RCVS:
Remote extension of Concurrent Versions System

Terry Hung

Stanford Linear Accelerator Center

abstract

Over the last few years there has been much interest in Concurrent Versions System(CVS) for code management in both adademia and industry. The nature of large geographic regional collaboration in high energy physics calls for the working of CVS across wide area network. The tools presented in this paper provides remote access to the source code maintained by CVS.

Table of Contents

- **1. Background:**
  1.1 code management in general
  1.1.1 SCCS
  1.1.2 RCS
  1.1.3 general features of RCS/SCCS
  1.1.4 concerns over RCS/SCCS
  1.2 CVS
  1.2.1 remote extension (RCVS)

- **2. Concerns:**
  2.1 disk space
  2.2 rdist

- **3. Features:**
  3.1 improvement to CVS in local mode
  3.2 features in RCVS

- **4. Program Structure:**
  4.1 program structure on client
  4.2 program structure on server

- **5. Mechanism of RCVS:**
  5.1 Messaging between RCVS server and client
  5.2 locking mechanism
  5.3 cvs co
  5.3.1 rcvs co
  5.4 cvs commit
  5.4.1 rcvs commit

5.5 cvs add
5.5.1 rcvs add
5.6 cvs remove
5.6.1 rcvs remove

- **6. Usage:**
  6.1 how to set CVSROOT
  6.2 how to set up repository for multiple (group) users
  6.3 how to set up network connection
    6.3.1 .rhost
    6.3.2 .cshrc

- **7. Summary.**
  7.1 speed consideration
  7.2 conclusion

1. Background

- **1.1 Code Management System on UNIX:**

  There are two systems that are widely known and used on UNIX:

- **1.1.1 SCCS**
  — SCCS (Source Code Control System) is developed by AT&T around 1975.
- **1.1.2 RCS:**
  — RCS (Revision Control System) is developed by Walter Tichy at Purdue University as an improvement on SCCS in early 1980's.
  — maintains source and history in single files (fn,v)
  — stores latest version plus backward differences (fast access)
  — support binary files.
- **1.1.3 General Features of RCS/SCCS:**
  — keeps source, revisions, and history in files.
  — stores differences in revision files.
  — supports revision branching.
  — expands keywords into source. e.g Id

- **1.1.4 Concerns of RCS/SCCS:**
  — serial locking prevents concurrent development.
  — handle files in single directory, can not handle large hierarchy of directories.
  — due to these reasons, additional scripts must be written to manage the project as a whole.

- **1.2 CVS**

  CVS (Concurrent Versions System) is originally developed by Dick Grune at Vrije University, Amsterdam, and later by Brain Berliner and Jeff Polk of Prisma, Inc. (Now at Sun and BSDI).

  — It is a front end to RCS system, inherit its simplicity
  — can handle hierarchy of directories, not just single directory
  — support for tracking third-party vendor source distribution while maintaining the local modifications.
  — a software release can be symbolically tagged.

- **1.2.1 RCVS**

  — RCVS (Remote CVS) is the remote extension of CVS. It is developed at SLAC by Terry Hung and Paul Kunz.
  — network transparent real-time access.

2. Concerns of RCVS

- **2.1 Disk Space:**

  For RCVS with single user, there is 100% overhead in turns of disk space for having a clone area on local host. As the number of user increases to 5, the overhead is reduced to 20%.

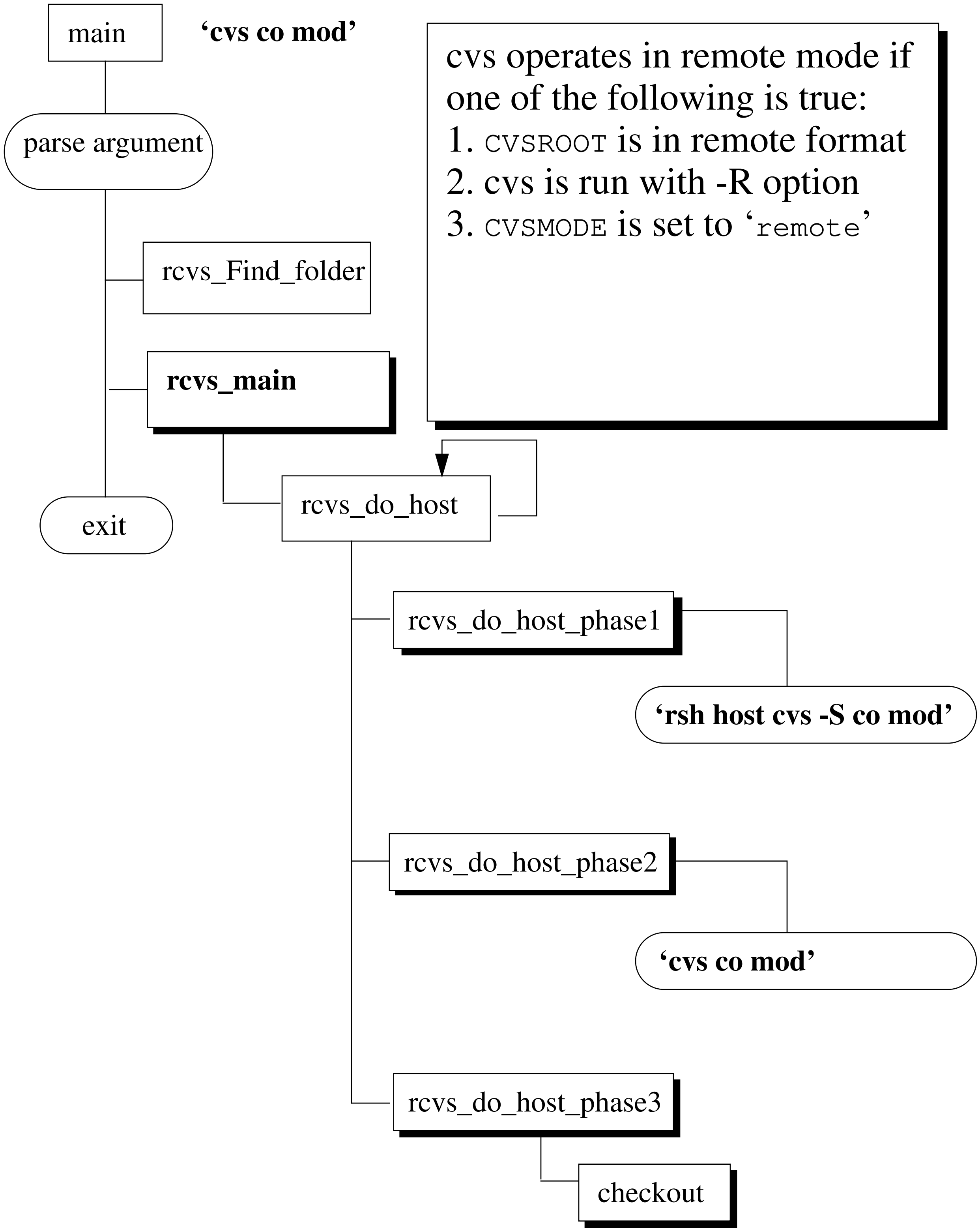
- **2.2 rdist:**
  - — rdist does not return adaquate error code, thus RCVS has to detect the error by checking the error messages. Add to the problem is the fact that different version of **rdist** generate different error messages.

