GNU**net**

presentation for DC^{10}

by

— not disclosed due to DMCA —

GNUnet Requirements

- Anonymity
- Confidentiality
- Deniability
- Accountability
- Efficiency

Applications

- anonymous sharing of medical histories
- distributed backups of important data
- ad-hoc communication between small devices
- and others

Infrastructure

We call GNUnet a network because:

- file-sharing is just *one* possible application
- most components can be re-used for other applications:
  - ⋆ authentication
  - ⋆ discovery
  - ⋆ encrypted channels
  - ⋆ accounting
- the protocol is extensible and extentions are planned

Related Work

Network	Gnutella[1, 4]	Chord[24]	Freenet[9]	MojoNation[17]
Search	bf-search	compute	df-search	broker
Anonymous	no	no	yes	no
Accounting	no	no	no	yes
File-Sharing	direct	migrated	insert	insert

Chord[24], Publius[15], Tangler[16], CAN[19] and Pastry[21, 7] are equivalent from the point of view of this discussion.

Outline of the Talk

1. Encoding data for GNUnet

2. Searching in GNUnet

3. Anonymity in GNUnet

4. Accounting in GNUnet

Encoding in GNUnet

- Requirements
- Trees
- Blocks
- Limitations
- Benefits

Problems with existing Systems

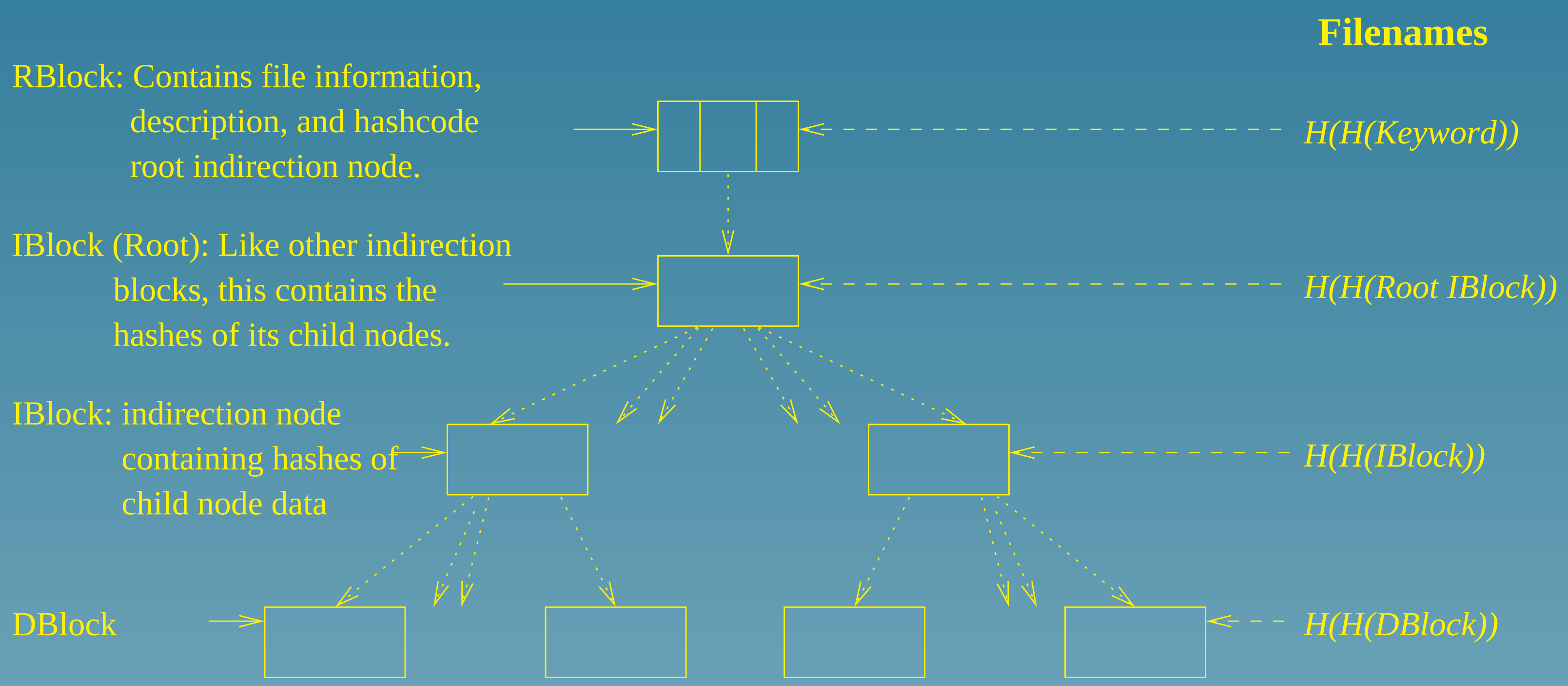
- Content submitted in plaintext, or
- content must be inserted into the network and is then stored twice, in plaintext by the originator and encrypted by the network (e.g. Freenet[9]);
- in some systems, independent insertions of the same file results in different copies in the network (e.g. Publius[15])

