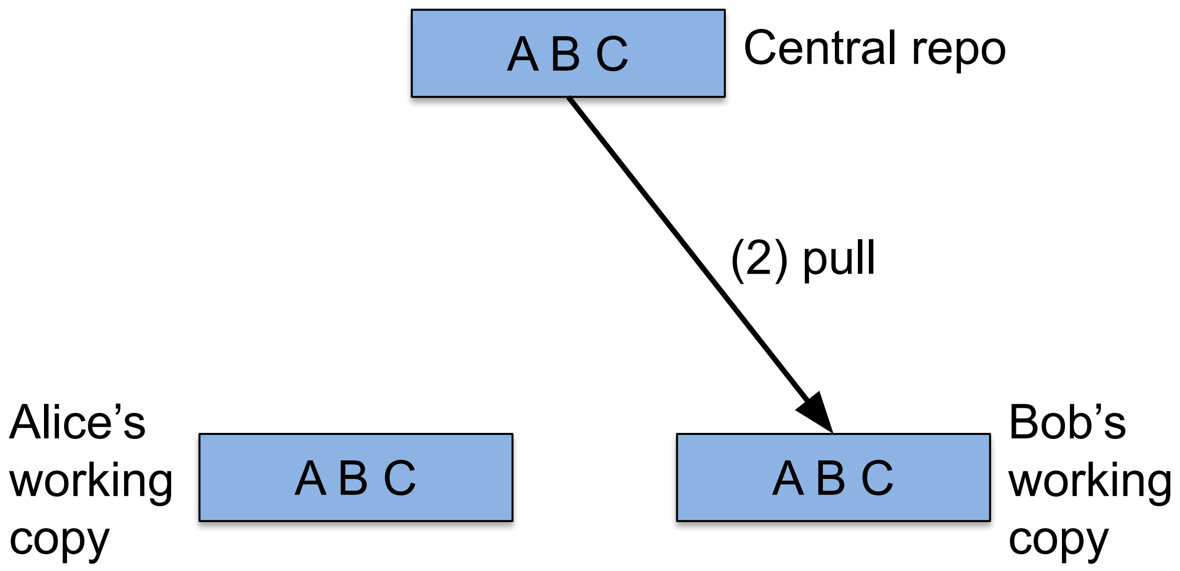
Centralized VCSs



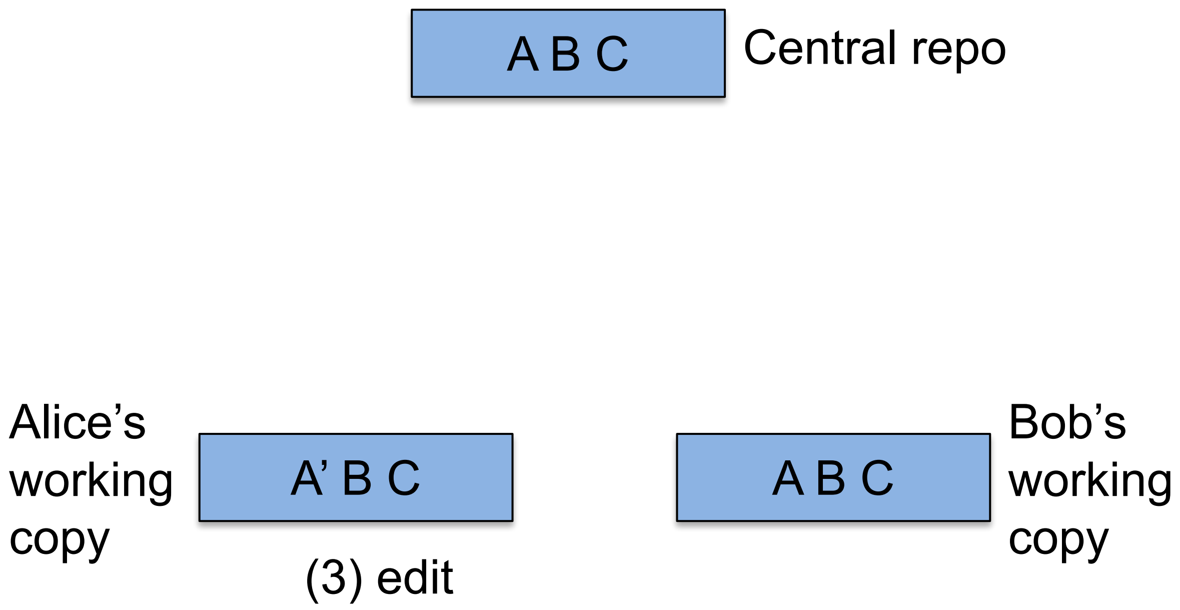
Centralized VCSs



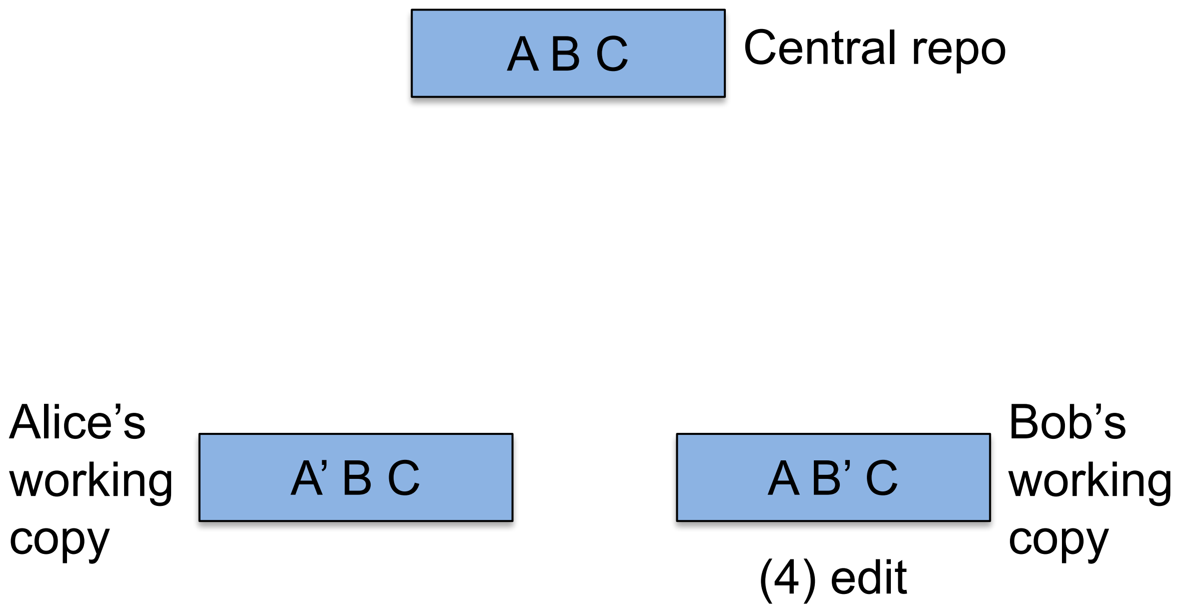
Centralized VCSs



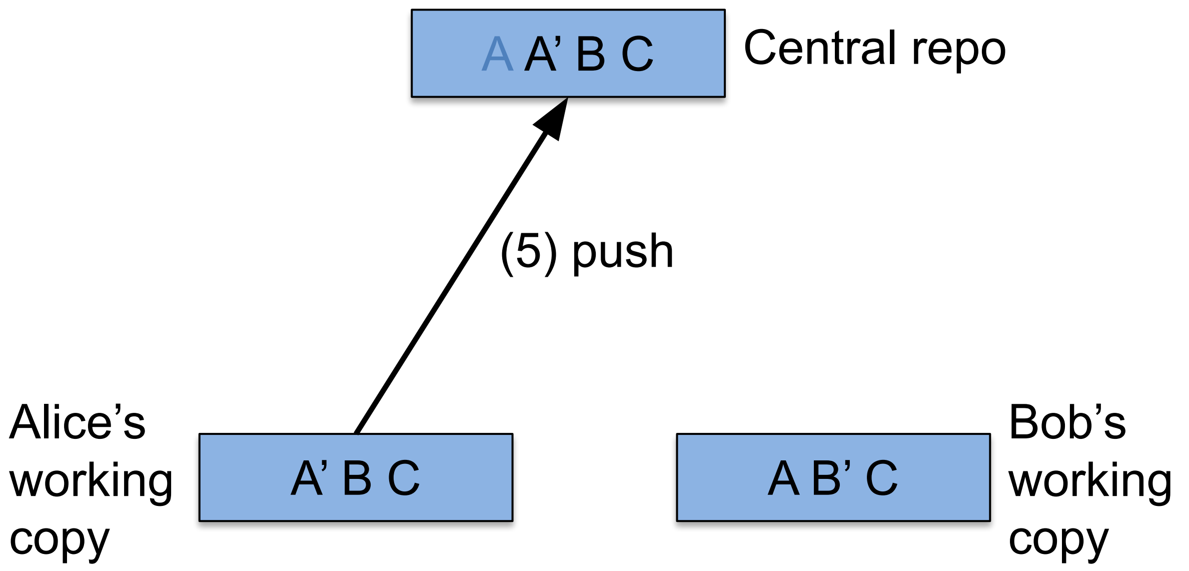
Centralized VCSs



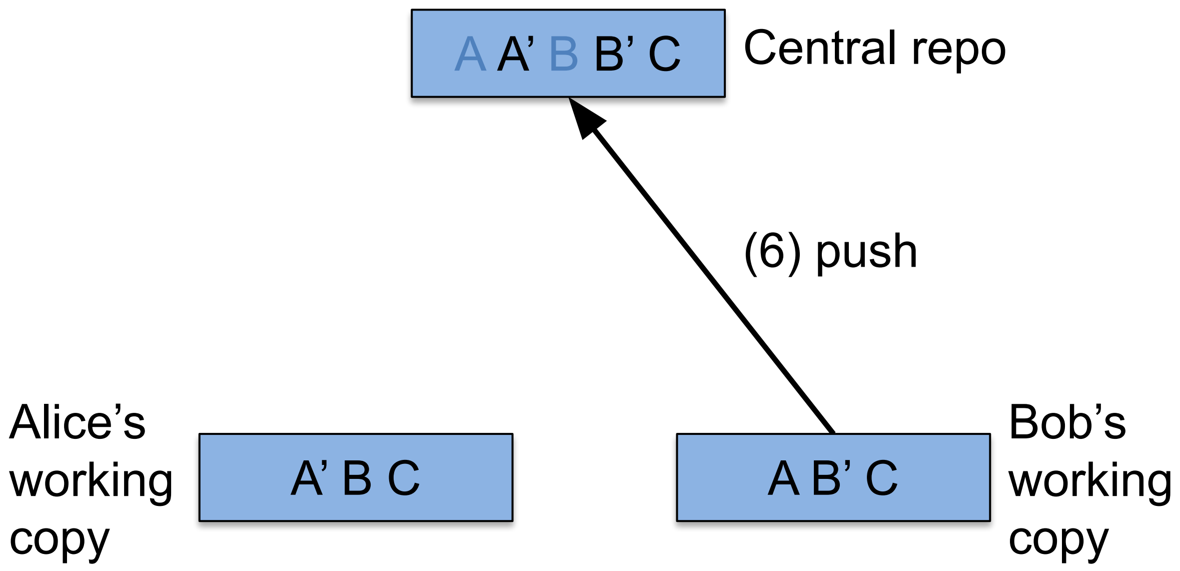
Centralized VCSs



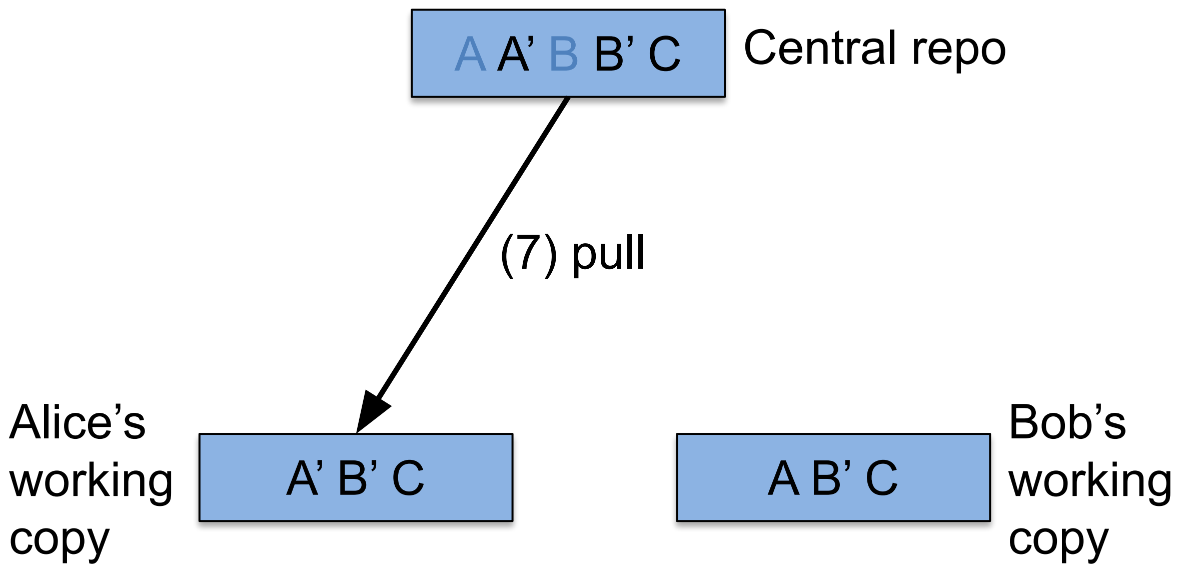
Centralized VCSs



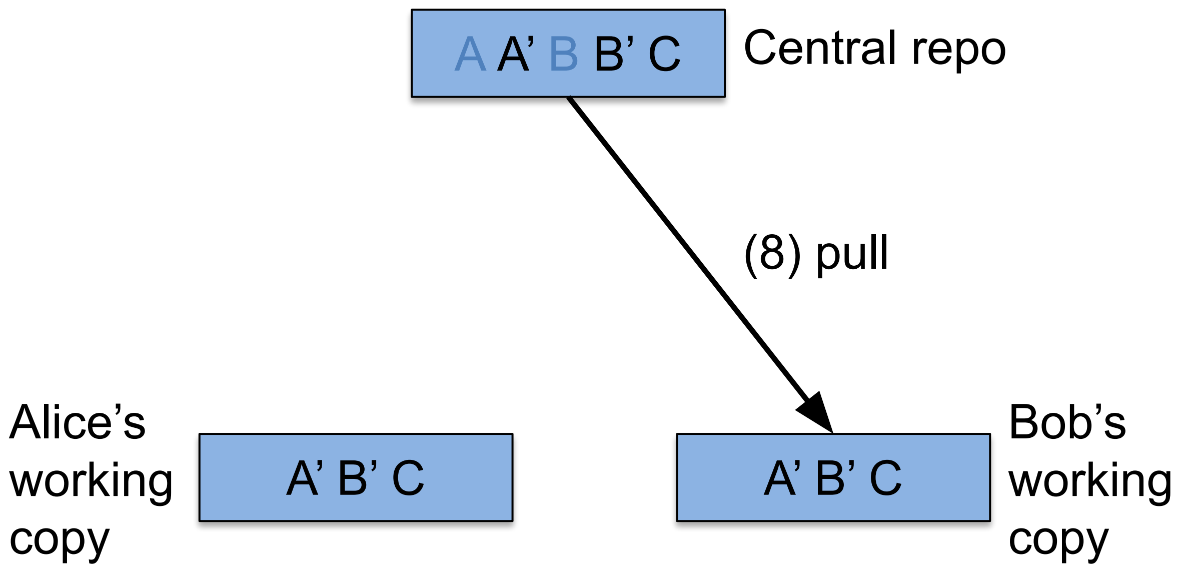
Centralized VCSs



Centralized VCSs



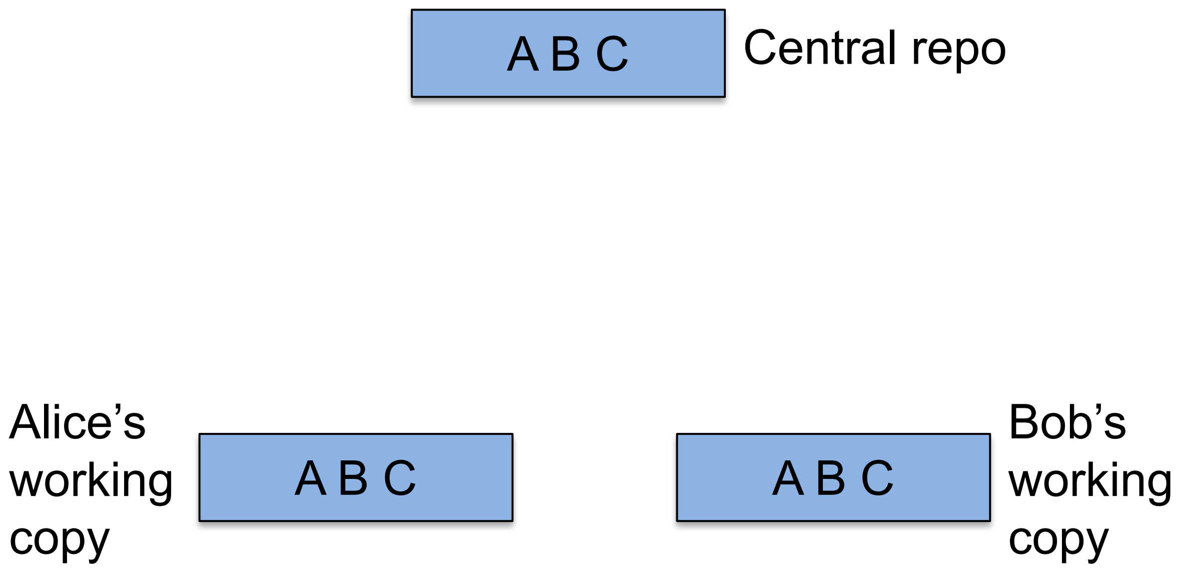
Centralized VCSs



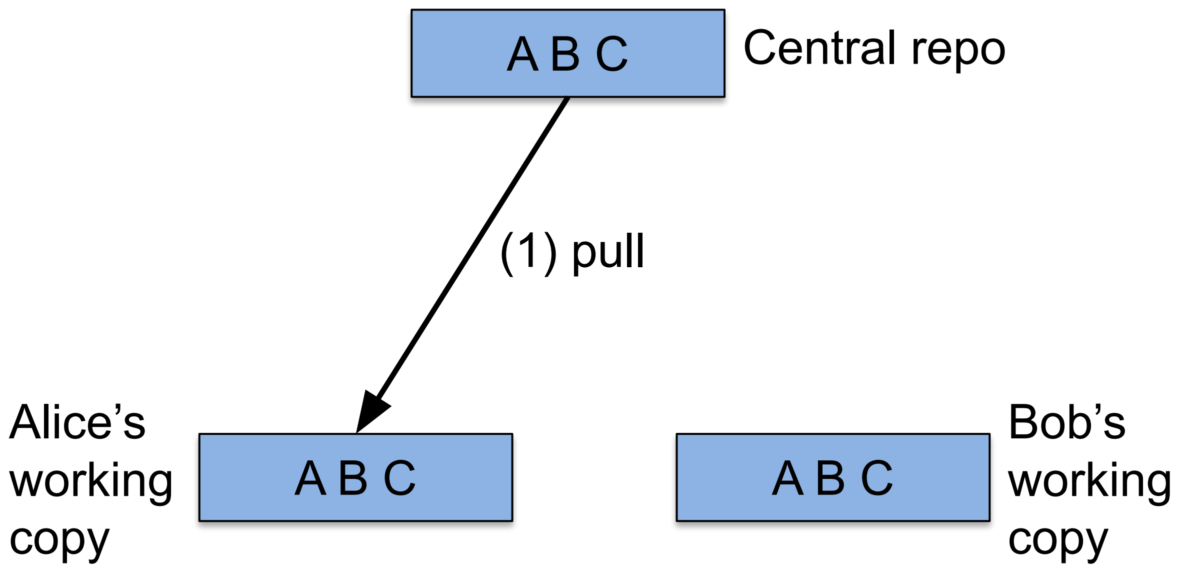
Centralized VCSs

- **Problem**: ***Conflicts***
- **Example**:

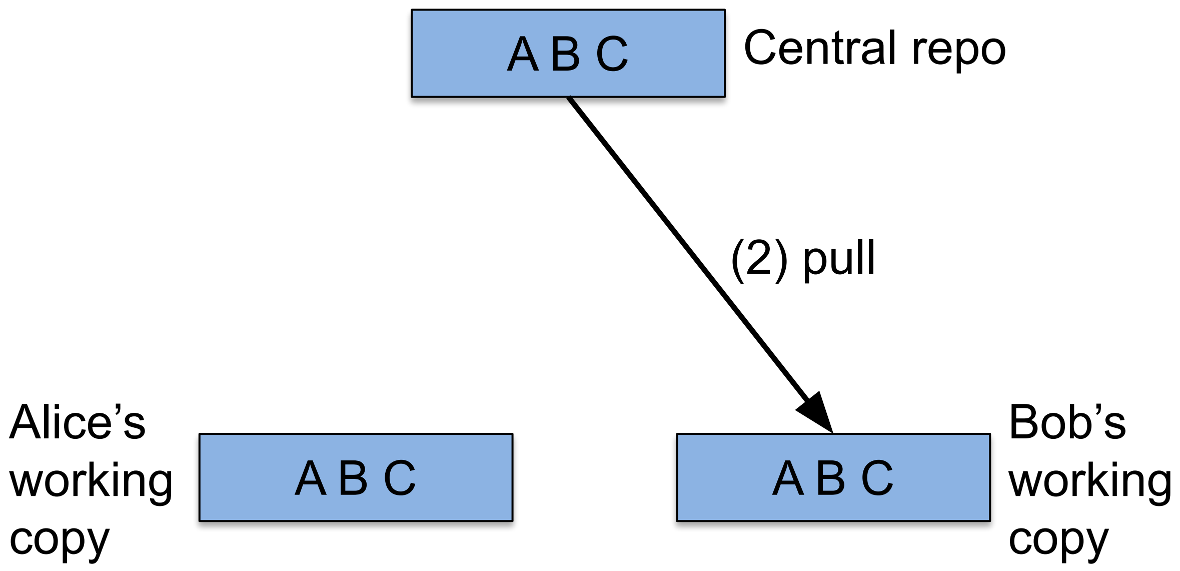
Centralized VCSs



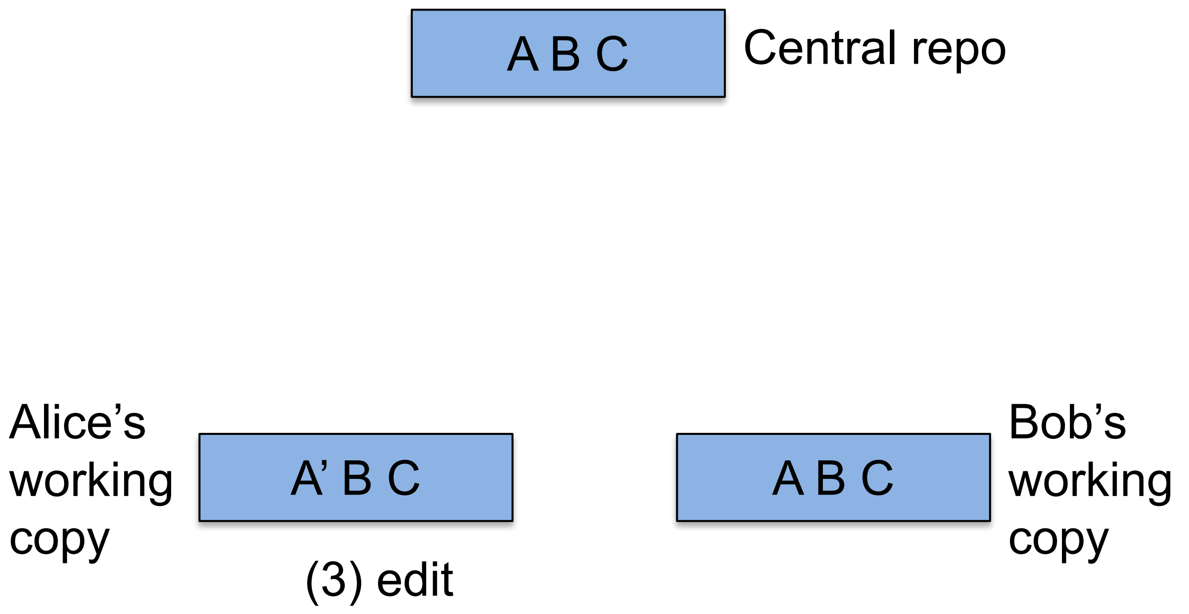
Centralized VCSs



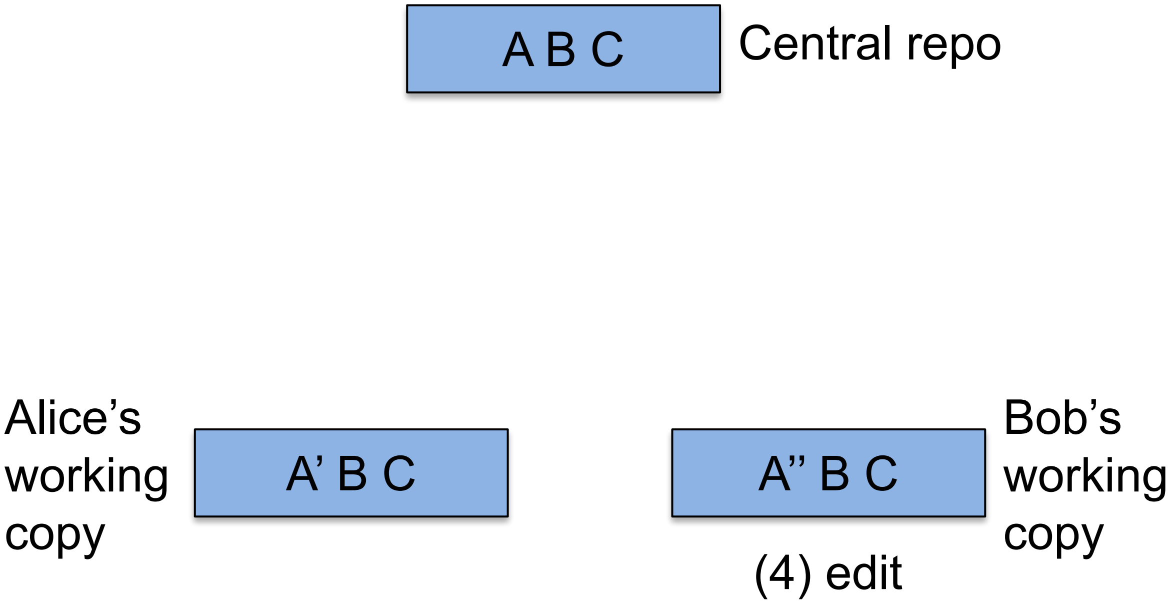
Centralized VCSs



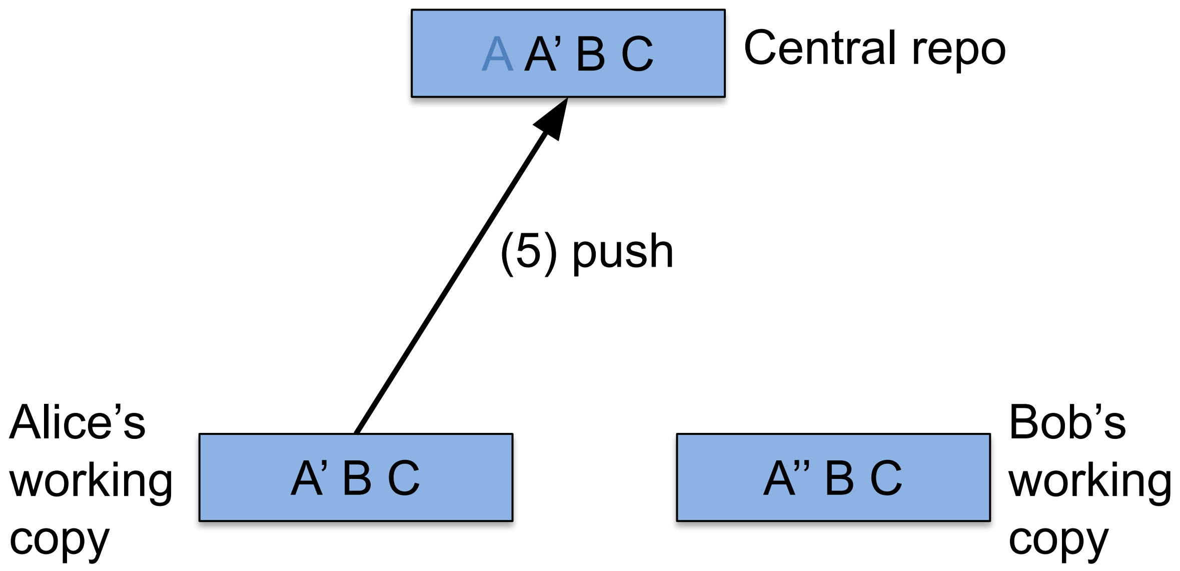
Centralized VCSs



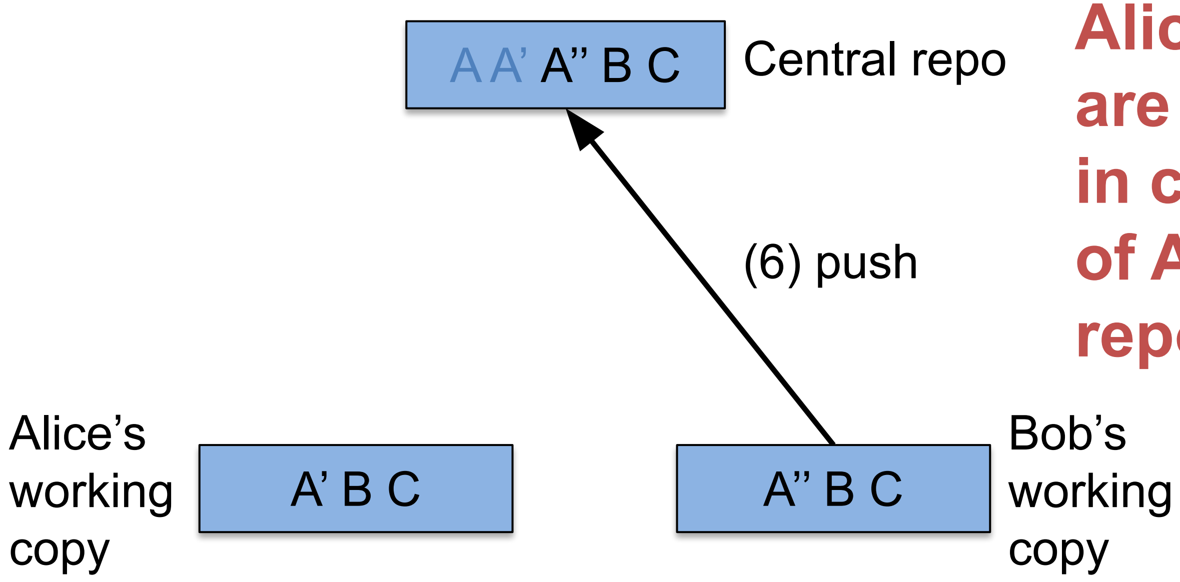
Centralized VCSs



Centralized VCSs

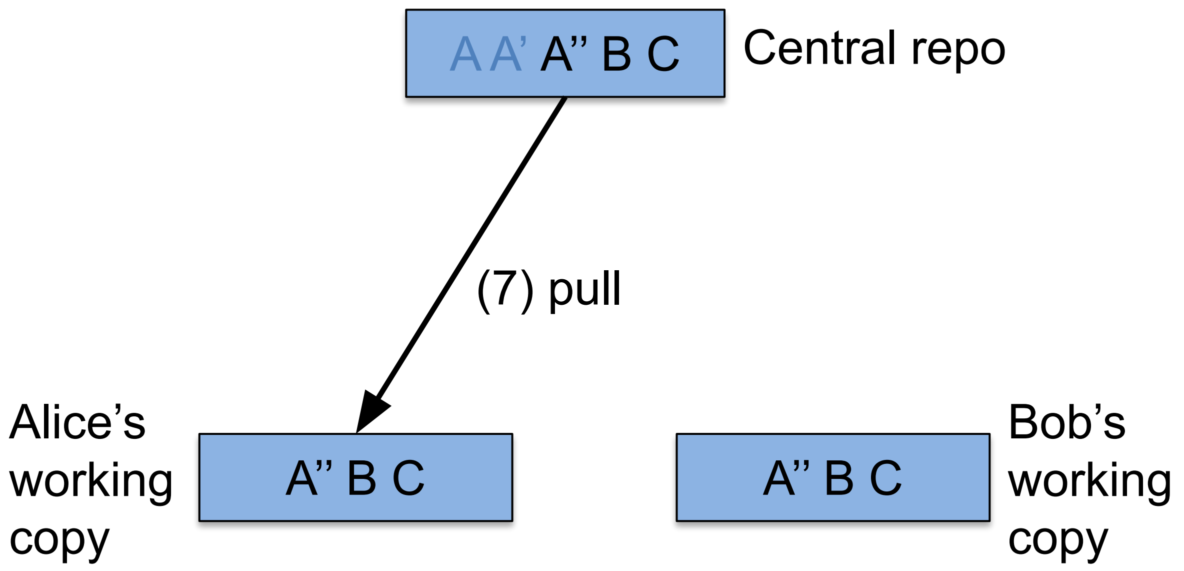


Centralized VCSs



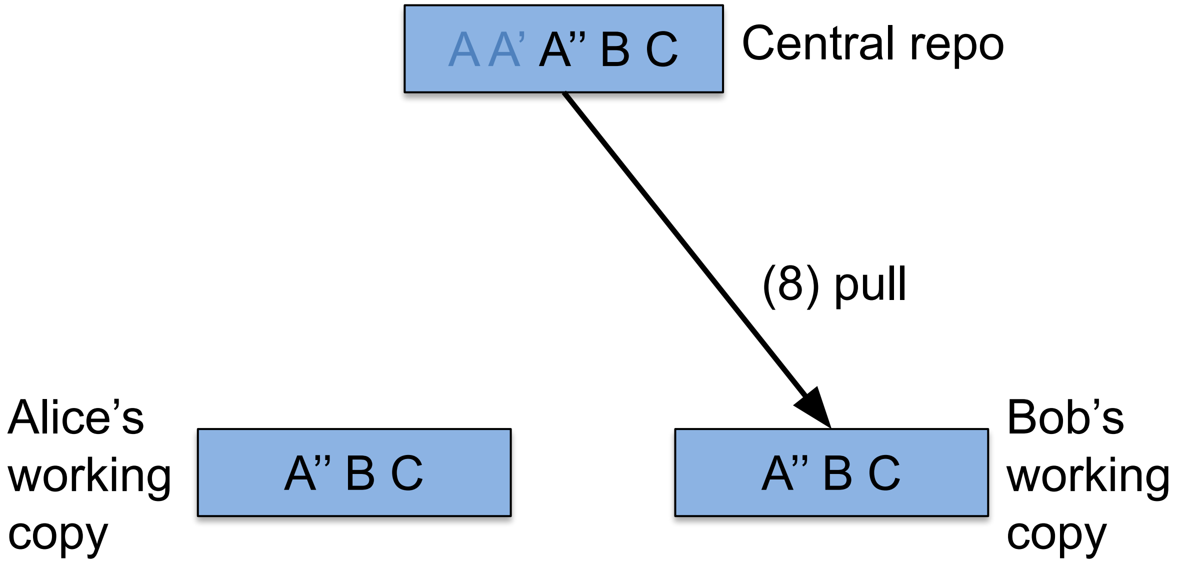
Alice's edits are not present in current version of A in central repo!!!

Centralized VCSs



Alice loses her own edits!!!

Centralized VCSs



Bob never receives Alice's edits!!!

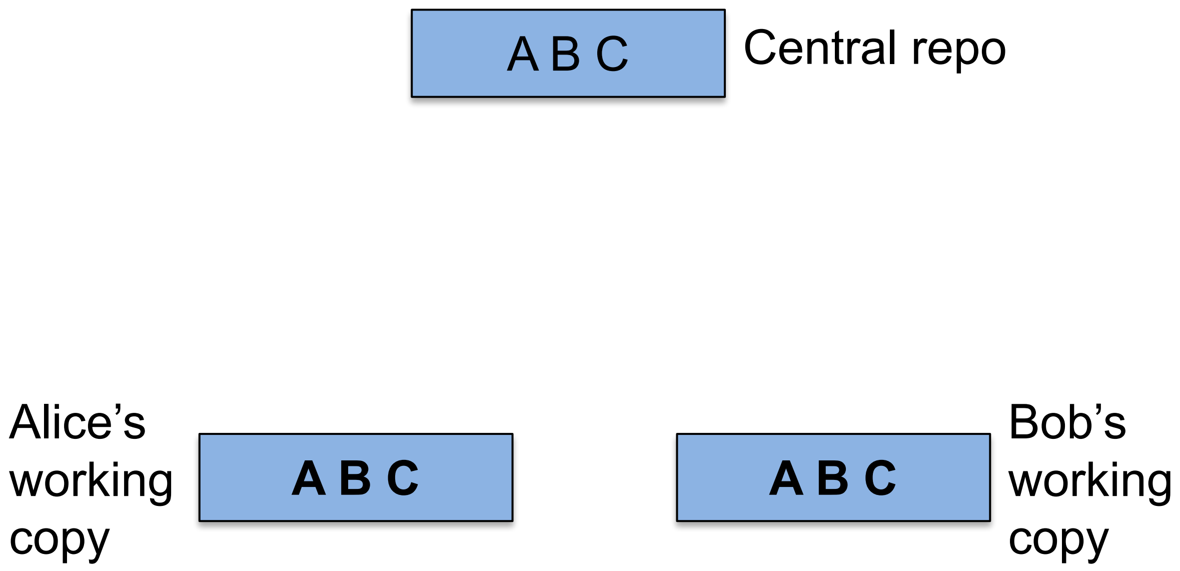
Agenda

- Personal VCSs
- Centralized VCSs
- **Locking VCSs**
- Merging VCSs
- Distributed VCSs

Locking VCSs

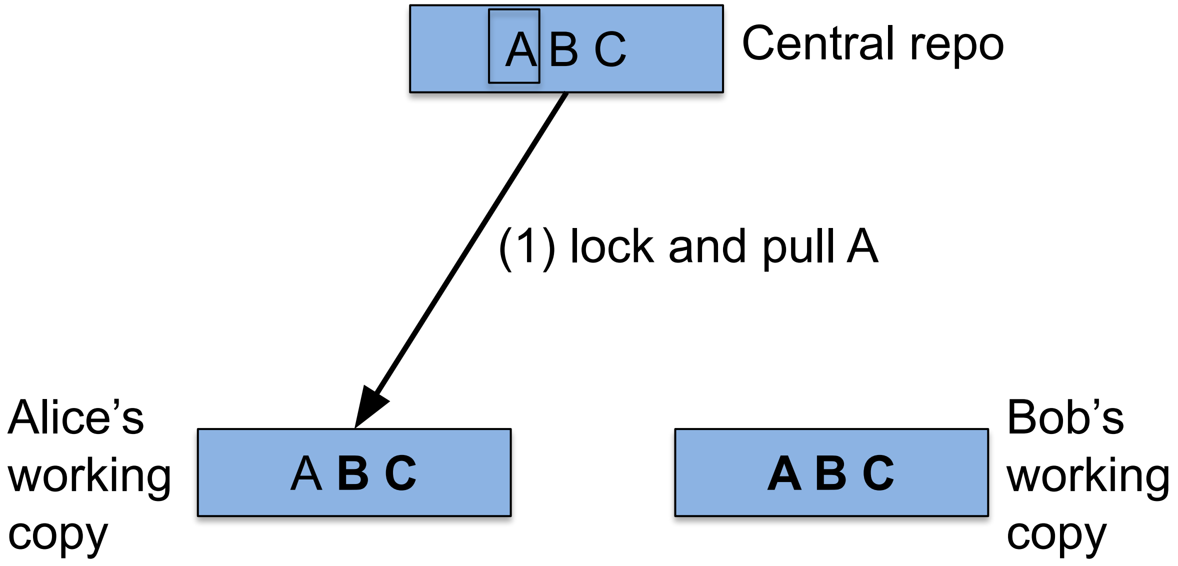
- **Solution**: locking VCS
  - Version control + central repo + ***locking***
  - Programmer requests lock on file A
  - VCS grants lock iff file A currently is unlocked
  - Popular system: ***Polytron Version Control System (PVCS)***
- **Example**:

Locking VCSs



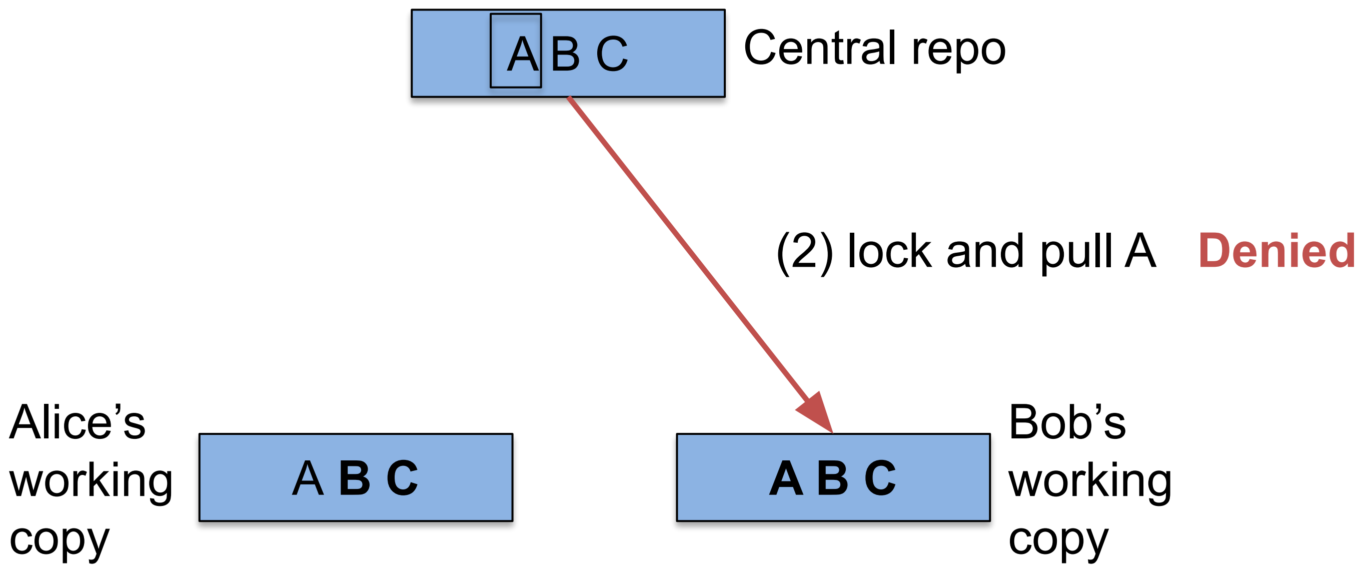
boldface =>
file is read-only

Locking VCSs

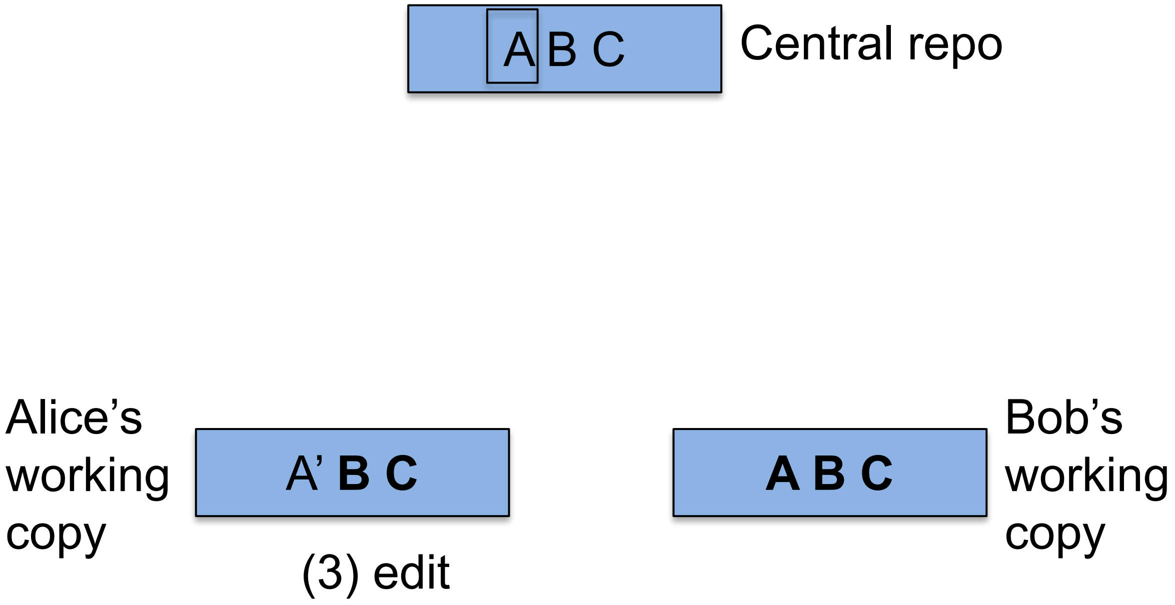


box =>
file is locked

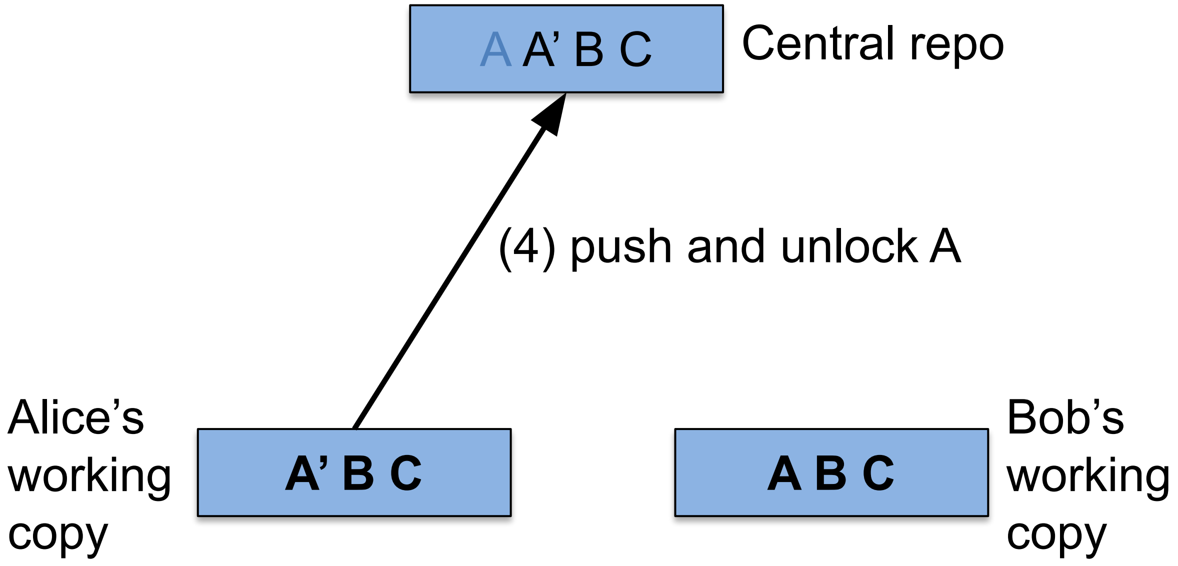
Locking VCSs



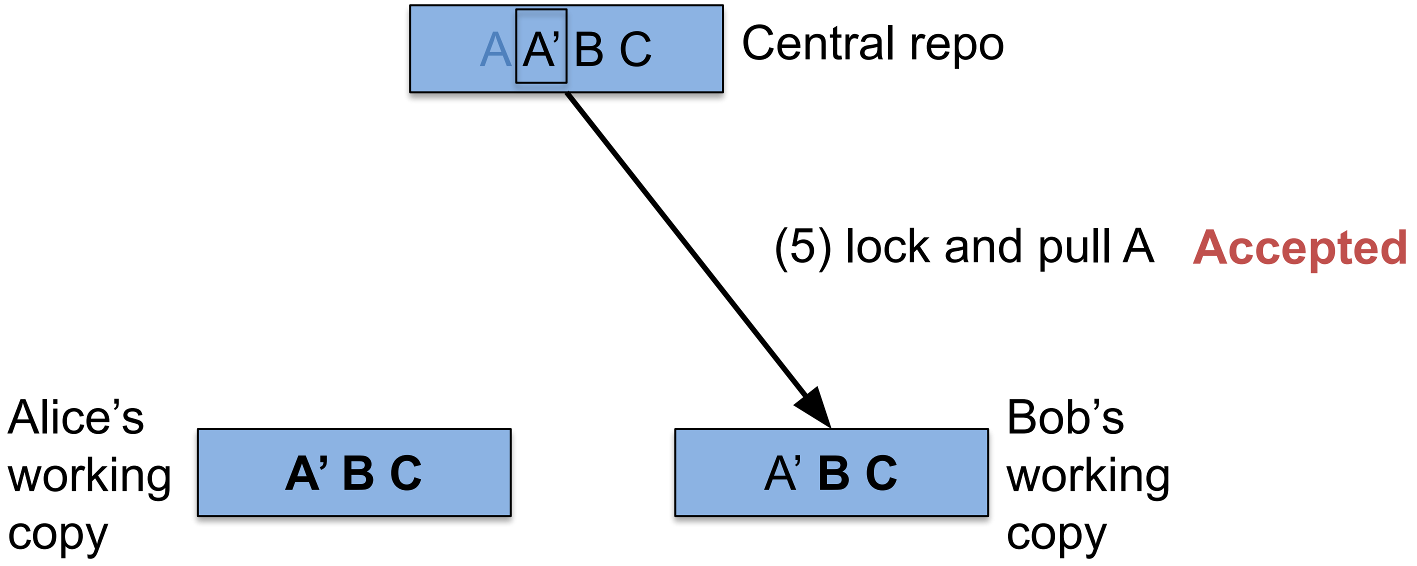
Locking VCSs



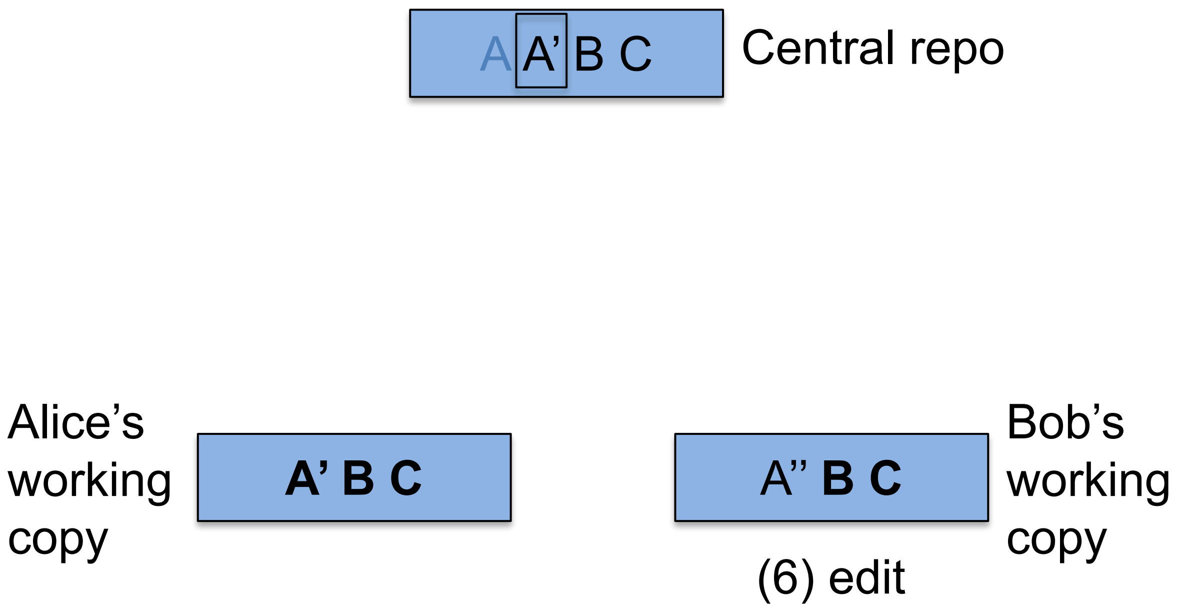
Locking VCSs



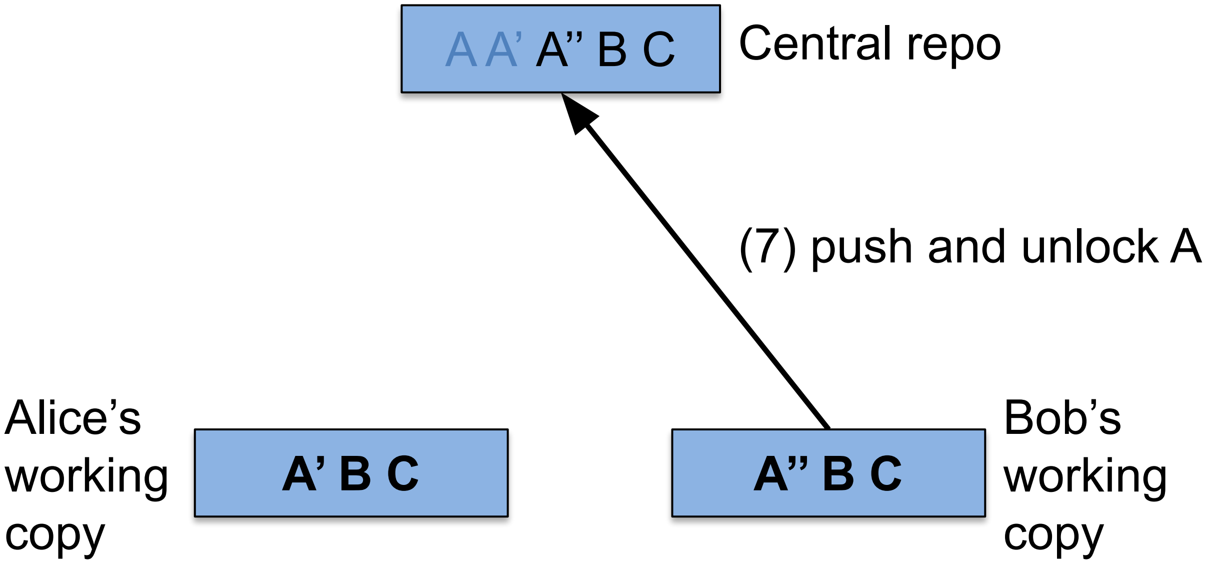
Locking VCSs



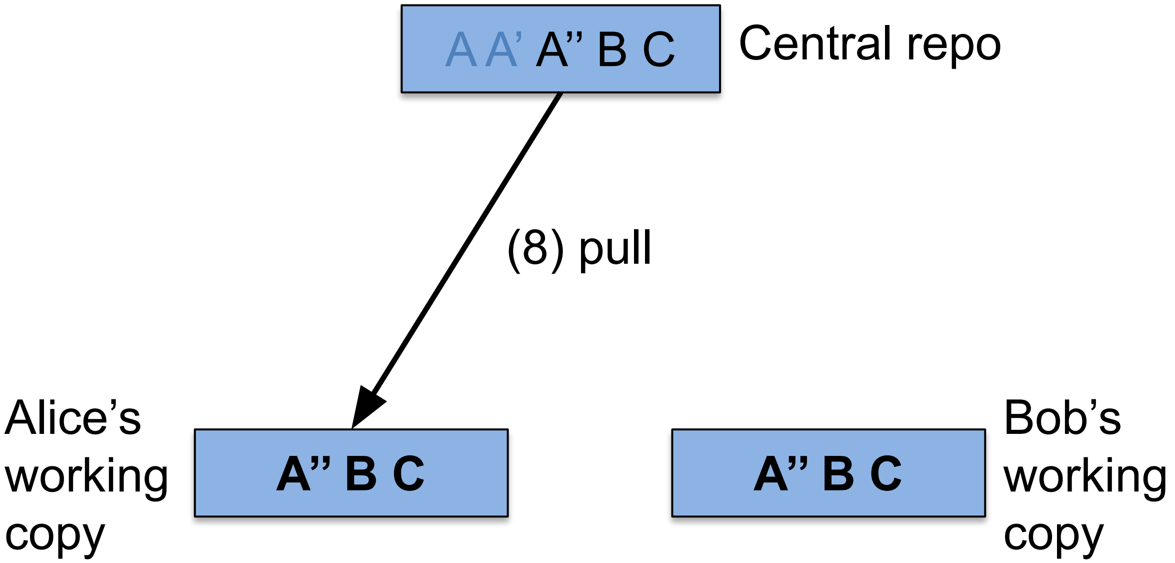
Locking VCSs



Locking VCSs



Locking VCSs



Locking VCSs

- **Problems:**
  - Awkward administration
    - Alice may forget to unlock A
  - Unnecessary serialization