- **2.3 USC's rdist:**
  - — The USC version of rdist generates more fuzzy warning messages than standard UNIX's version. And it does not report special situation properly, e.g. disk full condition.

3. Features

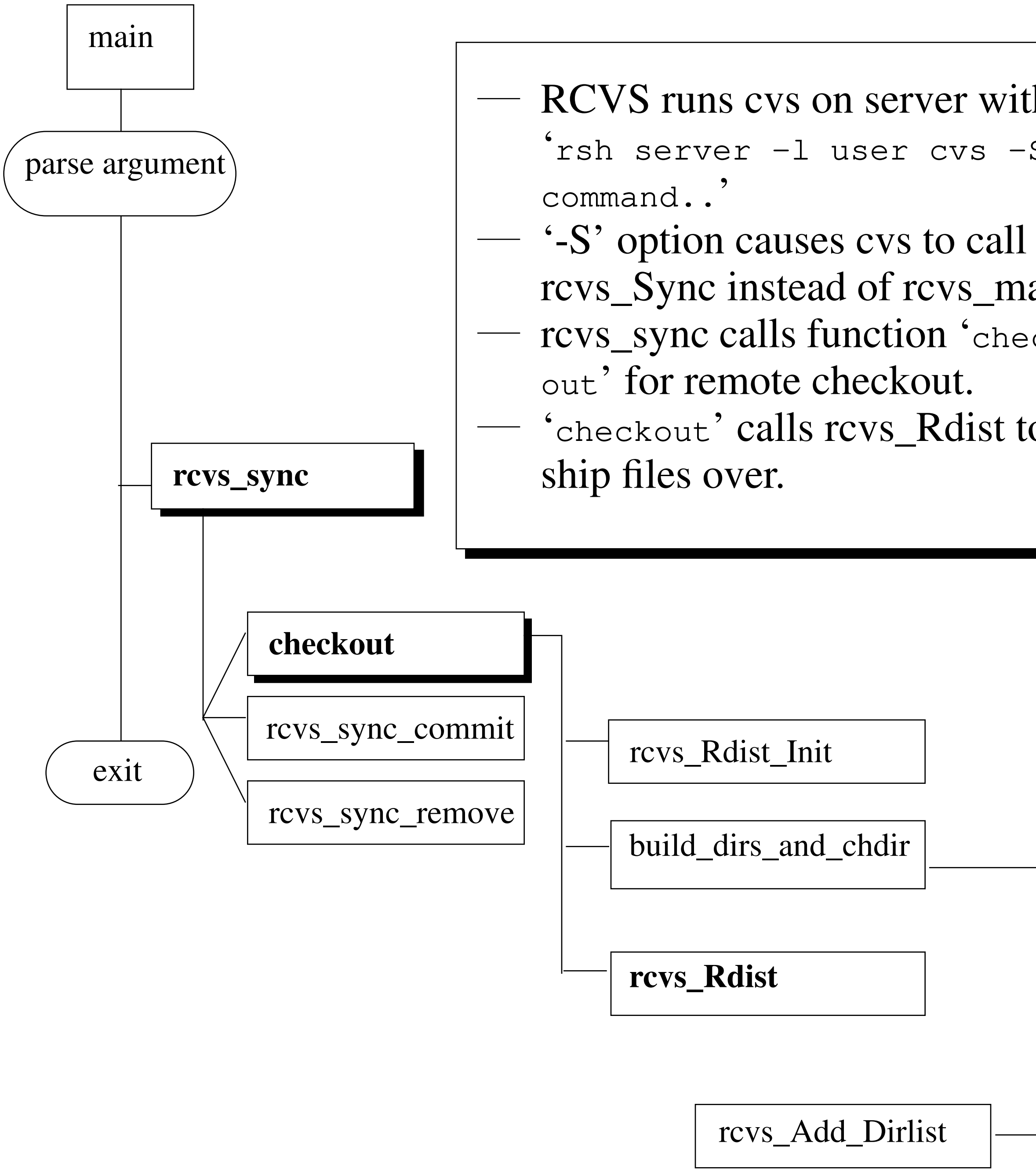
- **3.1 Improvements to local CVS:**
  - — provides -c option that enable user to change repository of local folder.
  - — check against local operation on remote folder.
- **3.2 features in remote CVS:**
  - — supports module concept with "file mode".
  - — memorizes the remote repository information to automate commit, update, etc.
  - — allows combination of phases to provide flexibility.
  - — provides -C option that enable user to change repository
  - — check against local operation on remote folder, and vice versa.

