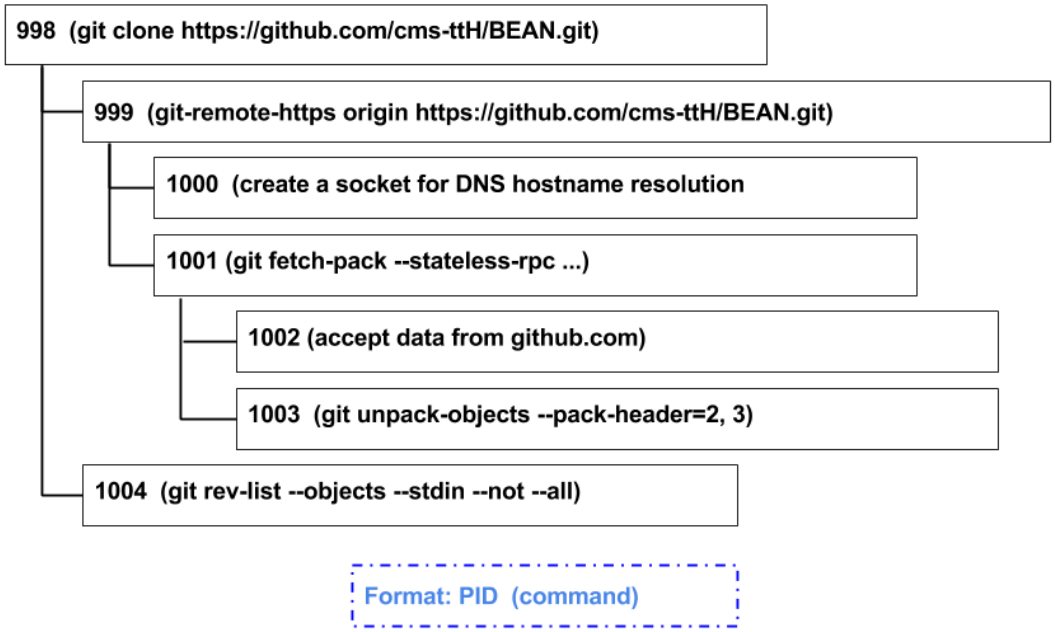
All the dependencies of one program can be divided into three categories: local file system (e.g., /usr, /lib and /lib64), remote file systems which can be mounted as local directories (e.g., /cvmfs and /hdfs), and other remote network dependencies (e.g. http, https and ssh).

Through utilizing **ptrace** to trap each file-relevant system call used by a program, the first two categories of dependencies can be collected.

Motivation of tracking network dependencies: Linkrot is a common and great threat to the preservation of scientific applications. A URL which works normally today may be unavailable permanently. A method which can help the scientists figure out all the network dependencies of their applications provides the chance to evaluate the stability of each network dependencies and preserve the unstable network resources before linkrot happens.

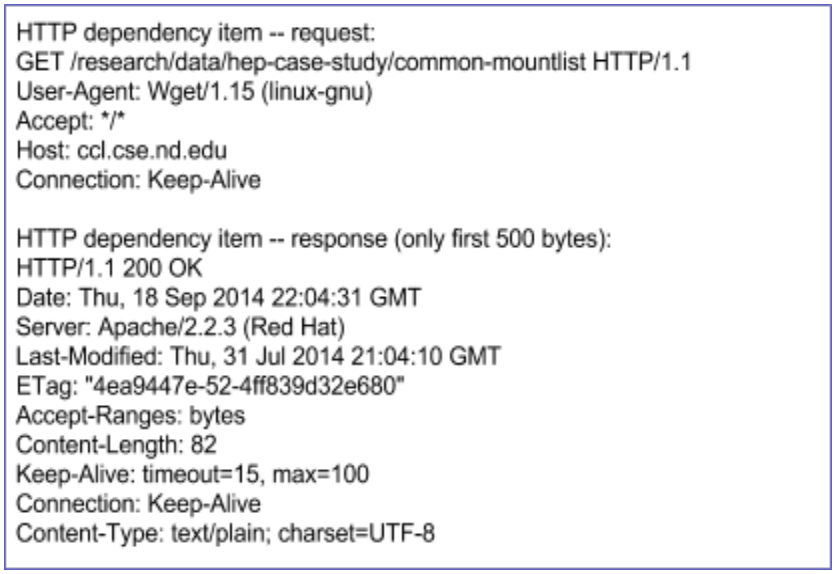
One direct solution to track the third category of dependencies is to utilize **ptrace** to check each exec system call used by a program and record each network-relevant executable and its parameters. However, an executable may create one or more child processes to finish some tasks, and a child process may create their own children processes. The following figure shows all the processes created by a git command. A process with PID 998 is created to clone a remote git repository into local machine, which totally has 6 child processes. Moreover, a network-relevant executable name collection is necessary to identify network dependencies. The diversity of network-relevant executables and the large process tree generated for a network executable make this solution infeasible.



