Using Git to circumvent censorship of access to the Tor network

Björgvin Ragnarsson and Pieter Westein
University of Amsterdam
Research Project 2

Outline

- Introduction
- Research Question

- Git overview

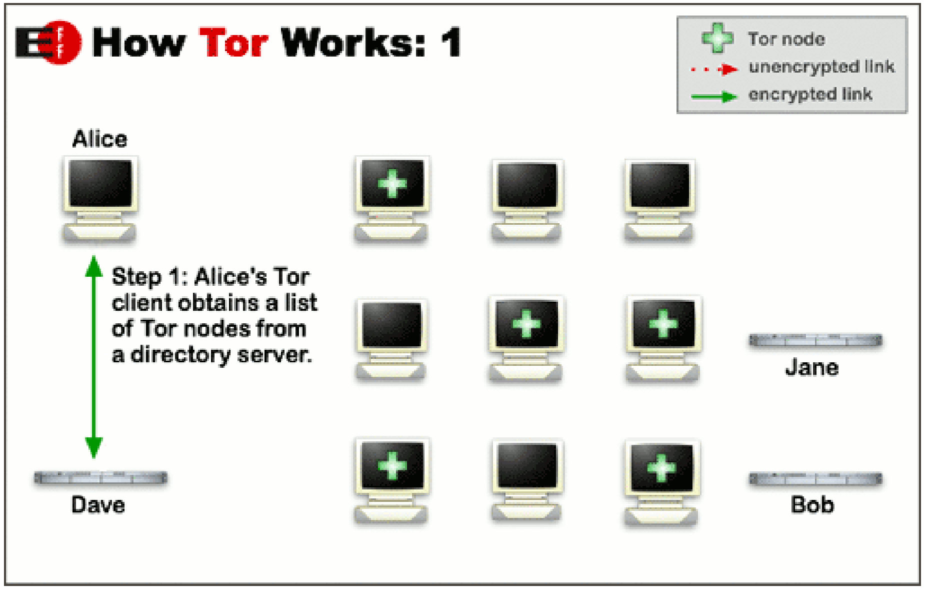
- Design overview
- Demo

- Performance measurements
- Prototype evaluation

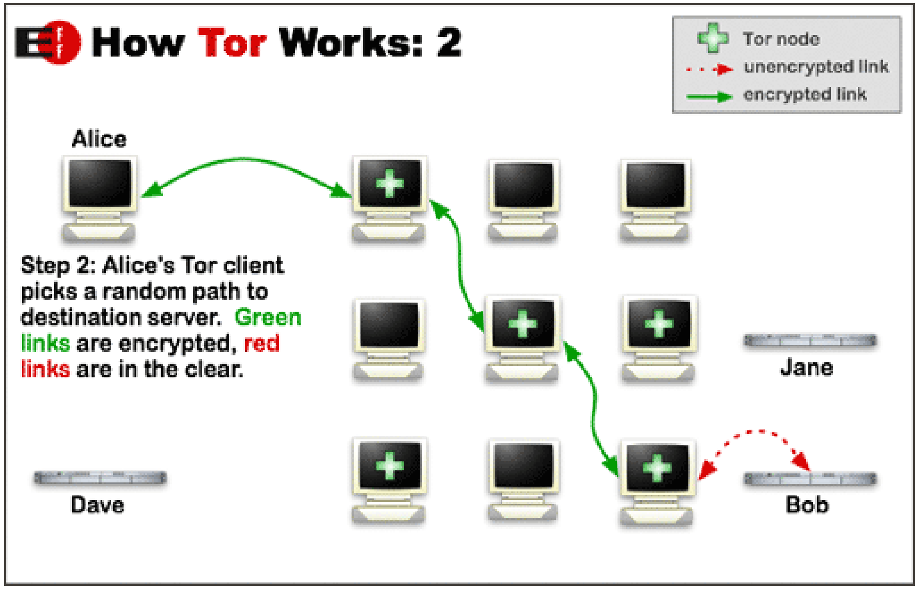
- Conclusion and Future work

- Questions

Tor overview



Step 1: Alice's Tor client obtains a list of Tor nodes from a directory server.



Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.