4.1 Program Structure on Client's Side



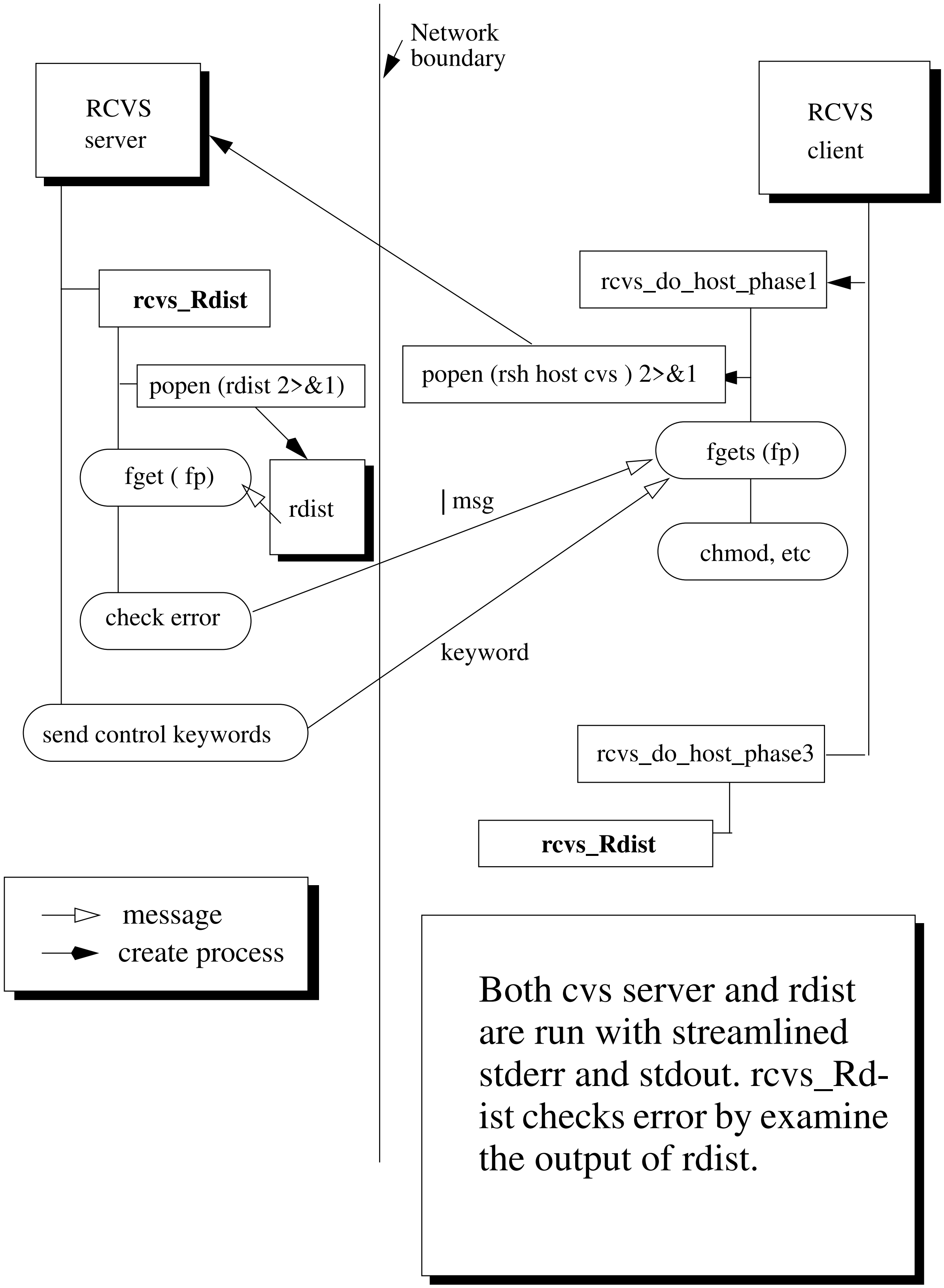
4.2 Program Structure on Server Side

‘cvs -S -d user@client:/clone:/repo co mod’



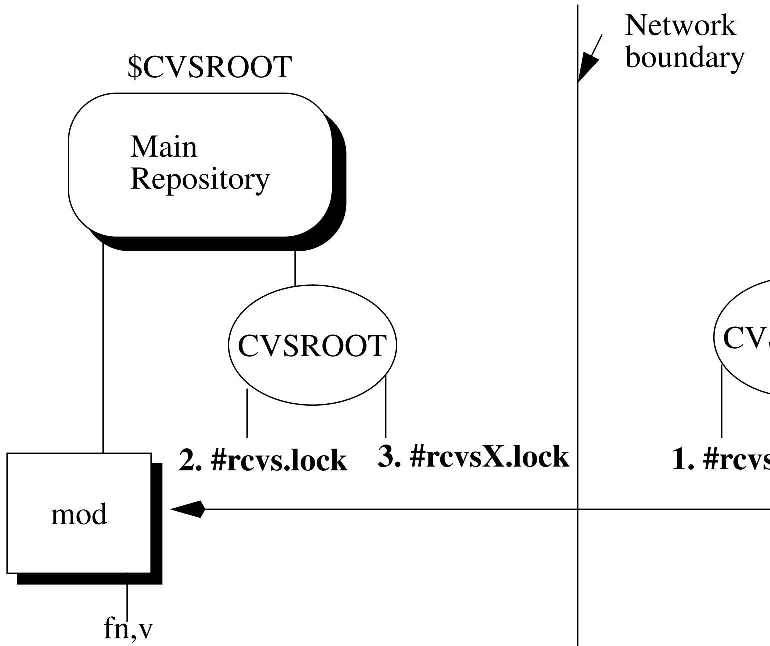
- RCVS runs cvs on server with ‘`rsh server -l user cvs -S command..`’
- ‘-S’ option causes cvs to call rcvs_Sync instead of rcvs_main.
- rcvs_sync calls function ‘`check-out`’ for remote checkout.
- ‘`checkout`’ calls rcvs_Rdist to ship files over.

5.1 Messaging between RCVS server and client



Both cvs server and rdist are run with streamlined stderr and stdout. rcvs_Rdist checks error by examine the output of rdist.