Encoding data for GNUnet: Requirements

- intermediaries can not find out content or queries
- hosts can send replies to queries and deny knowing what the query or the content was for
- keep storage requirements (and bandwidth) small

Tree Encoding

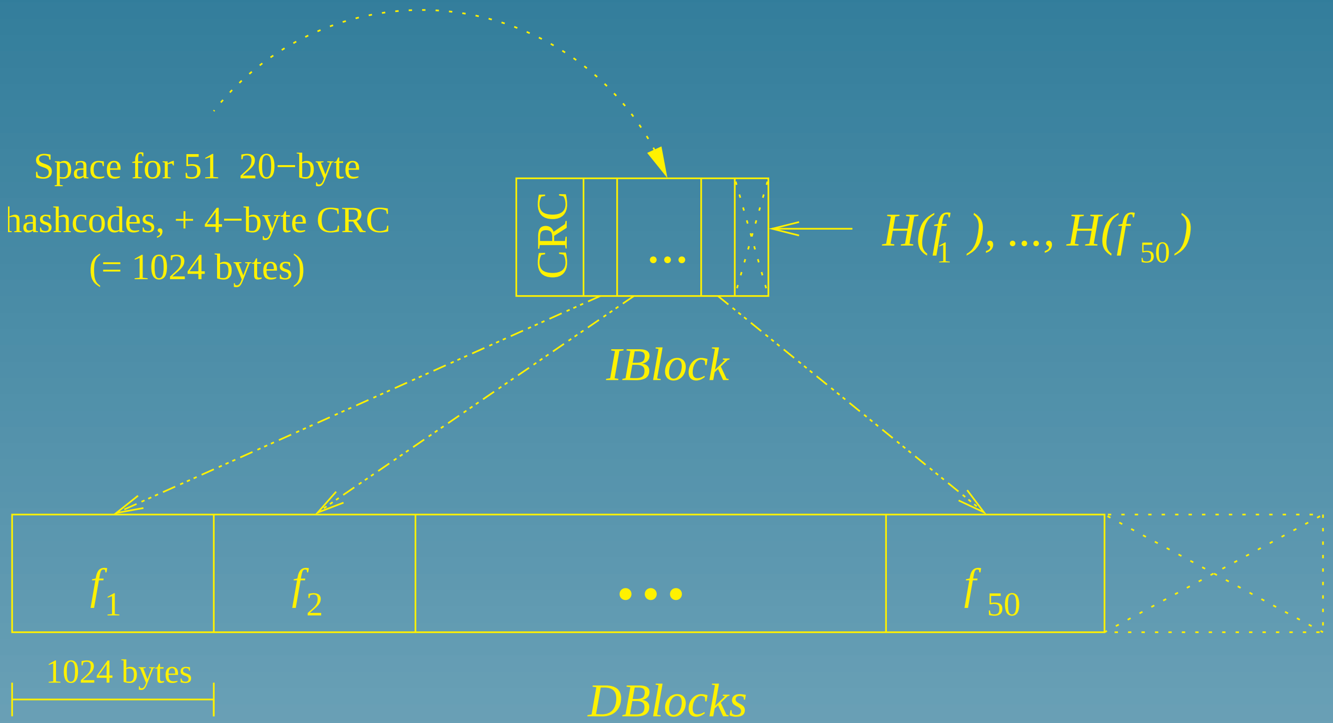
Files in GNUnet are split into 1k blocks for the transport[6]:



Encoding of the entire file

Block Encoding

The hash of 51 blocks and a CRC are combined to an *IBlock*:



Encoding of the entire file

“Algorithm”

- split content into 1k blocks B (UDP packet size!)
- compute $H(B)$ and $H(H(B))$
- encrypt B with $H(B)$, with Blowfish
- store $E_{H(B)}(B)$ under $H(H(B))$
- build inner blocks containing $H(B)$
- root-node R contains description, file-size and a hash

Limitations

- If the keywords can be guessed... participating hosts can decrypt the query.
- If the exact data can be guessed... participating hosts can match the content.
- This is intended to reduce storage costs!