    - Alice and Bob may be editing *different parts* of A
  - Potential deadlock
    - Suppose A & B depend upon one another
    - Alice locks A; Bob locks B
    - Alice needs lock on B; Bob needs lock on A

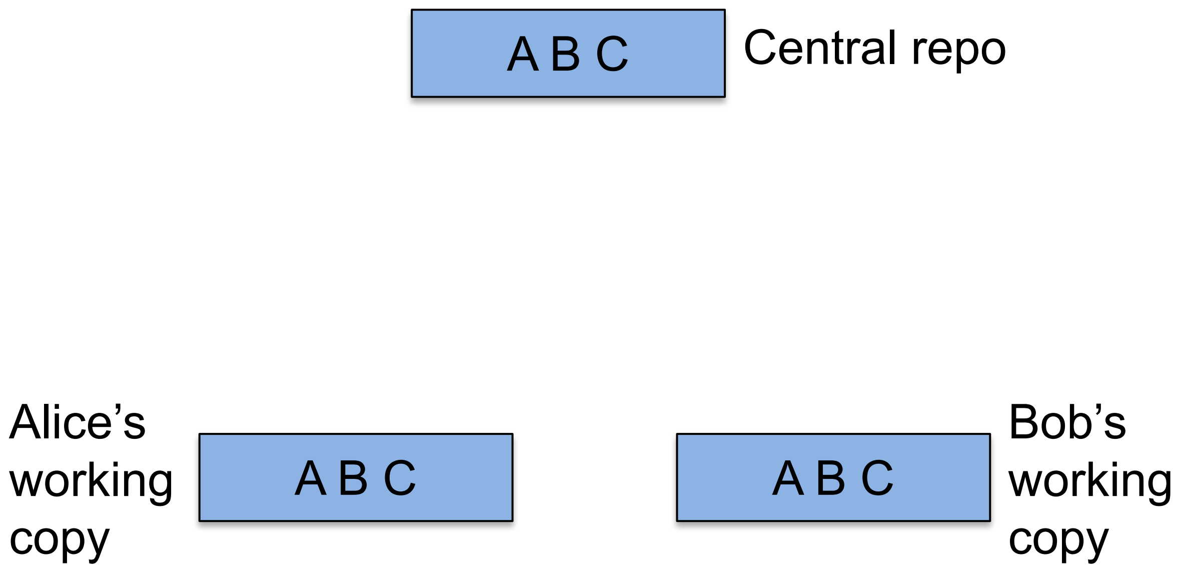
Agenda

- Personal VCSs
- Centralized VCSs
- Locking VCSs
- **Merging VCSs**
- Distributed VCSs

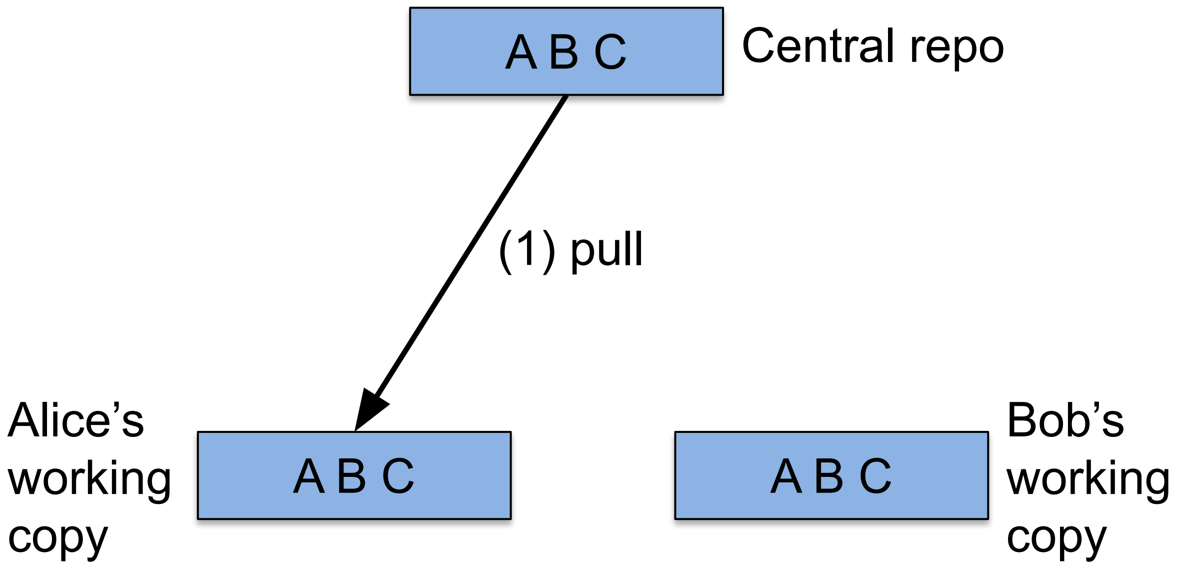
Merging VCSs

- **Solution**: merging VCSs
  - Version control + central repo + ***merging***
  - Motivated by open source development
    - Locking is unrealistic
  - Popular systems: ***Concurrent Versioning System (CVS), Subversion, Perforce***
- **Example**:

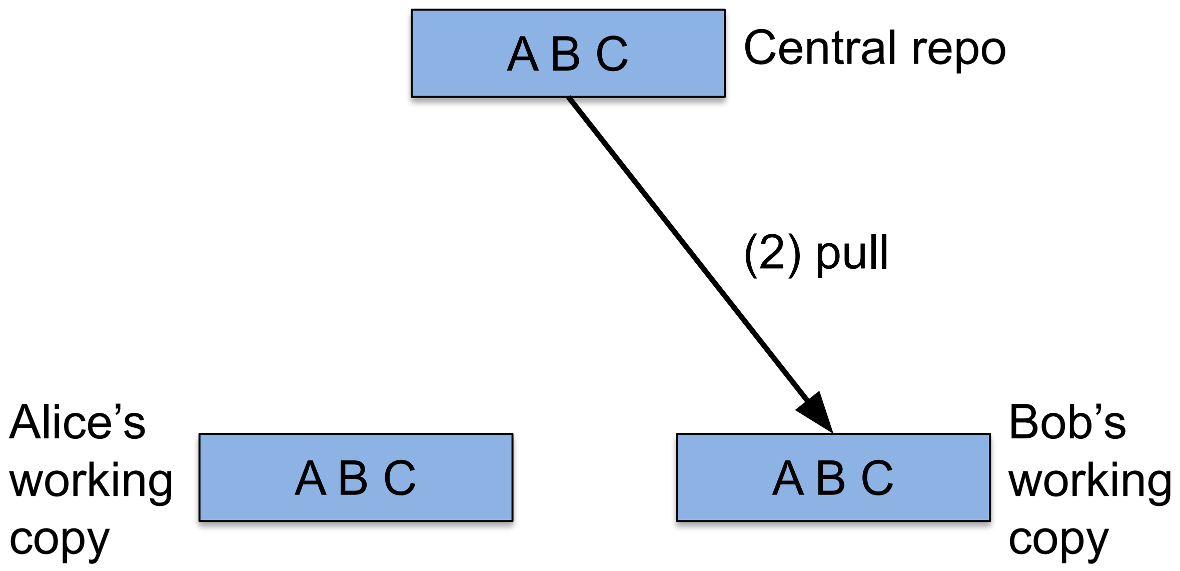
Merging VCSs



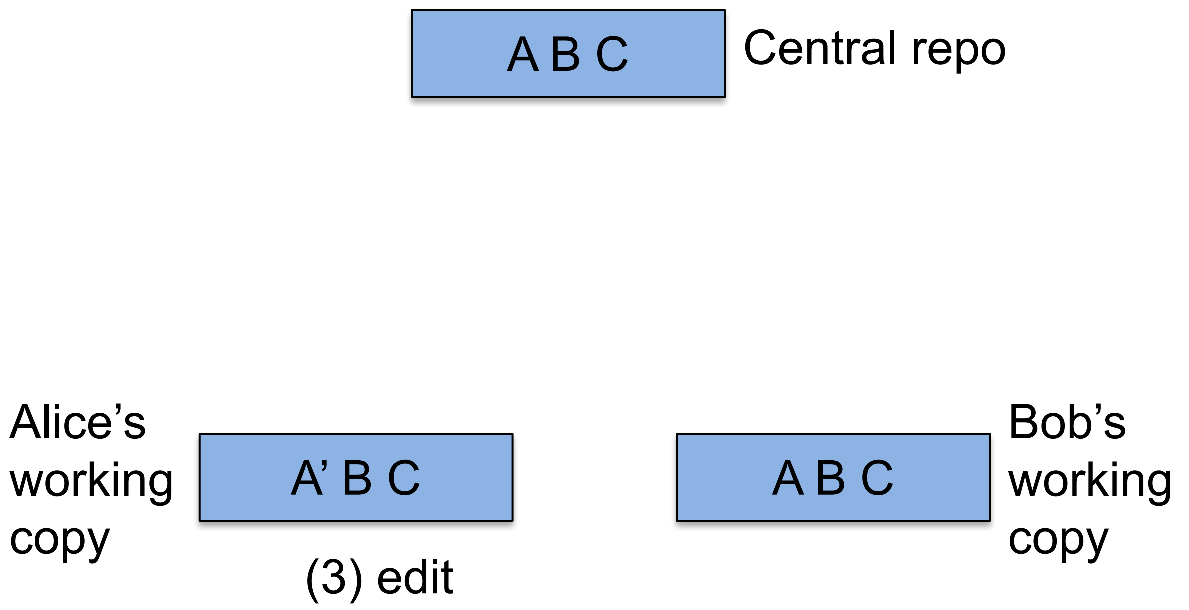
Merging VCSs



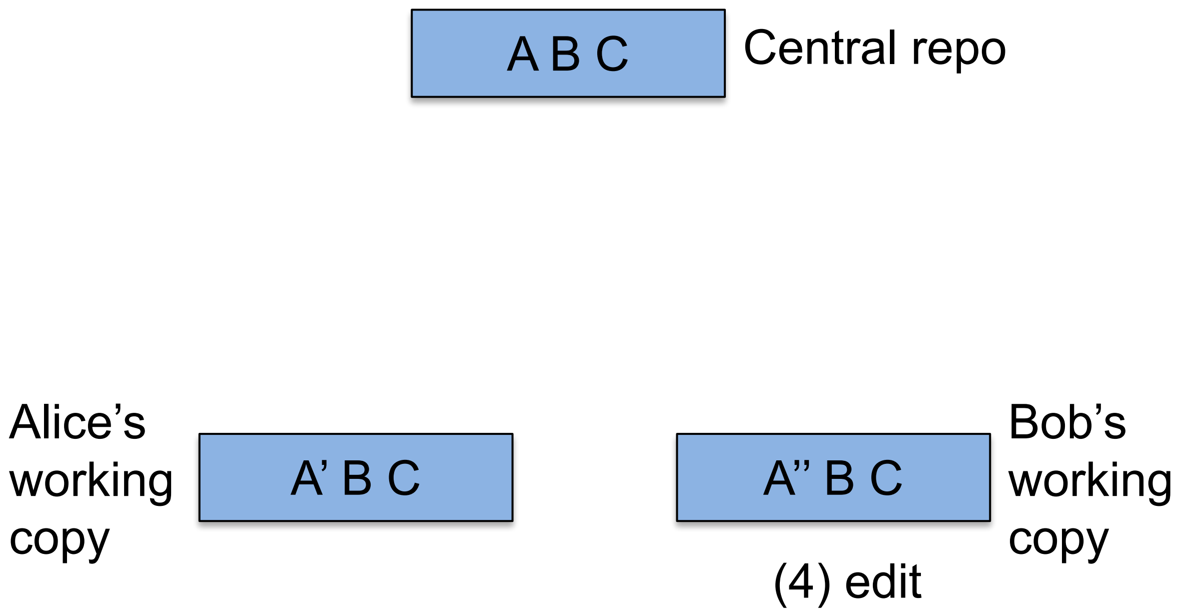
Merging VCSs



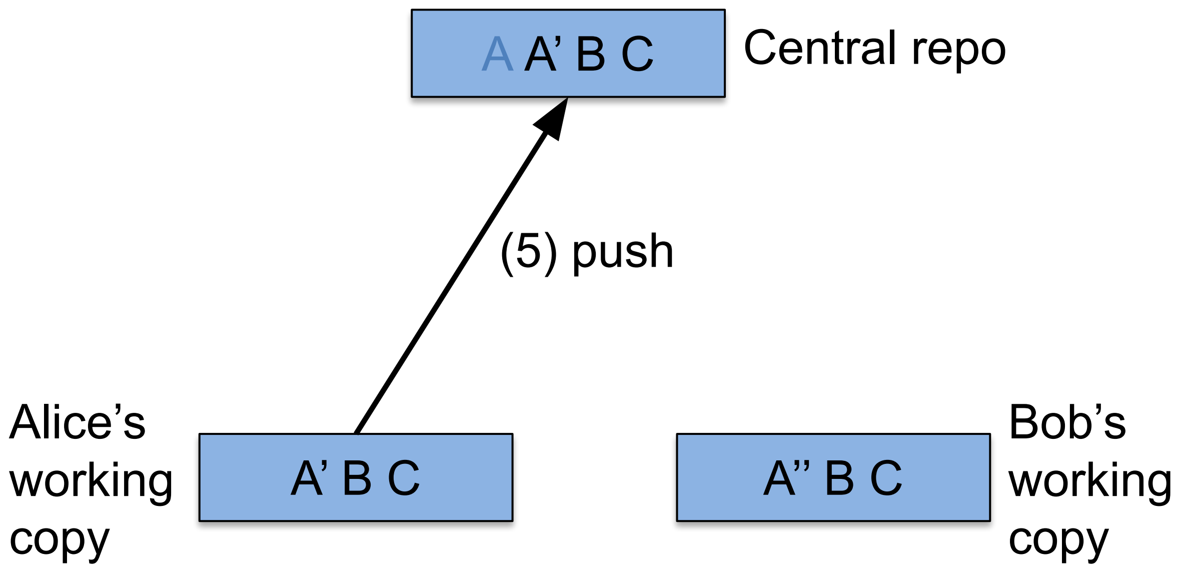
Merging VCSs



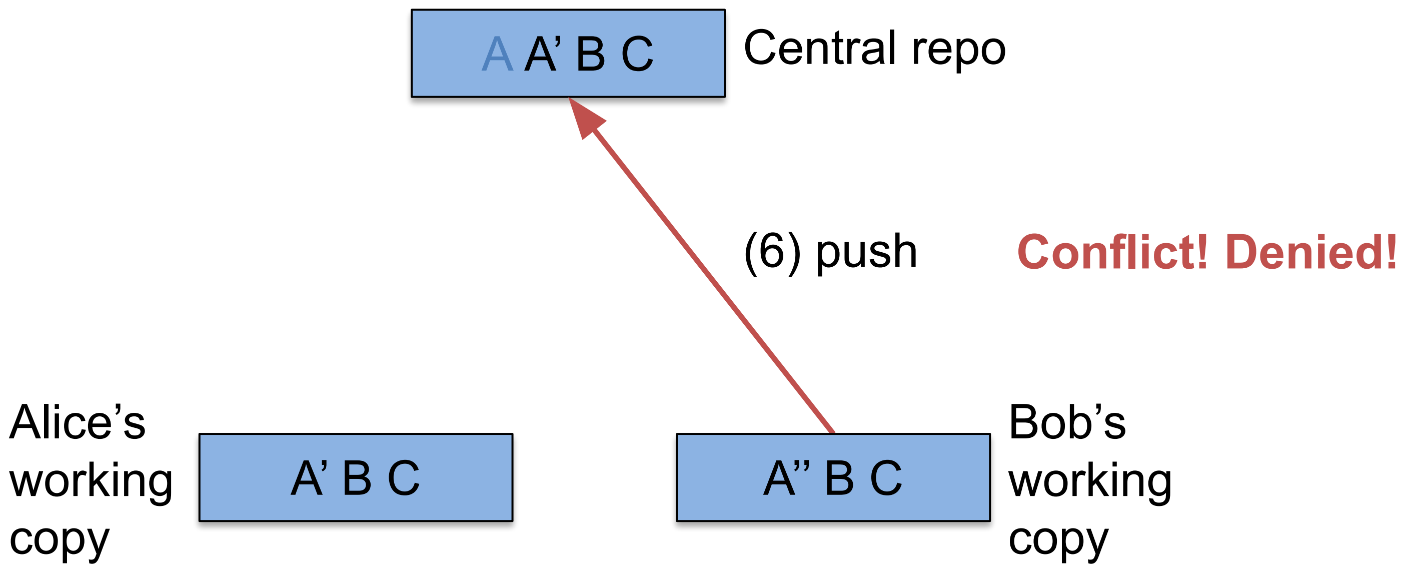
Merging VCSs



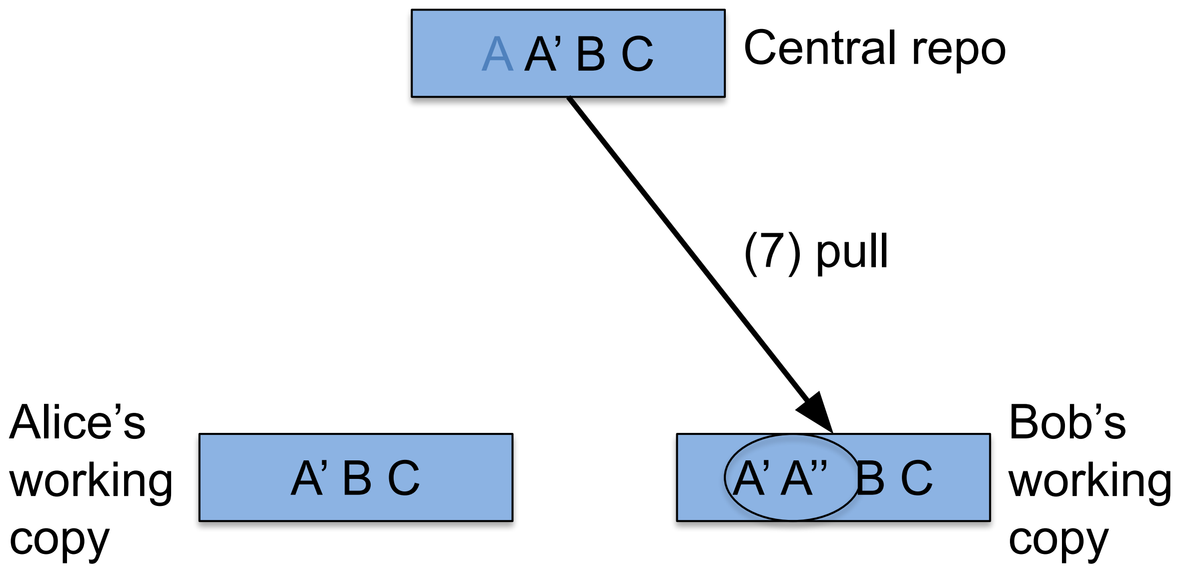
Merging VCSs



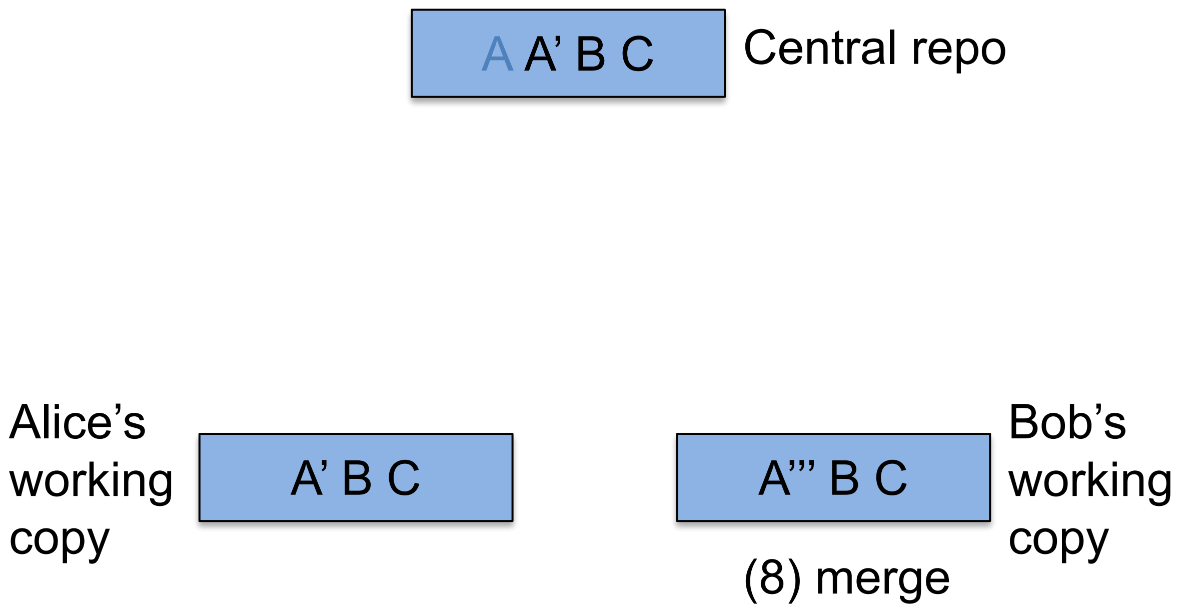
Merging VCSs



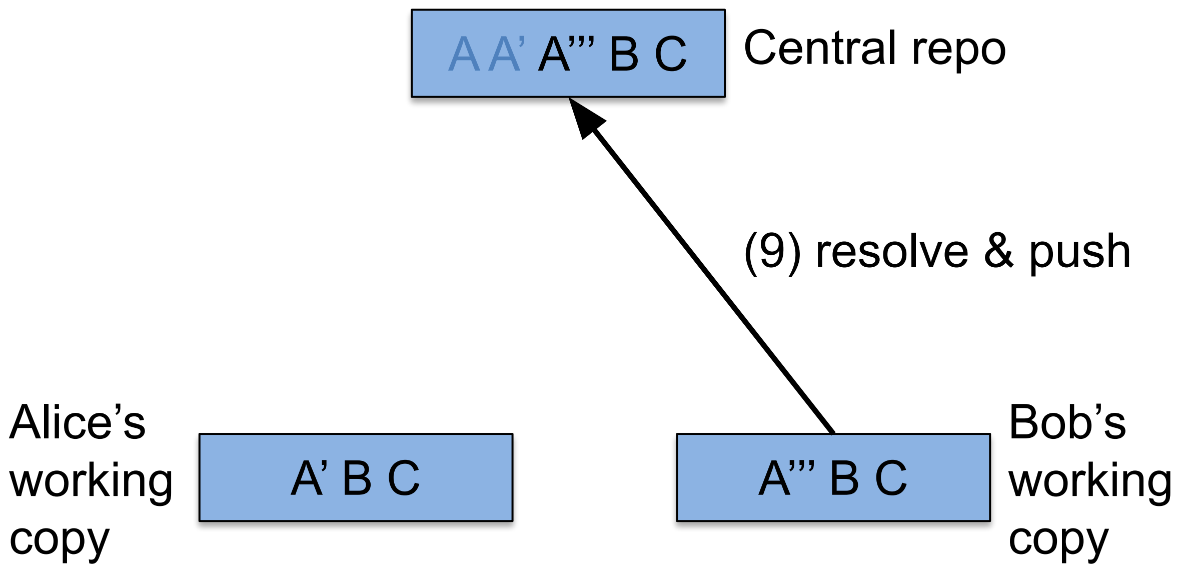
Merging VCSs



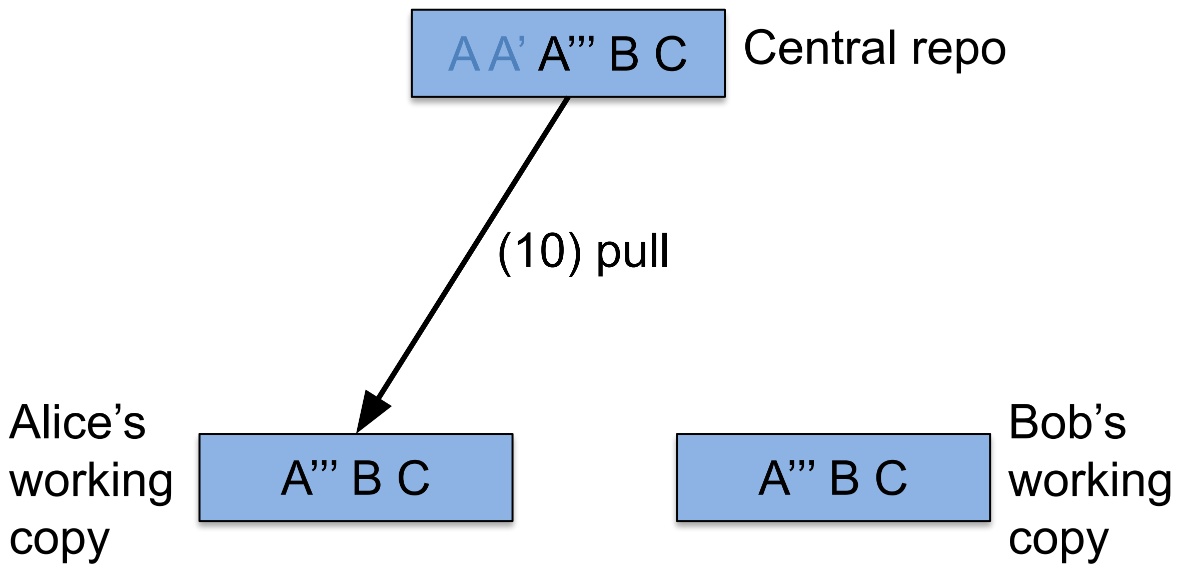
Merging VCSs



Merging VCSs



Merging VCSs



Merging VCSs

- **Problems**:
  - Tough to work offline