5.2 RCVS Locking Mechanism



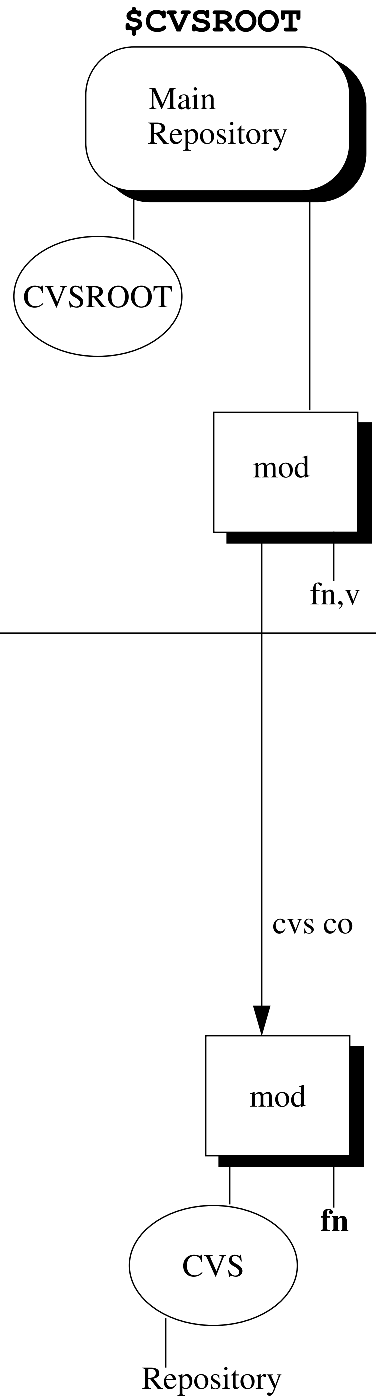
— There are three RCVS locks for `cvs commit`:
1. `#rcvs.lock` on client
2. `#rcvs.lock` on server
3. `#rcvsX.lock` on server

— `#rcvs.lock` freezes the system during remote '`cvs commit`'.

— `#rcvsX.lock` freezes the system against rcvs during local '`cvs commit`'.

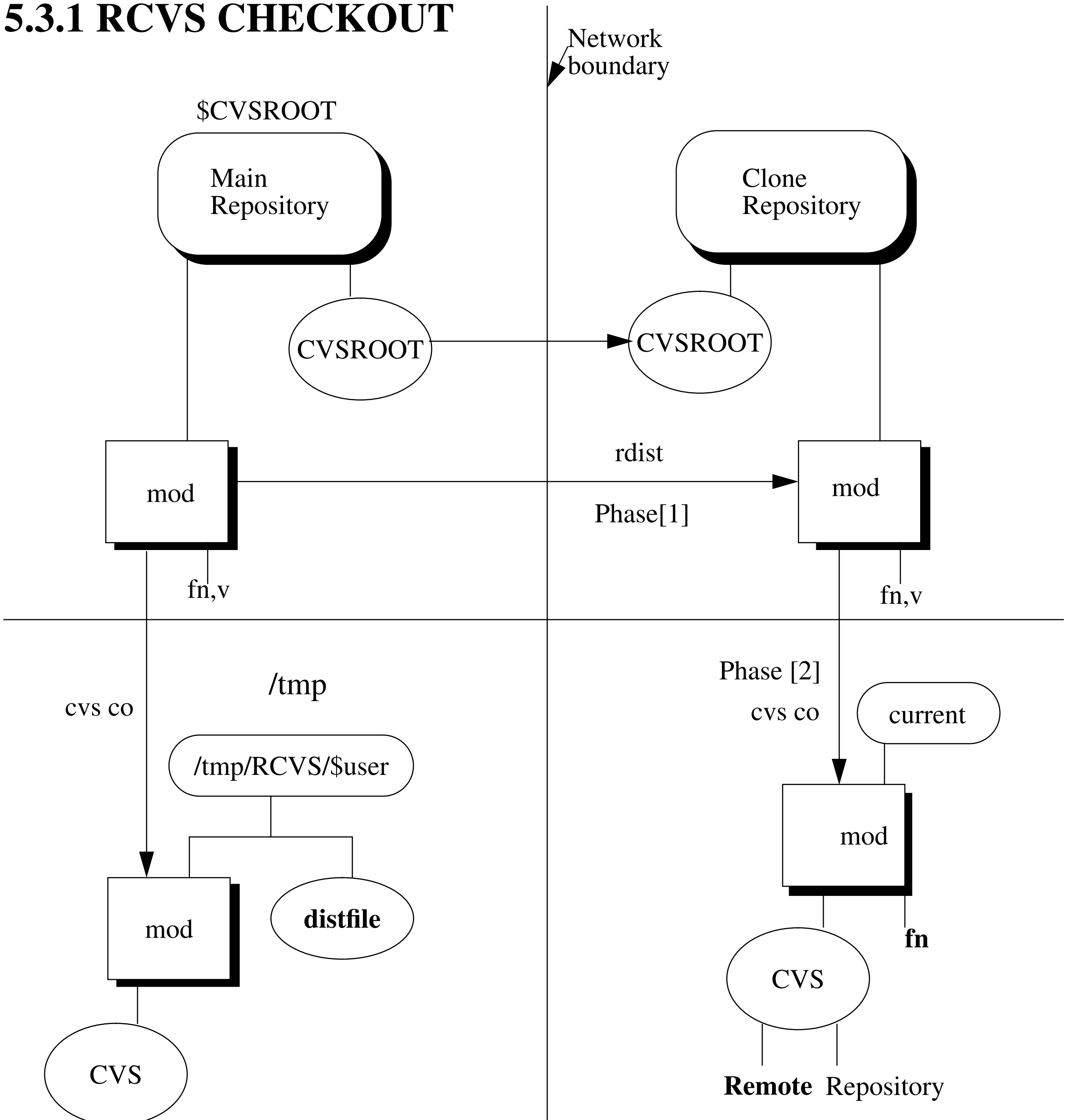
— Uses '-U' option to remove these locks.