Benefits

- encryption of blocks independent of each other
- inherent integrity checks
- multiple (independent) insertions result in identical blocks
- very fast, minimal memory consumption
- little chance of fragmentation on the network
- small blocksize enables us to make traffic uniform and thus traffic analysis hard

Searching in GNUnet

- Requirements
- Boolean queries
- Searching: Triple-Hash
- Routing
- Anonymity preview

Problems with existing Systems

- Centralized, or
- easy to attack by malicious participants.
- Queries in plaintext, or
- hard to use keys.
- Not anonymous, or
- malicious participants can send back garbage without begin detected.

Requirements

- retrieve content with simple, natural-language keyword
- guard against traffic analysis
- guard against malicious hosts
- do not expose actual query
- do not expose key to the content
- be unpredictable
- support arbitrary content locations
- be efficient

Ease of Use

GNUnet must be easy to use:

- search for “mp3” AND “Metallica” AND “DMCA”
- GNUnet returns list of files with description
- user selects interesting file
- GNUnet returns the file

Encrypting the root-node R

For each file, the user specifies a list of keywords to `gnunet-insert`. Then:

- For each keyword K:
- GNUnet saves $E_{H(K)}(R)$ under $H(H(K))$.

If the user searches for “foo” and “bar”:

- Search for “foo”, search for “bar”.
- Find which root-nodes that are returned are for the same file ($=$ top-level hash). Display those.

Searching: Intuition

- Key for block B is $H(B)$.
- Filename for block B is $H(H(B))$.
- Intuition: ask for $H(H(B))$, return $E_{H(B)}(B)$.
- Problem: malicious host sends back garbage, intermediaries can not detect

Triple-Hash

- Send query: $H(H(H(B)))$.
- Reply is $\{H(H(B)), E_{H(B)}(B)\}$.
- Malicious host must at least have $H(H(B))$ and thus probably the content.
- It is *impossible* to do better together with anonymity and confidentiality of query and content for sender and receiver.

Routing

- keep a table of hosts that we are connected with
- forward query to n randomly chosen hosts
- select n based on load and importance of the query
- keep track of queries forwarded, use time-to-live to detect loops
- bias the random choice of the hosts slightly towards a Chord-like metric.
- take metric into account when migrating content

GNUnet: Traffic Analysis Nightmare

- Group several queries to one larger packet.
- Introduce delays when forwarding.
- Packets can contain a mixture of queries, content, node-discovery, garbage, etc.
- Make all packets look uniform (in size).
- Encrypt all traffic. Add noise if idle.

Open issues

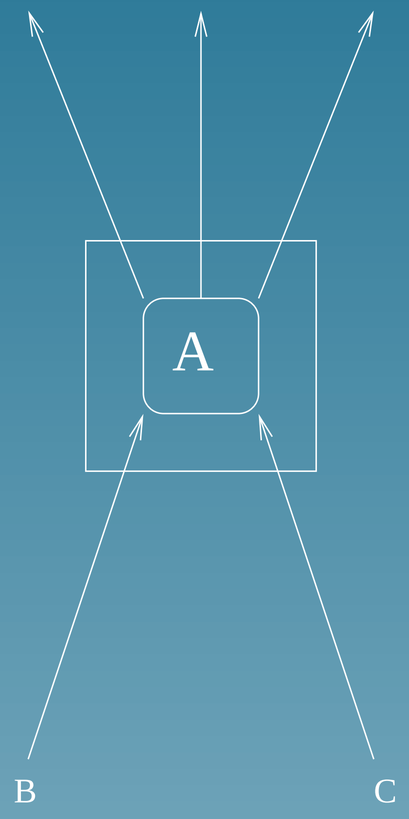
- Approximate queries.

Anonymity in GNUnet

- Techniques to achieve anonymity
- Attacks
- Efficiency
- A new perspective
- GNUnet is malicious

Building Blocks

- indirections[25]
- random delays[10]
- noise[11, 22]
- confidential communication[18]



Attacks on Anonymity

- traffic analysis[3]
- timing analysis
- malicious participants
- statistical analysis[20, 23]

Efficiency

If nodes indirect queries and replies, this has serious efficiency implications:

> For n indirections, the overhead in bandwidth (and encryption time) is n-times the size of the content.

Money Laundering

Let's illustrate GNUnet's perspective[5] with the example of money laundering. If you wanted to hide your financial traces, would you:

- Give the money to your neighbor,
- expect that your neighbor gives it to me,
- and then hope that I give it to the intended recipient?

Worse: trust everybody involved, not only that we do not steal the money but also do not tell the FBI?