    - Offline => no version history
  - Single point of failure
    - Server down => no version history
    - Central repo corrupted (and no backup) => version history lost
  - Constrained workflow
    - (More soon in this lecture)

Agenda

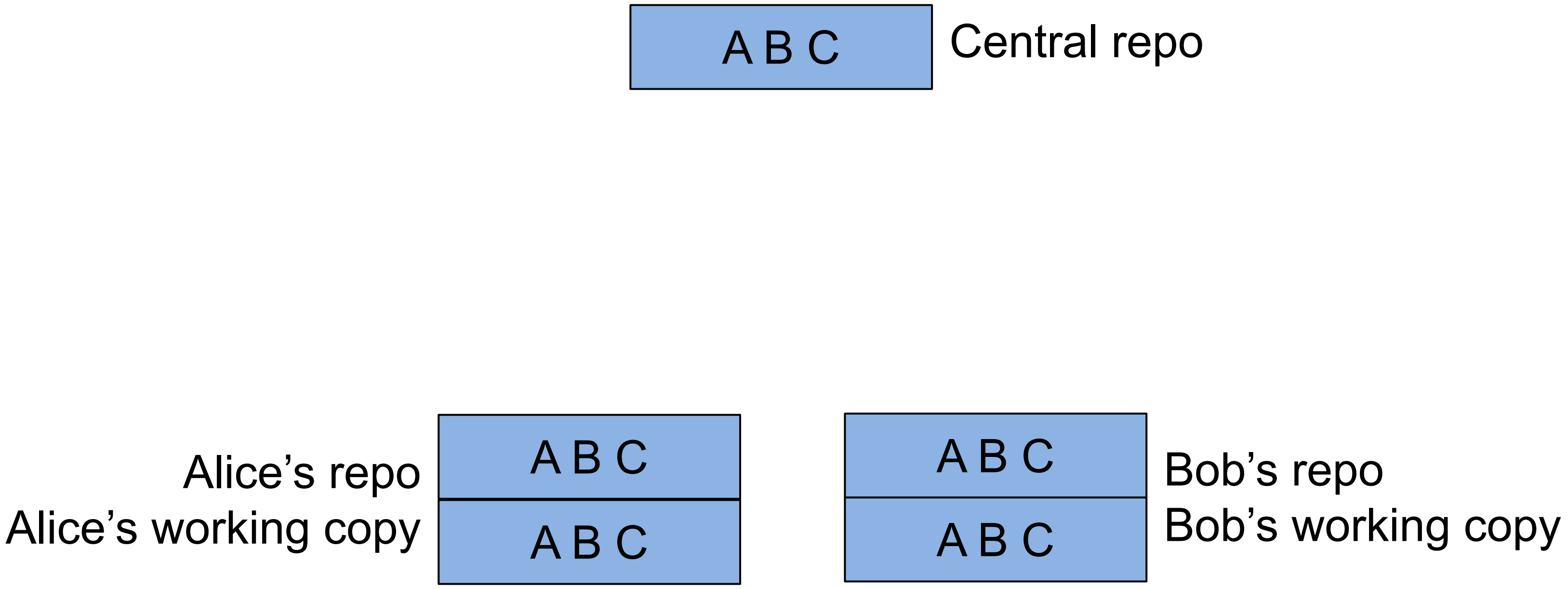
- Personal VCSs
- Centralized VCSs
- Locking VCSs
- Merging VCSs
- **Distributed VCSs**

Distributed VCSs

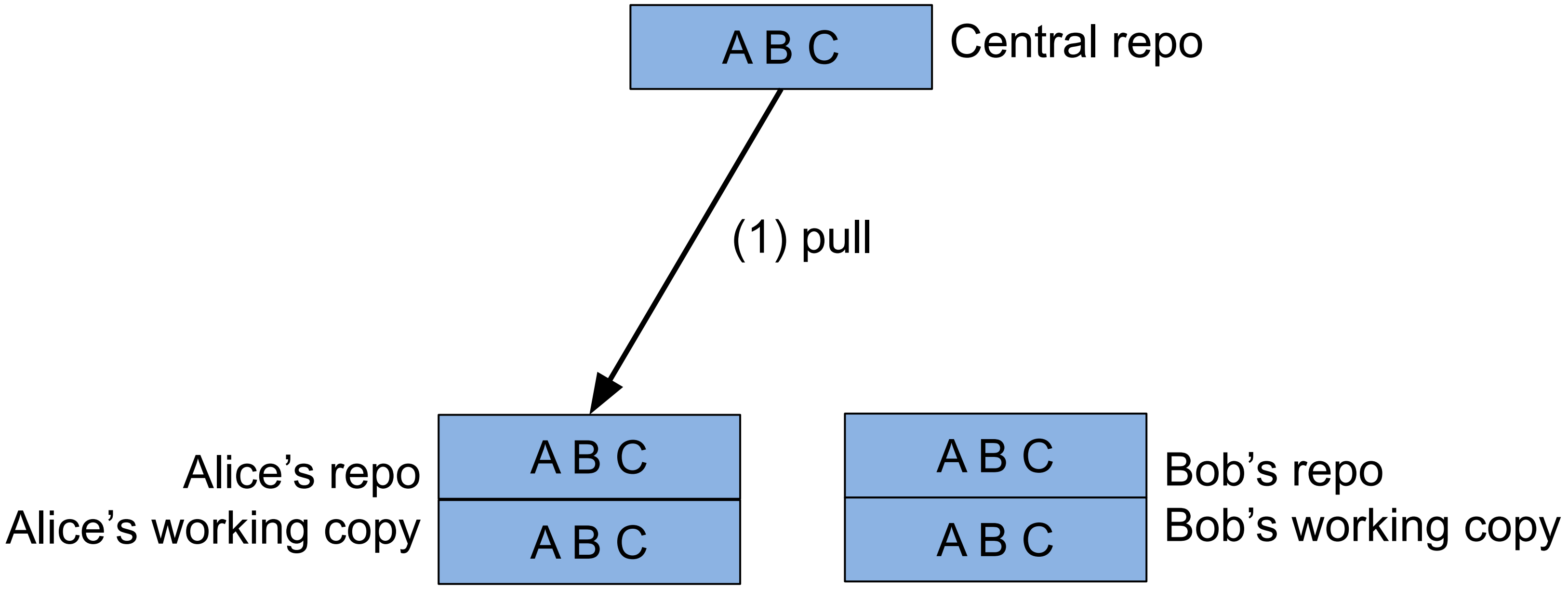
- **Solution**: distributed VCS
  - Version control + ***distributed repos*** + merging
  - Popular systems: ***Git****, ***Mercurial***
- **Example**:

*Git is the dominant VCS

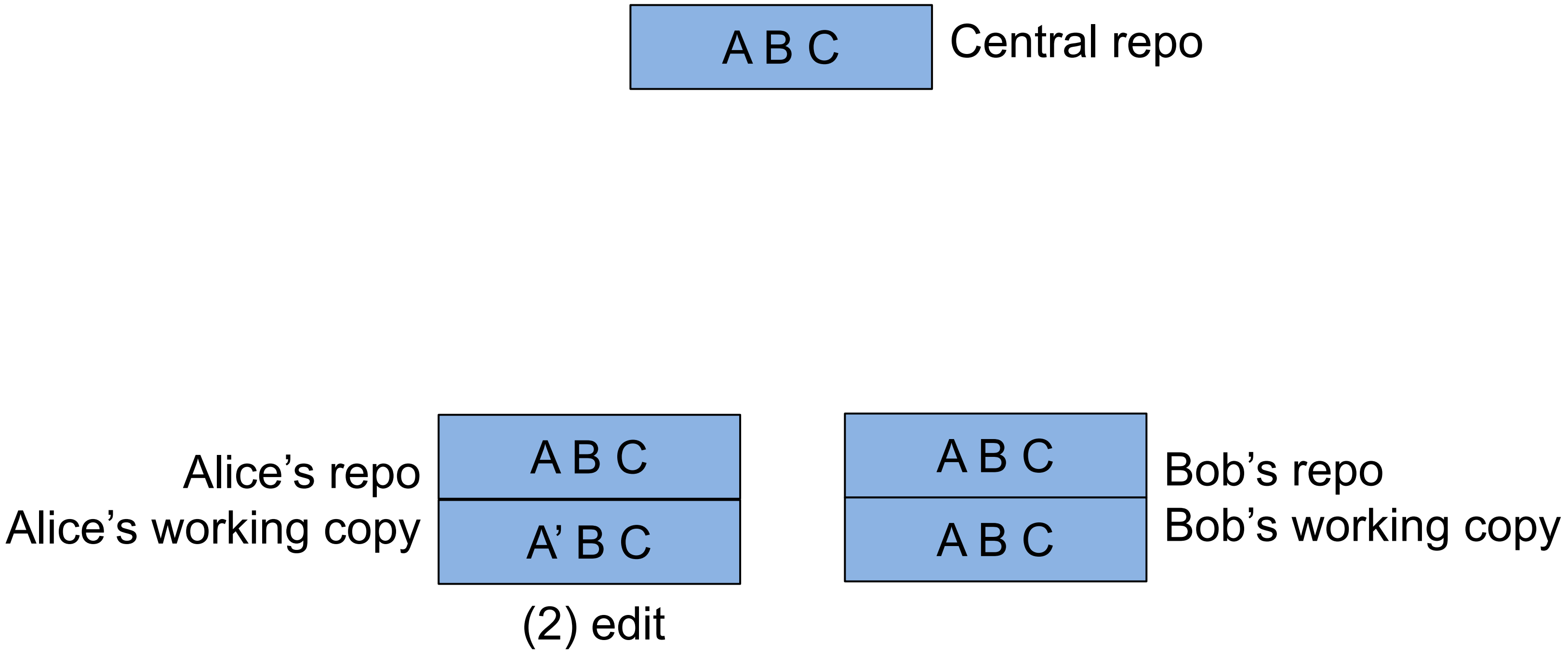
Distributed VCSs



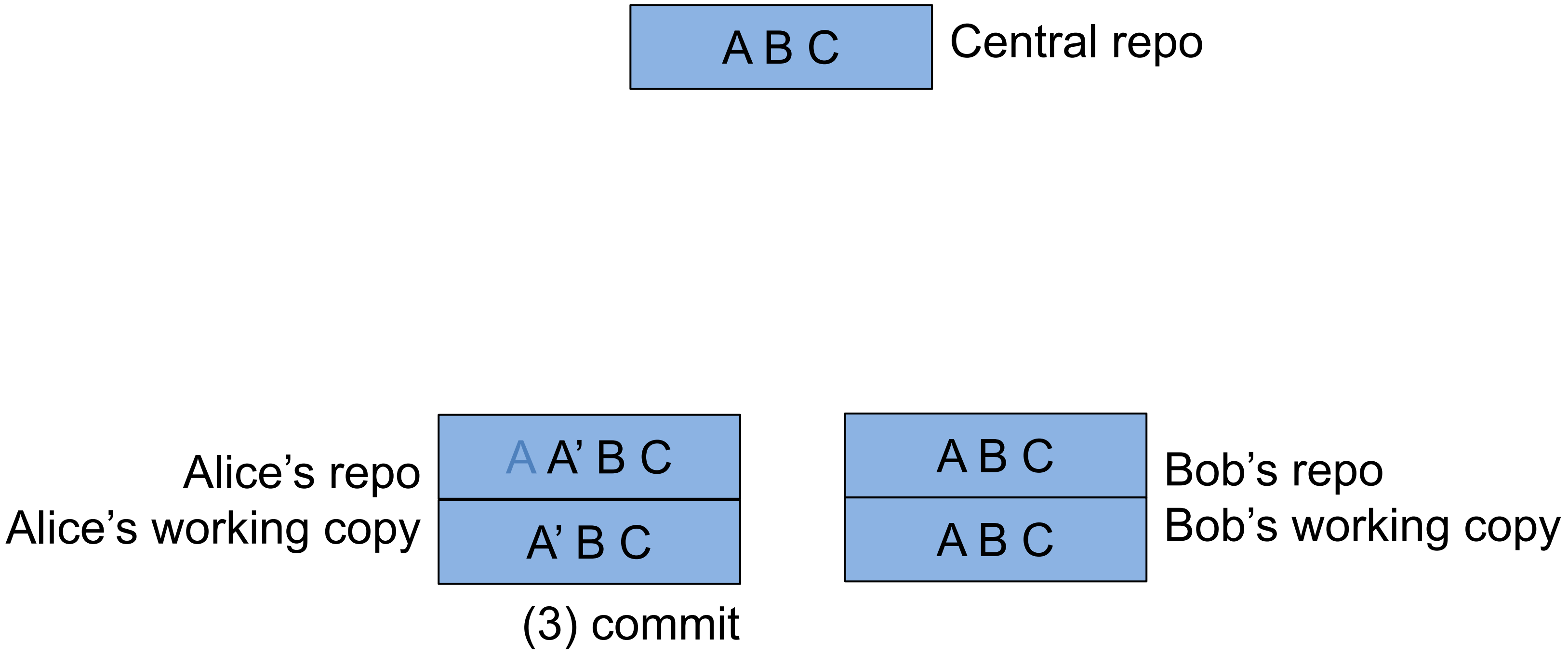
Distributed VCSs



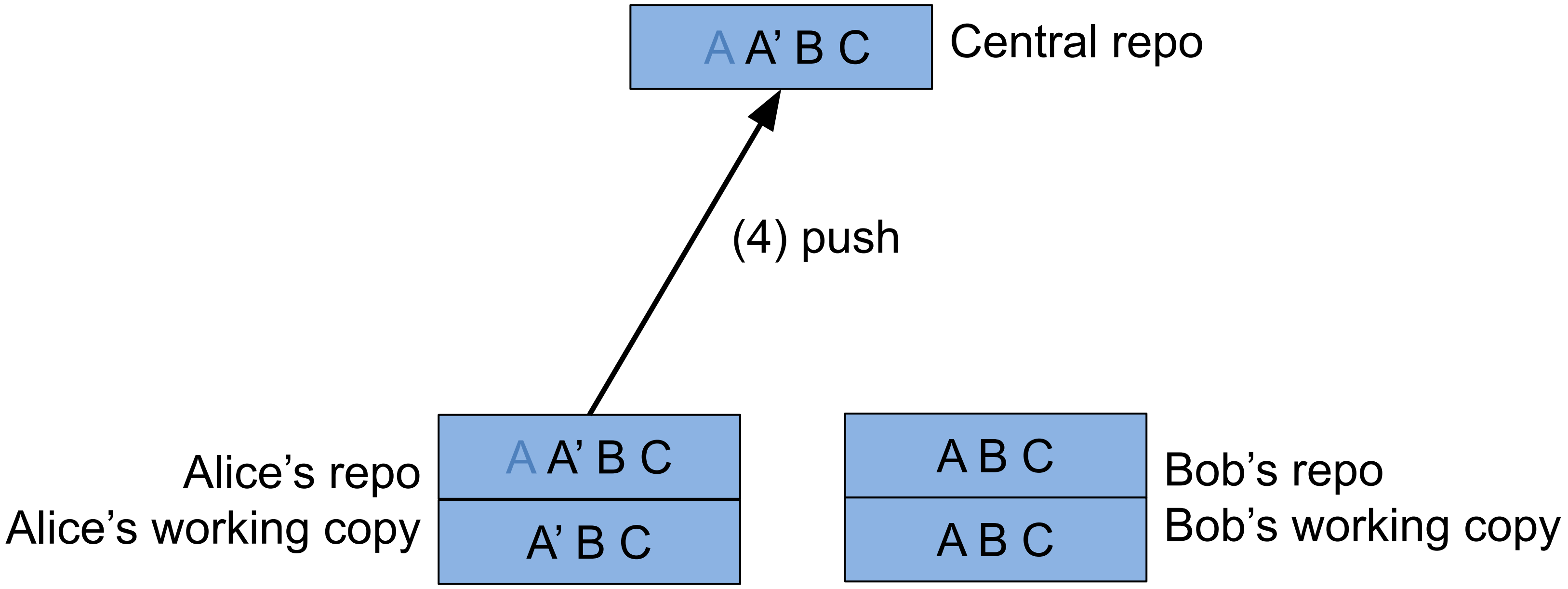
Distributed VCSs



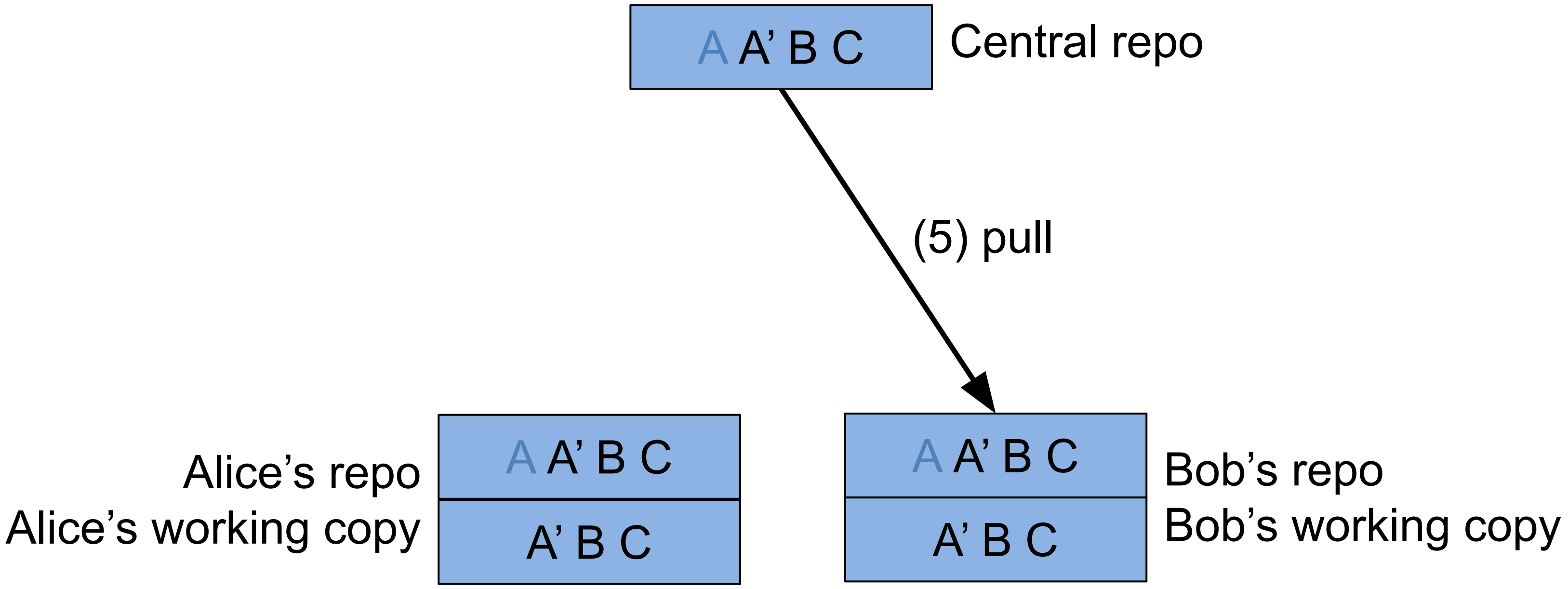
Distributed VCSs



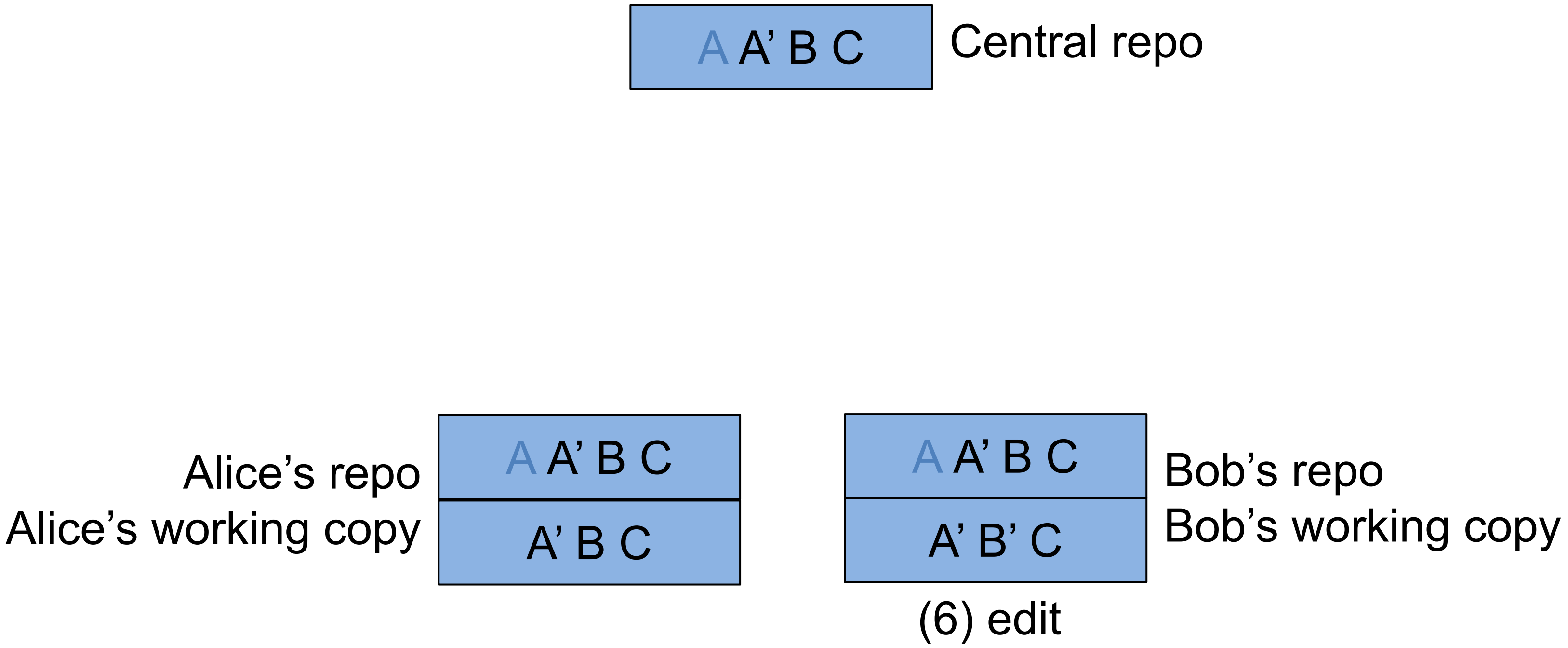
Distributed VCSs



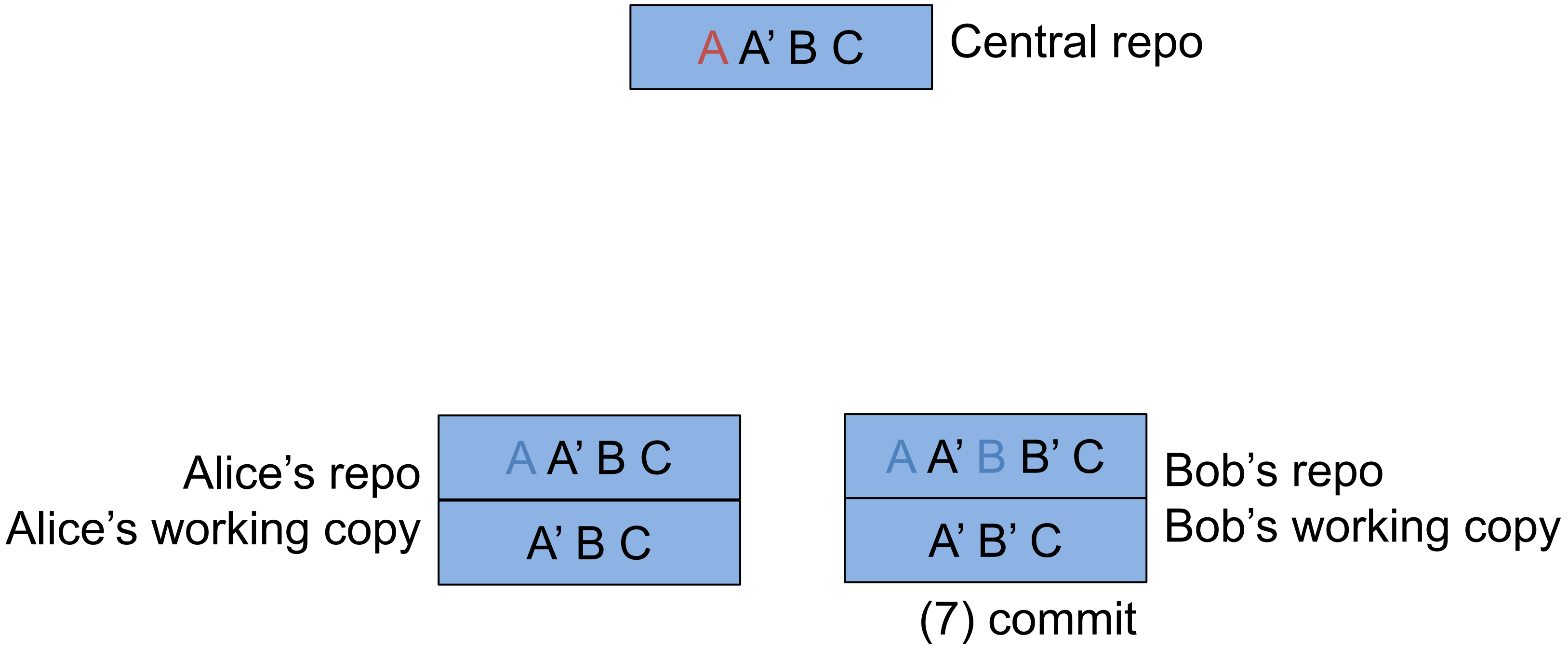
Distributed VCSs



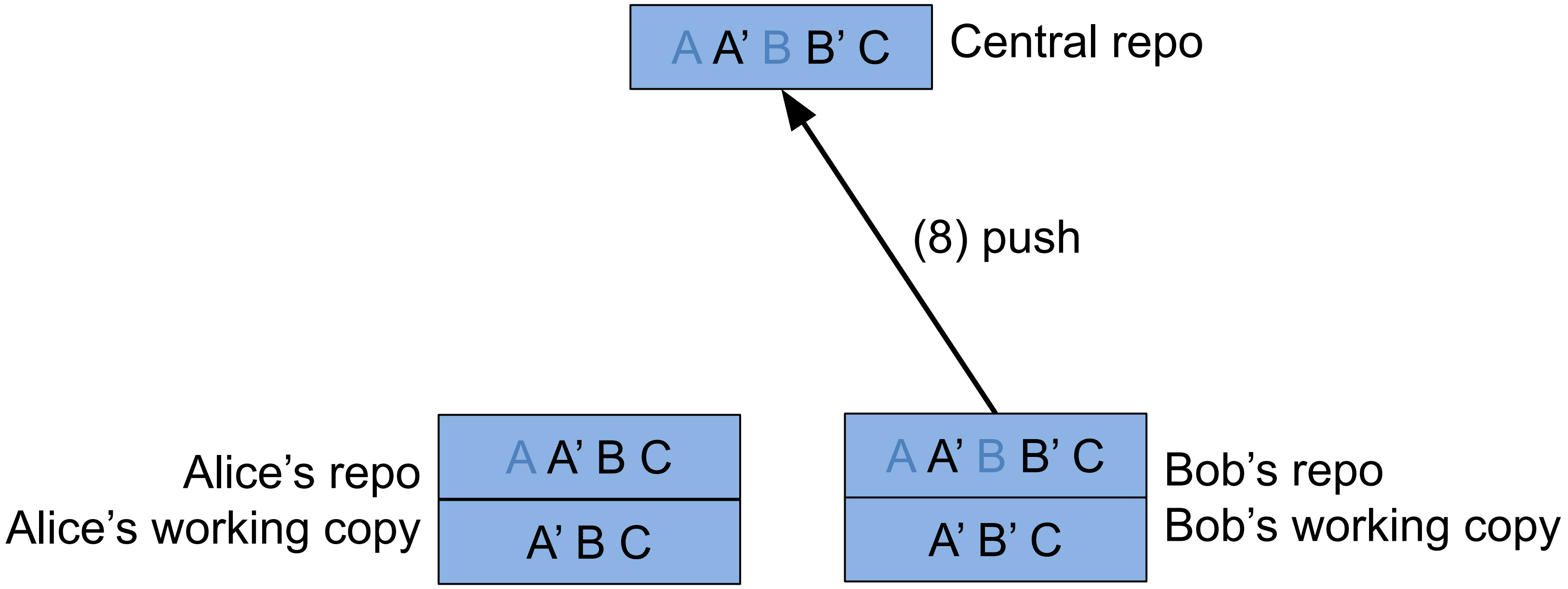
Distributed VCSs



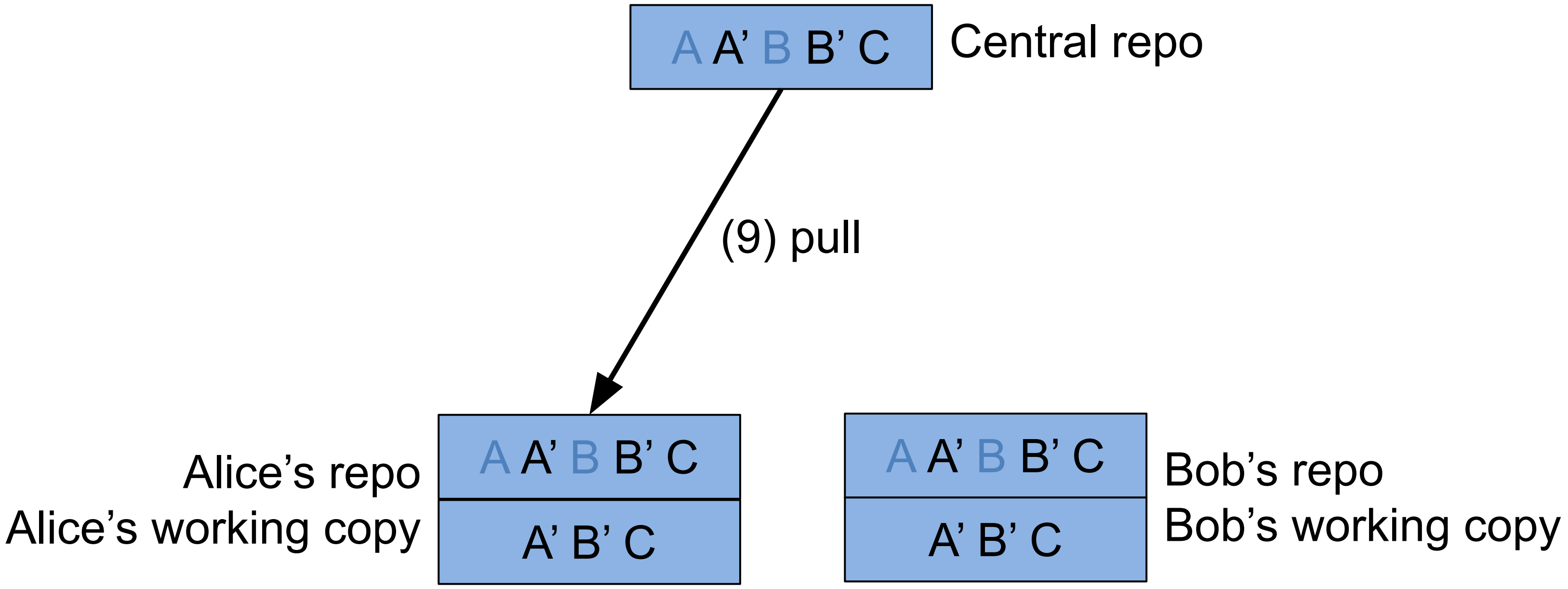
Distributed VCSs



Distributed VCSs



Distributed VCSs

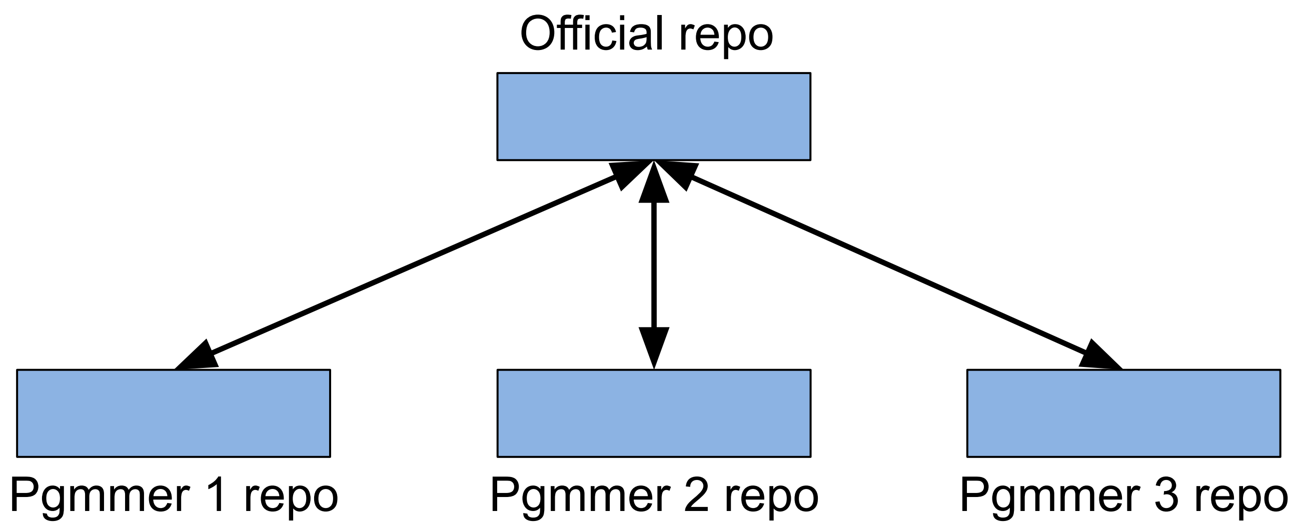


Distributed VCS Topologies

- Central repository is not special
  - Programmers can **pull from** *any* repo
  - Programmers can **push to** *any* repo
  - Permits variety of topologies…

Distributed VCS Topologies

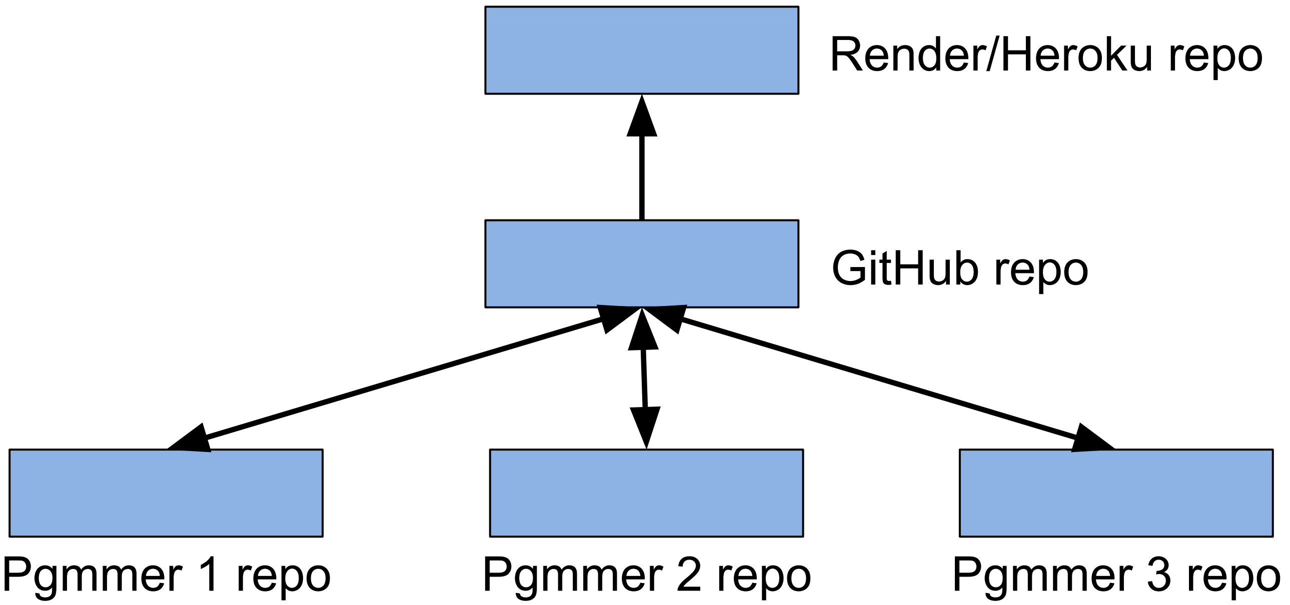
Centralized Topology



Each programmer merges into shared repository
Probably sufficient for COS 333 assignments