5.3 CVS CHECKOUT



- `setenv CVSROOT /usr/repo`
  `cvs checkout mod`
- `CVSROOT` must be set to local format to checkout a module from a local main repository

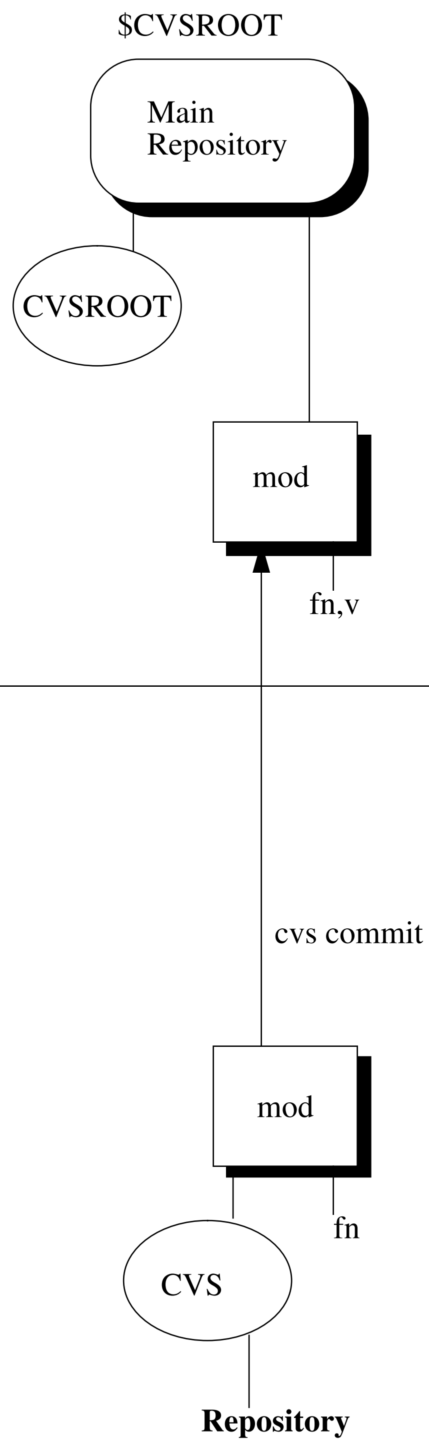
5.3.1 RCVS CHECKOUT



An empty directory tree and a `distfile` is created in `/tmp` so that cvs on server can traverse the tree and derive list of directories/files that is needed by a module.

A `CVS/Remote` file is created in working directory to record the remote repository. This file is the symbol of a remote folder.

5.4 CVS COMMIT



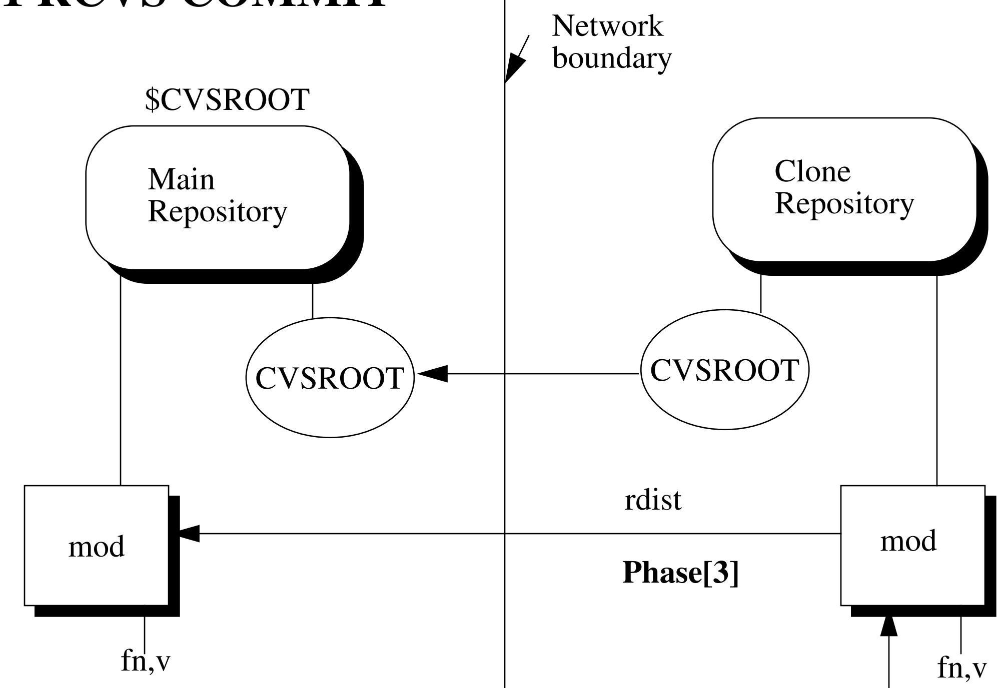

```
cvs commit
cvs commit fn
```

cvs commits file back to main repository according to `CVS/Repository` of the working directory.