Banks!

In reality, banks are in the best position to launder money:

- Take 1.000.000 transactions from customers,
- add your own little transaction,
- and better not keep any records.

As long as not *all* external entities cooperate against the bank, nobody can prove which transaction was ours.

Why indirect?

- Indirections do not protect the sender or receiver.
- Indirections can help the indirector to hide its own traffic.
- If the indirector cheats (e.g. by keeping the sender address when forwarding) it only exposes its own action and does not change the anonymity of the original participants.

Key Realization

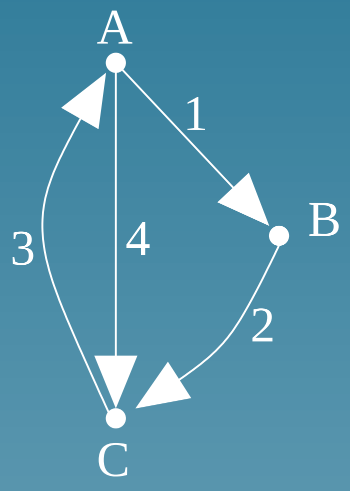
Anonymity can be measured in terms of

- how much traffic from non-malicious hosts is indirected compared to the self-generated traffic
- in a time-interval small enough such that timing analysis can not disambiguate the sources.

GNUnet: anonymity for free

From this realization, we can motivate GNUnet's anonymity policy:

- indirect when idle,
- forward when busy,
- drop when very busy.



Rationale: if we are indirecting lots of traffic, we don't need more to hide ourselves and can be *more efficient* by merely forwarding.

Accounting in GNUnet

- Goals
- Requirements
- Human Relationships!
- Digital Cash?
- Transitivity
- Open issues

Common Problems

- No accounting: easy to mount DoS attack[12]
- Overpricing legitimate use[2]
- Centralization[8]
- Lack of acceptance for micropayments
- Patents

Goals

- Reward contributing nodes with better service.
- Detect attacks:
  - ⋆ detect flooding,
  - ⋆ detect abuse,
  - ⋆ detect excessive free-loading, but
  - ⋆ allow *harmless* amounts of free-loading

Requirements

- No central server (rules out [17, 8]).
- No trusted authority (problem of initial accumulation, see [13]).
- Everybody else is malicious and violates the protocols.
- Everybody can make-up a new identity at any time.
- New nodes should be able to join the network.

Human Relationships

- We do not have to *trust* anybody to form an opinion.
- Opinions are formed on a one-on-one basis, and
- may not be perceived equally by both parties.
- We do *not* charge for every little favour.
- We *are* grateful for every favour.
- There is no guarantee in life, in particular Alice does not have to be kind to Bob because he was kind to her.

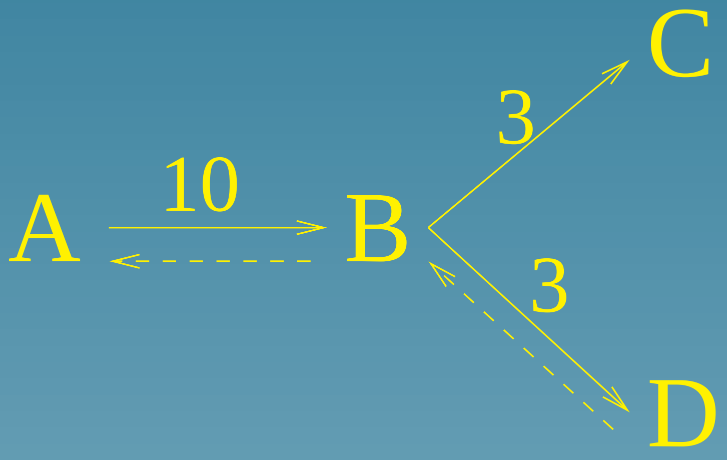
Excess-based Economy

GNUnet's economy[14] is based on the following principals:

- if you are *idle*, doing a favour for free does not cost anything;
- if somebody does you a favour, remember it;
- if you are *busy*, work for whoever you like most, but remember that you paid the favour back;
- have a *neutral* attitude towards new entities;
- never dislike anybody (they could create a new identity anytime).

Transitivity

If a node acts on behalf on another, it must ensure that the sum of the charges it may suffer from other nodes is lower than the amount it charged the sender:



Transitivity in the GNUnet economy.