Maybe sufficient for COS 333 project

Distributed VCS Topologies

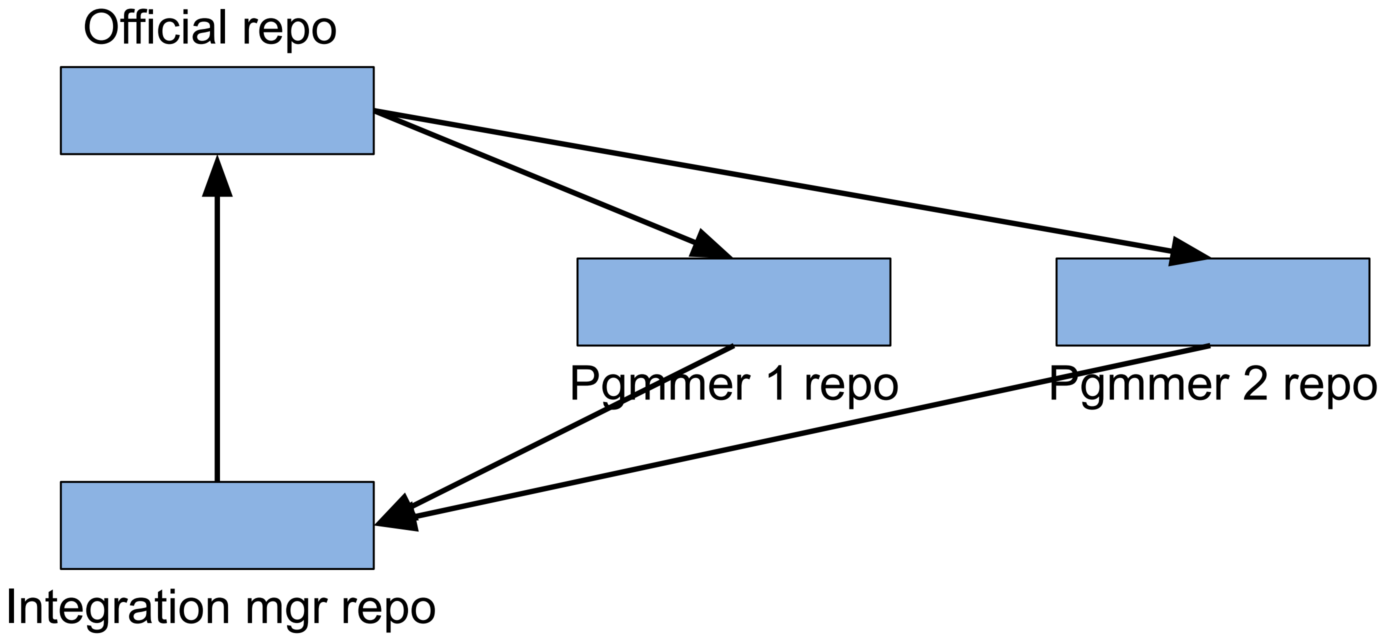
Centralized Topology



Most common for COS 333 projects

Distributed VCS Topologies

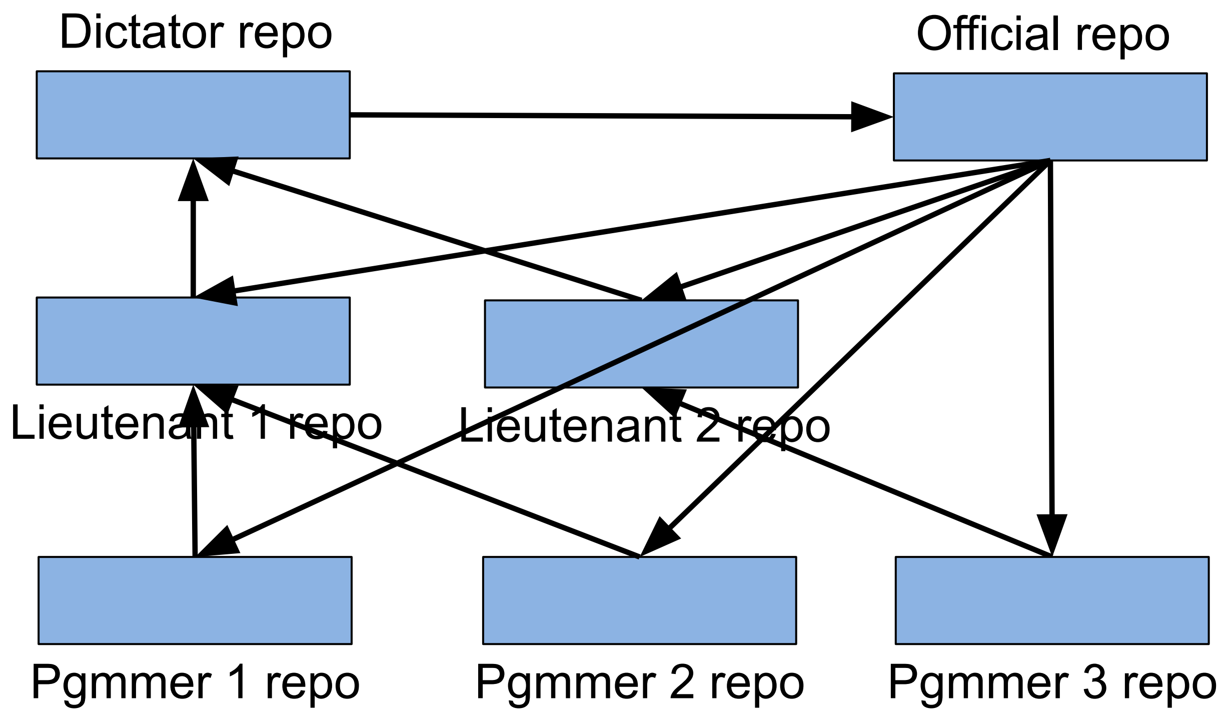
Integration Mgr Topology



Each programmer pushes to integration manager
Integration manager does all inter-programmer merges
Integration manager pushes to official repo

Distributed VCS Topologies

Dictator & Lieutenants Topology



Used by the Linux project
Dictator = Linus Torvalds

Distributed VCS Topologies

- Dictator & lieutenants topology
  - Each programmer:
    - Pushes to one lieutenant
  - Each lieutenant:
    - Manages one subsystem
    - Pushes to dictator
  - Dictator:
    - Manages the project as a whole
    - Pushes to official repo

Summary

- We have covered:
  - Version control systems (VCSs)
    - As mechanisms for…
    - Maintaining file versions
    - Safely sharing files with teammates