Source: https://www.torproject.org/about/overview.html.en

Censorship and resistance

- Tor relays are public, easy to block
- Introduction of Bridges
- Scanners actively trying to reach Bridges
- Introduction of Pluggable Transports

Pluggable Transports

- Modules for obfsproxy framework

- Can be used for other purposes than Tor
  - as a SOCKS proxy

- Existing transports
  - Obfs2,Obfs3, Skype, ScrambleSuit, Dust, StegoTorus, flashproxy

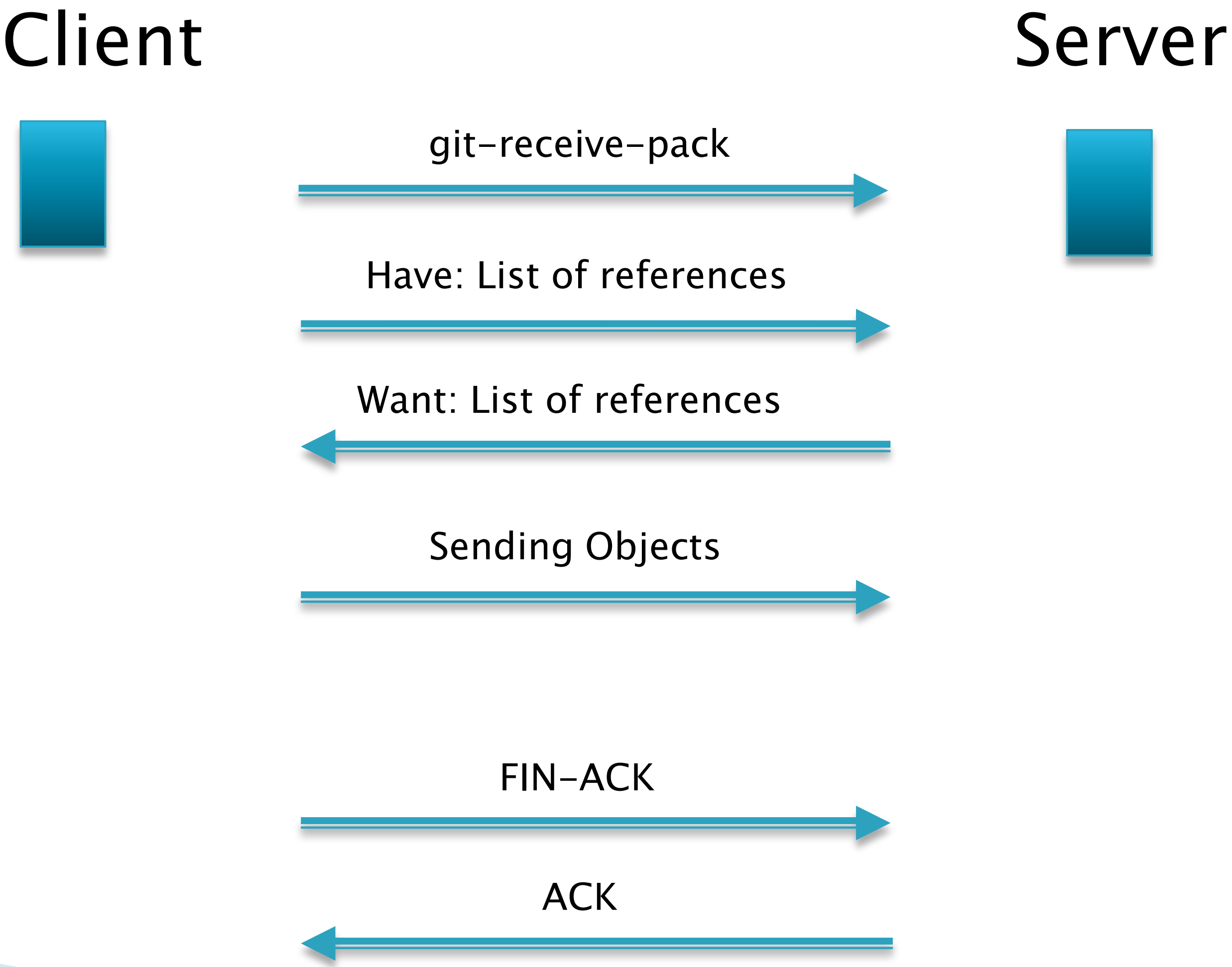
Research Question

- Is it possible to shape Tor traffic in such a way that it looks identical to the Git protocol?
- How could a censor identify Tor bridges and users using such an obfuscated protocol?

Git overview

- Version control system
- Push and pull mechanism
- Four transports protocols
  - SSH, Git, HTTP, HTTPS