CVS/Repository:

```
/usr/repo/mod
```

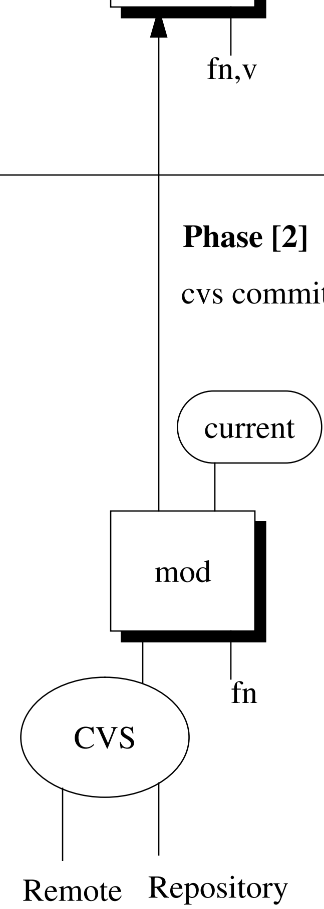
5.4.1 RCVS COMMIT



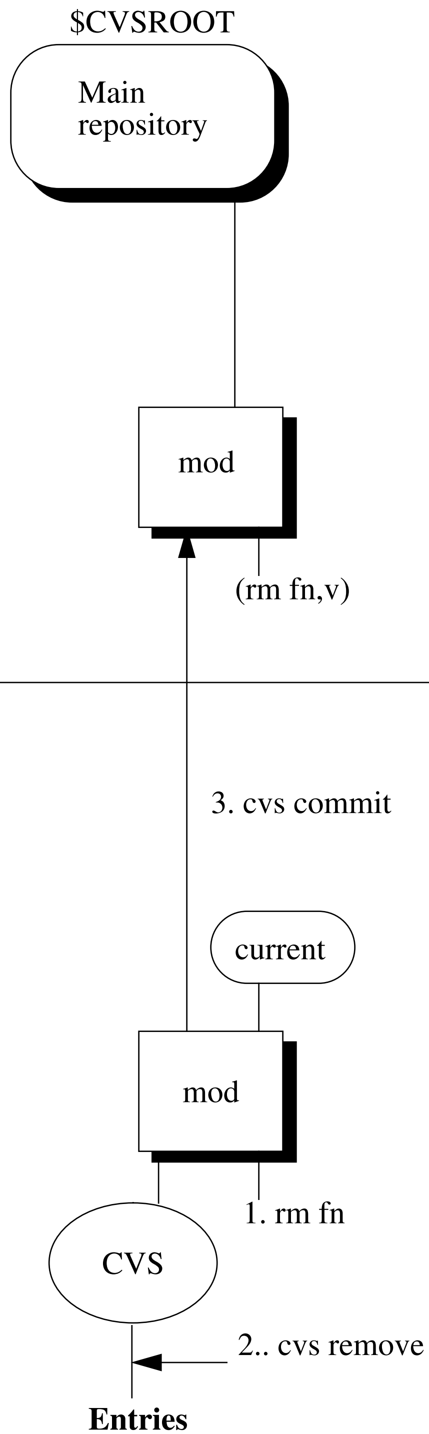

```
cvs commit
cvs commit fn
```

— The phase [2] of ‘`cvs commit`’ merge `fn` into `fn,v` in the clone repository.

— The phase [3] sync clone repository back to main repository.



5.5 CVS REMOVE




```
1. rm fn
2. cvs remove fn
3. cvs commit
```

— 1. a file must be removed first before ‘`cvs remove`’

— 2. ‘`cvs remove`’ removes an entry from `CVS/Entries`

— 3. ‘`cvs commit`’ removes the file from main repository since it’s not in `CVS/Entries` anymore.

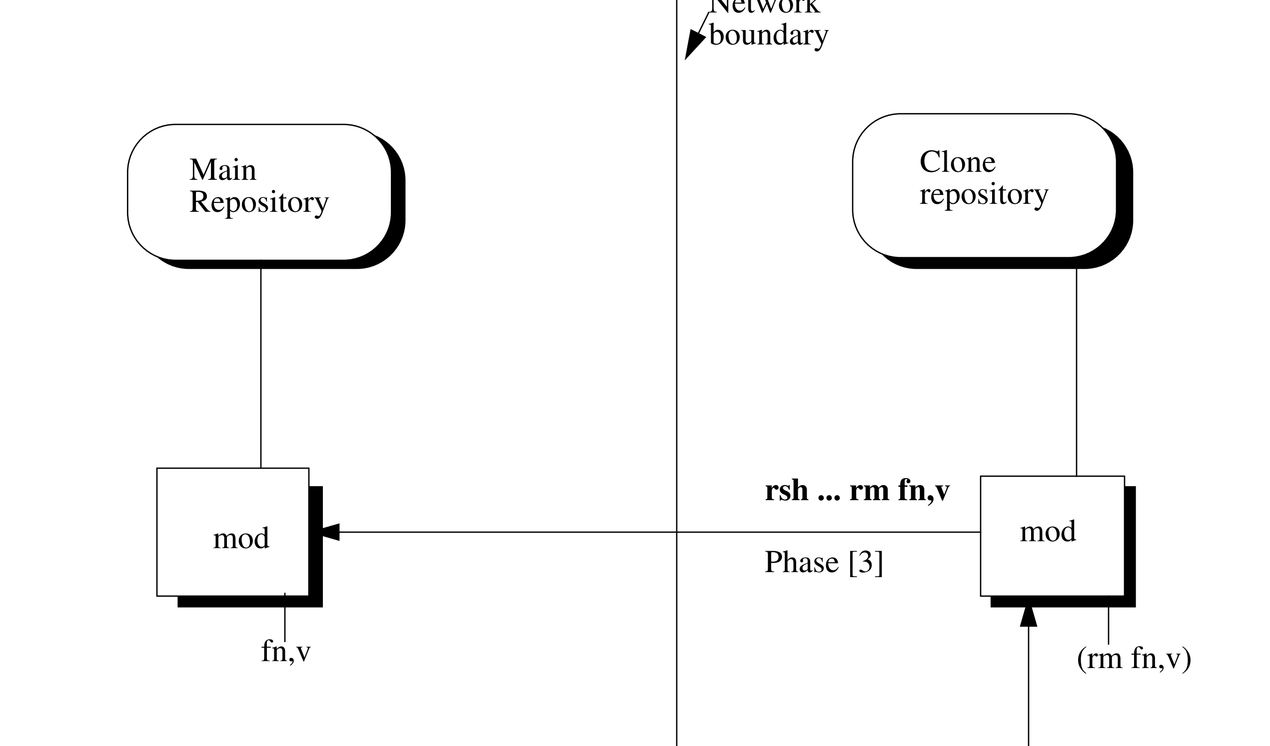
CVS/Entries before `cvs remove`:

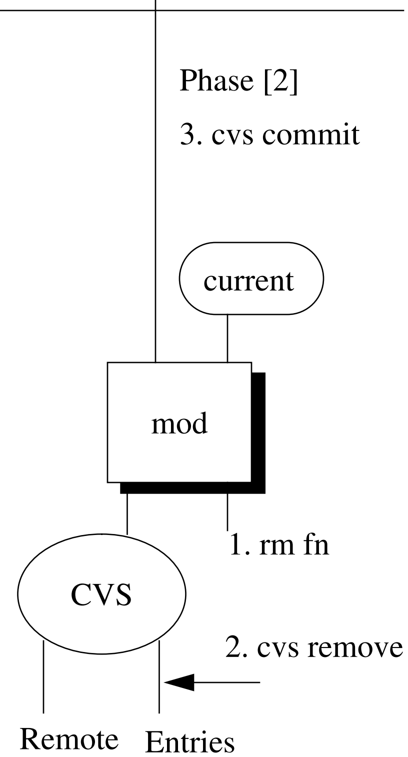
/f1/1.7/date//
/**fn**/1.2/date//

CVS/Entries after `cvs remove`:

/f1/1.2/date//

5.5.1 RCVS REMOVE



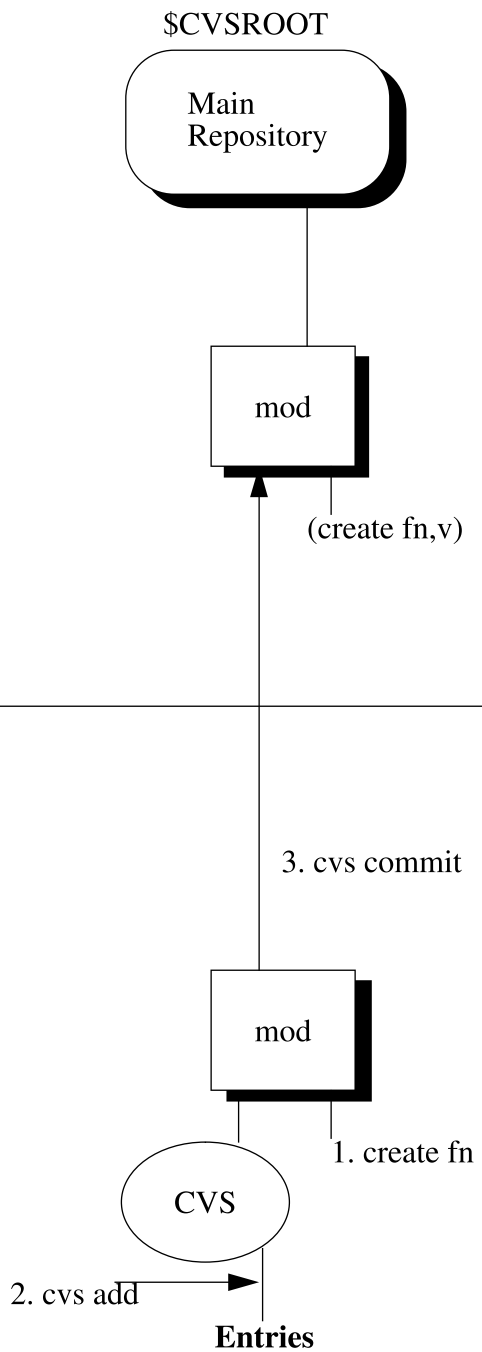


1. `rm fn`
2. `cvs remove fn`
3. `cvs commit fn`

Executing '`cvs remove`' on a remote folder is the same as a local folder. Most of the work is done by '`cvs commit`' afterwards.

Since `fn` does not exist anymore, `rdist` can not be used to sync files in '**file mode**'. a '`rm fn,v`' command is executed via '`rsh`'.

5.6 CVS ADD



1. `emacs tempo`
2. `cvs add tempo`
3. `cvs commit`

— 1. a file must be created in local folder before '`cvs add`'

— 2. '`cvs add`' insert new entry into CVS/Entries.

— 3. '`cvs commit`' actually creates `fn,v` into repository.

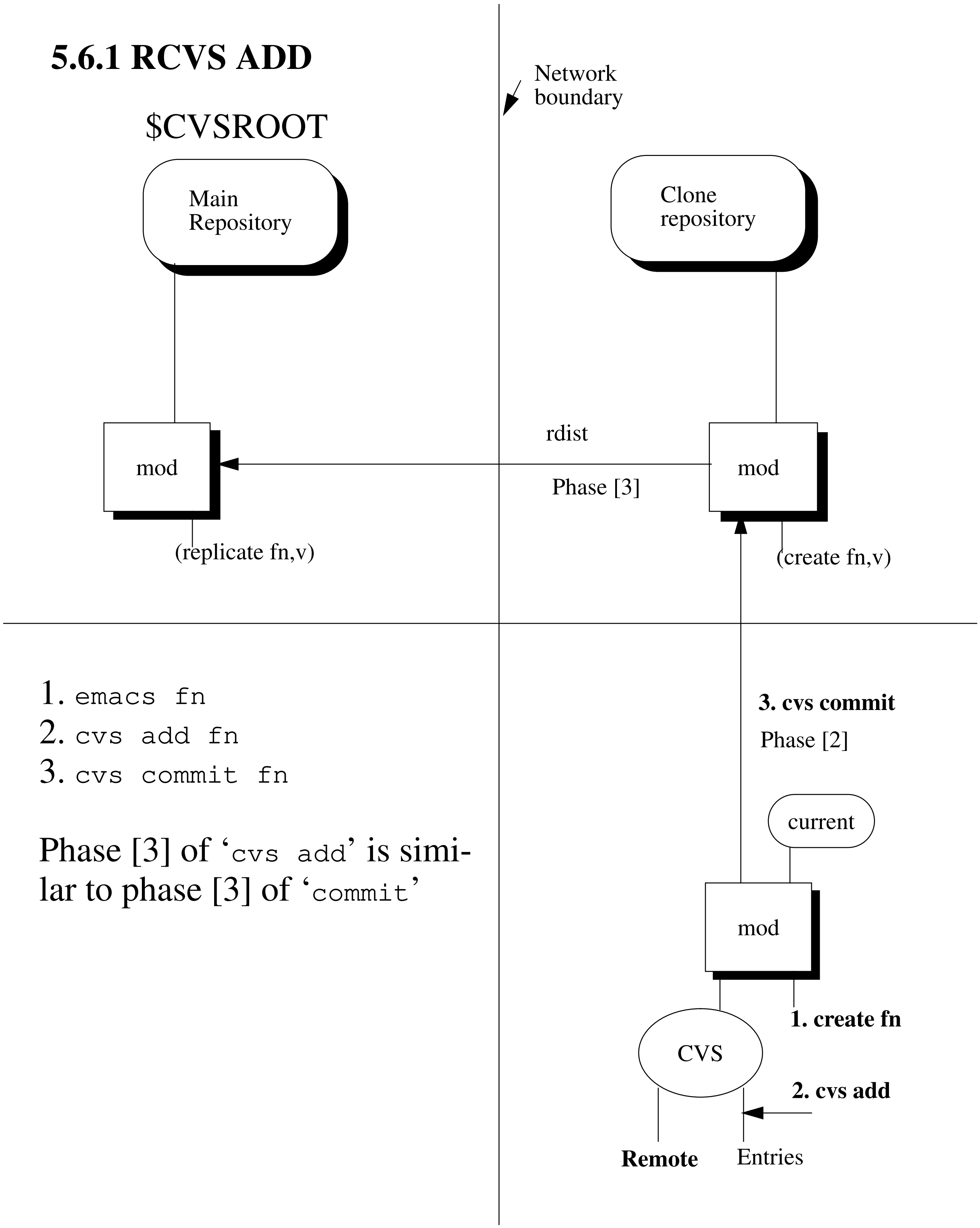
CVS/Entries before `cvs add`:

```
/f1/1.7/date//
```

