Proposal:
How to Collaborate on Model Development

Andras Varga, Rudolf Hornig
Several branches
• in private repositories
Several extensions
• various web sites
Patches, bugfixes posted to the mailing list
End User Perception

- Patches, modifications, more patches...
- "Which is the latest version?"

- At minimum, end users need:
  - a "stable" version
    - well tested, well documented, etc
    - conservative development model
  - a "bleeding edge" version
    - all-in-one
    - latest-and-greatest
    - may be rough around the edges
Biggest project: Linux kernel

- every release of the Linux kernel is being developed by nearly 1,000 developers, working for more than 100 corporations.
- an average of 3,621 lines of code are added to the kernel tree every day

How do they manage? Distributed Version Control.

- git
  - formerly they used BitKeeper
  - before that they used tarballs and emailed patches to each other
    "which was a vastly superior form of version control than CVS"
  - Linus Torvalds
Distributed Version Control

Centralized Version Control
(cvs, svn)

Distributed Version Control
(git, darcs, mercurial, bazaar)
DVCS Advantages

- allows private work
  - people don't need a password to a central repo to begin hacking on the project!
  - no central repo, but one may be designated as such
- easy branching and merging
  - enables parallel versions co-exist, with possibility of exchanging code between them
  - enables implementing new protocols/features/fixes in a topic branch
- efficiency
  - allows work without network connection
  - most operations are much faster
Why git?

- extremely fast
- very scalable
- tracks content not files (!)
- supports several branches per repo
- sophisticated merge algorithms (e.g. "subtree")
- avoids versioning number chaos
- makes everything possible (even rewriting history)
- graphical tools, Eclipse plug-in
- github!
INET on github

inet-framework / Inet

Description: INET framework for the OMNeT++ discrete event simulator
Clone URL: git://github.com/inet-framework/inet.git

inets

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>message</th>
</tr>
</thead>
<tbody>
<tr>
<td>.project</td>
<td>4 days</td>
<td>removed quegga, files moved one level up. [Hornig Rudolf]</td>
</tr>
<tr>
<td>.medfolders</td>
<td>4 days</td>
<td>removed quegga, files moved one level up. [Hornig Rudolf]</td>
</tr>
<tr>
<td>.oppspec</td>
<td>4 days</td>
<td>removed quegga, files moved one level up. [Hornig Rudolf]</td>
</tr>
<tr>
<td>.oppspec-b7</td>
<td>4 days</td>
<td>removed quegga, files moved one level up. [Hornig Rudolf]</td>
</tr>
</tbody>
</table>
Github names projects as: *user / project*

- **user**: **inet-framework**
- **projects**: **inet, inet-quagga**
- + "collaborators" (committers)

Branches in the INET repo:

- **master, stable**, topic/inetmanet

Current forks:

- **inetmanet / inetmanet**
- **ruengeler / sctpinet**
Simple Patch

remote repo (e.g. github)

Public Clone URL: `git://github.com/inet-framework/inet.git`

local repo

```bash
# git clone git://github.com/inet-framework/inet.git

# git checkout --b experimental

# edit files
# git commit --a

# edit files
# git commit --a

# git format-patch master --stdout > mypatch.diff

# email mypatch.diff
```
Forking a Repo on github

1. create a github user
   
2. add public key
   - ssh-keygen

3. fork inet

4. clone repo to local disk
   
   Your Clone URL: `git@github.com:avarga/inet.git`

   `$ git clone git@github.com:yourlogin/inet.git`
Making Changes

- **repo @github**
- **local repo**
- **git clone** `git@github.com:yourlogin/inetfork.git`

**edit files**
- **git commit --a**

**edit files**
- **git commit --a**

**git push** `origin master`
To create a new branch named "experimental", you can use the following commands:

1. `git branch experimental`
2. `git checkout experimental`

Next, you can edit files and commit them using:

1. `edit files`
2. `git commit -a`

After making changes, you can pull the latest changes from the master branch (optional) and then edit files and commit again:

1. `git pull master` (optional)
2. `edit files`
3. `git commit -a`

Finally, to switch back to the master branch and delete the experimental branch, you can use:

1. `git checkout master`
2. `git pull experimental`
3. `git branch -d experimental`
Contributing Commits Back to INET

- On github, forked projects can send a **pull request** to the original (or related) project
  - "please take over the following changes I made"
- Maintainer of original project reviews the code and checks it in
git for cvs/svn users

svn checkout    ~ git clone; git checkout
svn add         ~ git add
svn commit      ~ git commit -a  [+git push]
svn remove  --force ~ rm (with commit -a); or git rm
svn update      ~ git pull
svn revert      ~ git checkout .
Everyday tasks with GIT

- Create a repo: `git init`
- Clone a repo: `git clone url`
- Commit all changed files: `git commit -a`
- Add files to the index (=list of files to be committed): `git add file1 file2`
- Commit files added to the index: `git commit`
- Initial checkin: `git add .; git commit`
- To see the changes you will commit:
  - `git diff --cached`
Everyday tasks with GIT

- Changes compared to the last commit: `git diff`
- Getting history: `git log`
- Getting status info about files: `git status`
- Tagging (each commit is identified by its SHA1 hash, but you can give meaningful names to any version): `git tag rel2.0 acb56fea`
- Getting an older version: `git checkout rel2.0`
- Reverting your local changes: `git checkout .`
- Throwing away the last commit `git reset --hard HEAD^`
Branching and merging with GIT

- Branches are **cheap** - you should use them
- Show your current branches: `git branch`
- Create a new branch from the current head
  - `git branch experimental`
- Switch to a specific branch
  - `git checkout experimental`
- Merge the changes from the master to the experimental
  - `git merge master`
- Delete a branch: `git branch -d experimental`
Collaborating with GIT

- Clone a remote repository
  - `git clone git://github.com/inet-framework/inet.git`
- Create a name for a remote repo
  - `git remote add origin git://github.com/inet-framework/inet.git`
- Do local changes, commits etc. and then get the changes from the original remote repo
  - `git pull origin master`
- When ready to contribute back your changes, (you must have write rights for the remote repo)
  - `git push origin master`
OMNeT++ IDE Git support

- Uses the egit plugin ("Java Git / Eclipse Git")
- Usage:
  - start with: File | Import...
  - then: use Team submenu of the project menu
More about git

If you have time, watch to the following video on youtube:  
”Linus Torvalds on git”, Google Tech Talk
Git Resources

Tutorial *man* pages:
- gittutorial, gittutorial2, gitcore-tutorial, gitglossary

Various tutorials on the web:
- "Git - SVN Crash Course"
- "Manage source code using Git"
- ...

Git command *man* pages:
- man git-commit
- man git-pull
- ...
Questions, comments?
Discussion