Another solution to track network dependencies is to track the network sockets. The user process communicates with the network protocol stacks in the kernel through the network socket layer user interface on Linux. From the process tree shown in the above figure, the process with PID 1000 responds to create a

socket, connect to a DNS server, send a DNS request packet to the DNS server and receive a DNS response packet from the DNS server. The information gained from the network socket tracking can be divided into three categories.

1) The information of network sockets used to communicate with remote network dependences (through socket and connect system calls) can be used to figure out the port number, service name (such as, http, https, and ssh), socket type (stream and datagram), and the domain type (inet and inet6).
2) The contents of DNS packets can be used to figure out the hostname and IP address of each remote network dependency.
3) As for applications based on http protocol, all the http requests and responses can be seen, as shown in the following figure. However, as for applications based on https and ssh which encrypt network data using TLS/SSL, tracking network data on the socket level can only see the encrypted data.



The results of tracking network sockets used by a program involves two categories of network resource redirections, dns-level hostname redirection and website-level url redirection.

dns-level hostname redirection: Each DNS packet includes five parts, header, question, answer, authority, and additional. A DNS response packet with the type CNAME has multiple answers. For example, the DNS response packet which tries to resolve

www3.nd.edu includes two answers. The first answer provides an alias of www3.nd.edu, www-vip.cc.nd.edu, and the second answer maps the alias to the IP address.

```
www3.nd.edu CNAME www-vip.cc.nd.edu
www-vip.cc.nd.edu A 129.74.12.151
```

The CNAME-type DNS packets leave us a question, among the hostname used by the user, the alias(es), the IP address, what should we preserve?

website-level url redirection: Website-level url redirection happens when a website updates its location and still want their old users access their resources using the old url. The following figure illustrates the url redirection between www3.nd.edu/~ccl and ccl.cse.nd.edu. Except for the difference of hostnames, the application layer protocols used by the old-version website and the new-version website is also different: the old website provides services through the https protocol, while the new website provides services through the http protocol.

The usage of https protocol makes detecting of website-level url redirections through tracking system calls more difficulty and even impossible. All the application data will be encrypted at presentation layer through the TLS/SSL protocol, which is initialized at the session layer and works at the presentation layer.

```
$ wget https://www3.nd.edu/~ccl/research/data/hep-case-study/common-mountlist
--2014-09-19     14:16:28--          https://www3.nd.edu/~ccl/research/data/hep-case-
study/common-mountlist
Resolving www3.nd.edu (www3.nd.edu)... 129.74.12.151
Connecting to www3.nd.edu (www3.nd.edu)|129.74.12.151|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location:         http://ccl.cse.nd.edu/research/data/hep-case-study/common-mountlist
[following]
--2014-09-19  14:16:28--   http://ccl.cse.nd.edu/research/data/hep-case-study/common-
mountlist
Resolving ccl.cse.nd.edu (ccl.cse.nd.edu)... 129.74.152.167
Connecting to ccl.cse.nd.edu (ccl.cse.nd.edu)|129.74.152.167|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 82 [text/plain]
Saving to: 'common-mountlist'

100%
[=============================================================================
==================================================>] 82          --.-K/s   in 0s

2014-09-19 14:16:28 (4.96 MB/s) - 'common-mountlist' saved [82/82]
```