CVS/Entries after `cvs add`:

```
/f1/1.2/date//
/fn/1.2/date
```

5.6.1 RCVS ADD



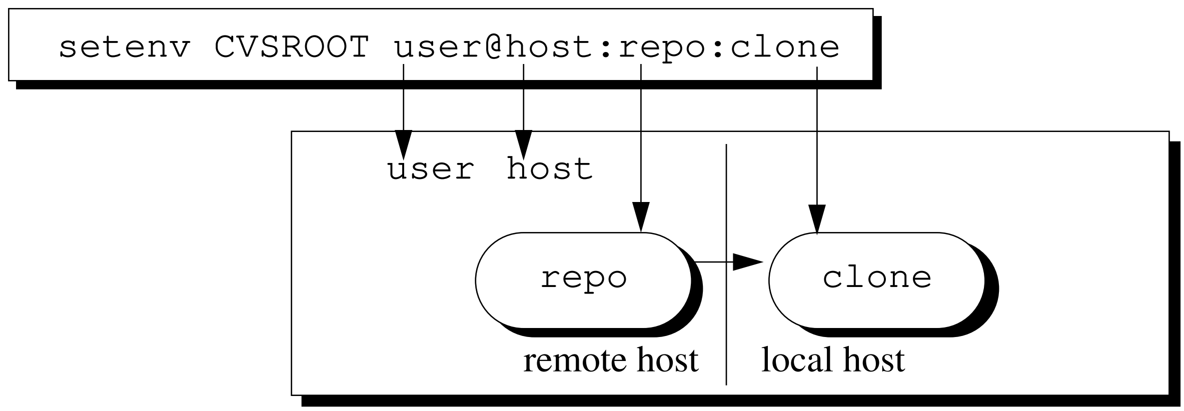
1. `emacs fn`
2. `cvs add fn`
3. `cvs commit fn`

Phase [3] of '`cvs add`' is similar to phase [3] of '`commit`'

6. RCVS Usage:

• 6.1 How to set $CVSROOT and/or CVSROOTr:

6.1.1 Using Single CVSROOT



Where ‘`user`’ is the login account on the server host. ‘`host`’ is the network address of the server host. ‘`ropo`’ is the main repository on the server host. ‘`clone`’ is the clone directory on local host.

example:

```
setenv CVSROOT terryh@dec.com:/repo:/clone
```

6.1.2 Optionally, use CVSROOT and CVSROOTr:

```
setenv CVSROOT local_repository
setenv CVSROOTr user@host:ropo:clone
```

Use option ‘`-R`’ to do remote checkout according to CVSROOTr. Ignoring ‘`-R`’ to do local checkout according to CVSROOT. Optionally, setenv CVSMODE to ‘`remote`’ or ‘`local`’ to switch between CVSROOT and CVSROOTr.

example:

```
setenv CVSROOTr terryh@dec.com:/repo:/clone
cvs -R co modules
```

- **6.2 How to set up main repository for group users:**

— ’`chgrp`’ all the directories (and sub-directories) in the main repository to the proper group name.

```
chgrp -R grp *
```

```
ls -l *

drwxr-xr-x terryh grp fn
```

— ’`chmod`’ them to ’`drwxrwsr_x`’ (`2775`) with ’`chmod g+ws`’. (The ’g+s’ turns on GID bit so that files created in a directory will carry the same group ID).

```
chmod -R g+ws *
```

```
ls -l *

drwxrwsr-x terryh grp fn
```

— The directories in the clone will be ’`chmod`’ed to the sam symbolic (not numeric) group name by RCVS if this group name exists on local host.

- **6.3 How to set up network connection:**

— 6.3.1 .rhosts:
Set up $HOME/.rhosts on both side of the network so that two

accounts can rlogin to each other

.rhosts on local host:

```
roc.slac.edu terryh
```

.rhosts on remote host

```
local.edu terryh
```

— 6.3.2 .cshrc:
rdist requires that no message was sent to stdout during remote login, if you have 'echo ...' in .cshrc, comment it out.

.cshrc on local or remote host:

```
echo "this is .cshrc"
```

7. Summary:

- **7.1 Speed consideration of RCVS**

— Some example of 'ping' response:

cithe3.cithep.caltech.edu	26ms
shemesh.lbl.gov	29ms
uhhepm.phys.hawaii.edu	71ms
cdsun.fnal.gov	73ms
astro.princeton.edu	112ms
mozart.physics.mcgill.ca	299ms

— rating network response:

$0 < avg < 100$	: fast, real time
$100 < avg < 200$	: medium, use file mode
$200 < avg$	: slow, use file mode, occasional update/commit

- **7.2 Conclusion**

— RCVS is currently used at SLAC to allow collaborators in U. Hawaii and McGill to involve in concurrent developement of HippoDraw, Reason, XmStep, etc.
— also used at CalTech, DESY, FNAL, GEM(SSC), IFH, LBL, LLNL, LTH, McGill, Princeton, SunWest, etc.
— With CVS's ability to support large hierarchy of directories with a large number of developers, plus the remote access provided by RCVS extension, it seems ideal to provide code management for HEP collaborators.