Pushing



Client

Server

git-receive-pack

Have: List of references

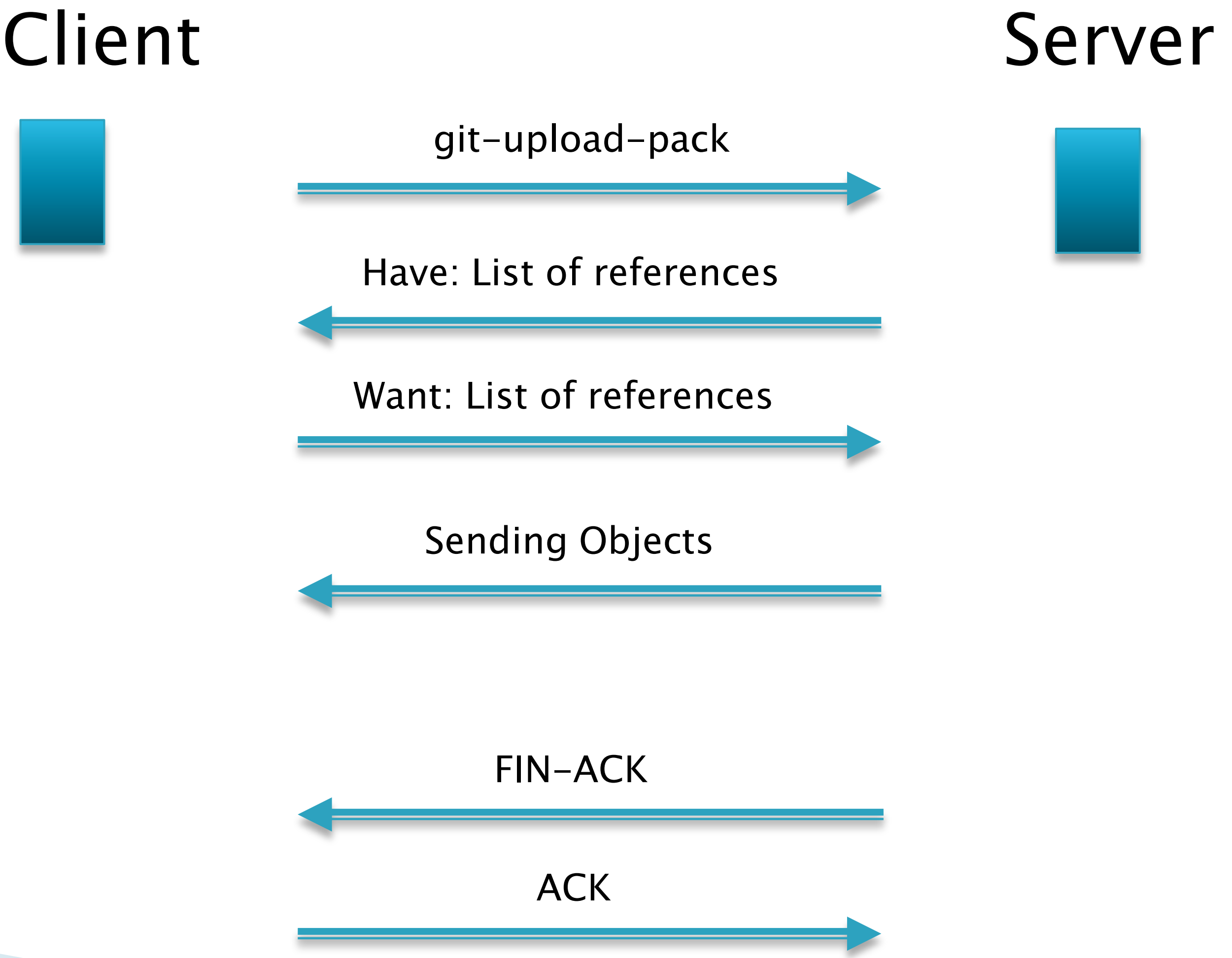
Want: List of references

Sending Objects

FIN-ACK

ACK

Pulling



Client

Server

git-upload-pack

Have: List of references

Want: List of references

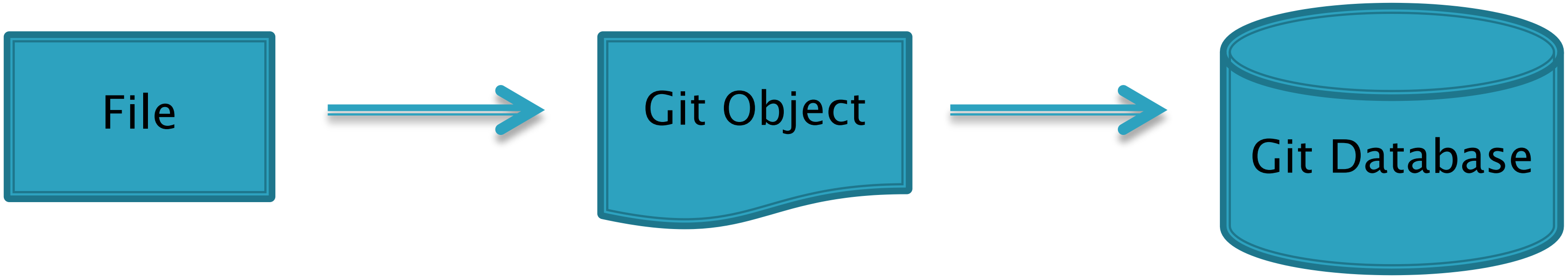
Sending Objects

FIN-ACK

ACK

Object Storage

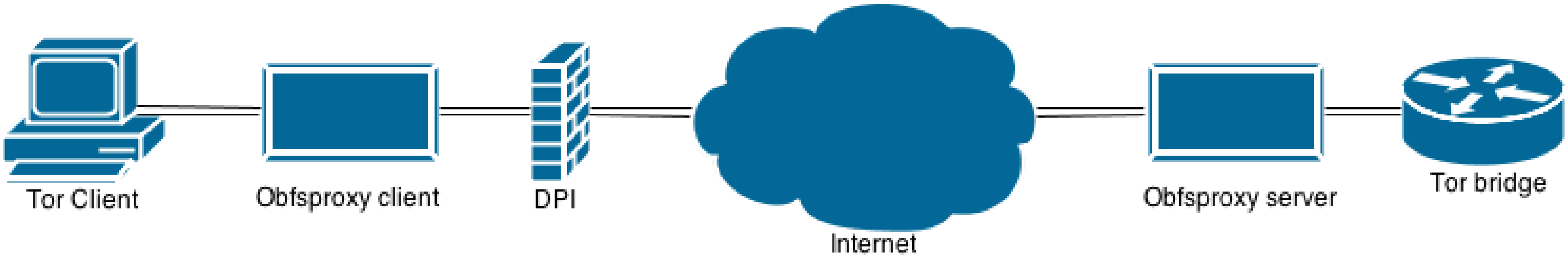
- Files compressed and stored in the Git database
- SHA1 hash of the content used as reference



Design overview

- TCP stream is stored as files in Git
- The Git program does the transfer
  - Makes it harder to fingerprint
  - Provides four transports in one:
    - (SSH/Git/HTTP/HTTPS)
- Client initiates send/receive

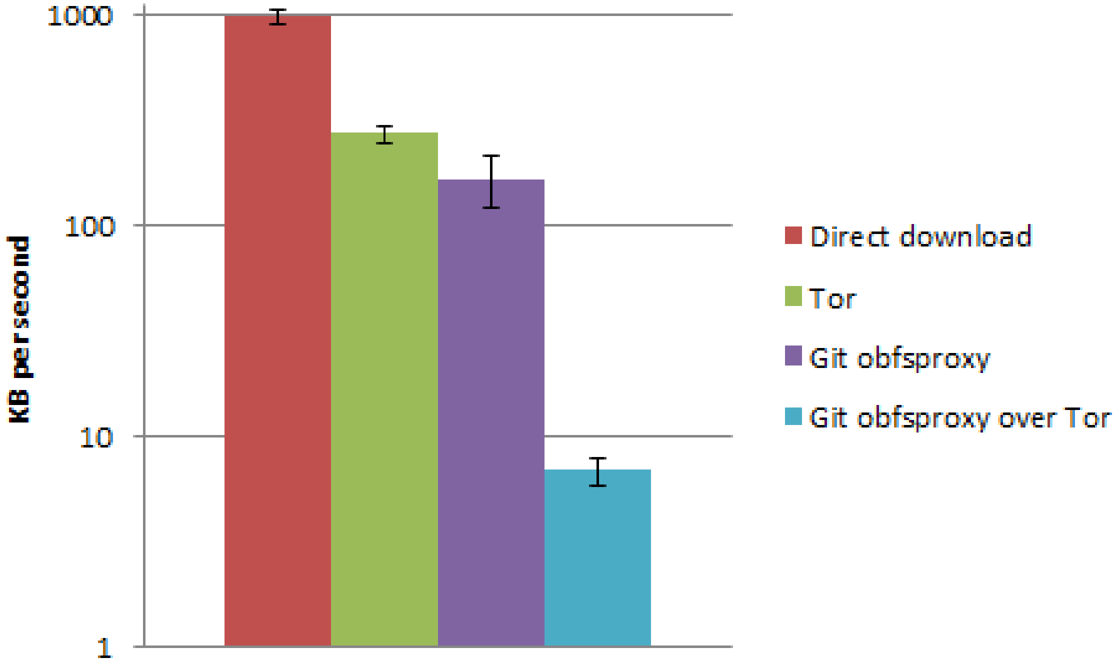
Obfsproxy


Tor Client
Obfsproxy client
DPI
Internet
Obfsproxy server
Tor bridge

Demo time!

Performance measurements

- Downloading a 10MB file using git over ssh
  - 7 KB/s over ssh through Tor
  - 166 KB/s over ssh without Tor



Prototype evaluation

- The frequency of pushes and pulls
- Tor data is compressed (not hidden)
- Git traces on disk

Conclusion

- Tor usage can be obfuscated as Git traffic
  - or any other VCS
- Prototype is slow, compared to normal Tor
- Polling and disk writes are weak points

Future work

- Using publicly available Git servers for relaying
- Layered obfuscation
- Eliminate disk writes

Questions?

Thank you for your attention

Track development at:
https://trac.torproject.org/projects/tor/ticket/9192