Open Issues

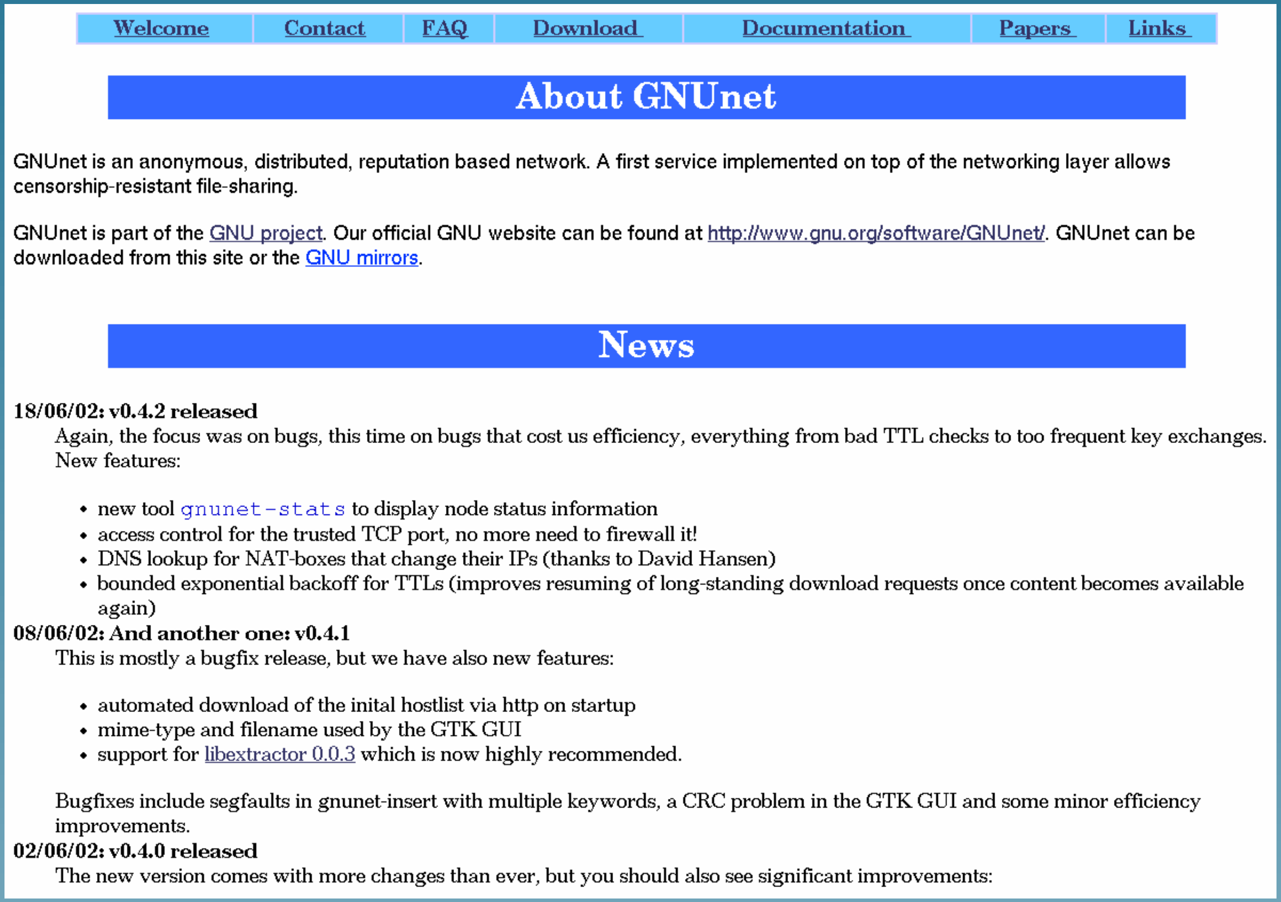
- if a node is idle, it will not charge the sender;
- if a node delegates (indirects), it will use a lower priority than the amount it charged itself;
- if an idle node delegates, it will always give priority 0.
- A receiver can not benefit from answering a query with priority 0.
- If the priority is 0, content will not be marked as valuable.

Conclusion

- GNUnet is a cool system for privacy.
- GNUnet can already be used.
- GNUnet could get much better.

GNUnet Online

http://www.ovmj.org/GNUnet/



About GNUnet

GNUnet is an anonymous, distributed, reputation based network. A first service implemented on top of the networking layer allows censorship-resistant file-sharing.

GNUnet is part of the GNU project. Our official GNU website can be found at http://www.gnu.org/software/GNUnet/. GNUnet can be downloaded from this site or the GNU mirrors.

News

18/06/02: v0.4.2 released

Again, the focus was on bugs, this time on bugs that cost us efficiency, everything from bad TTL checks to too frequent key exchanges. New features:

- new tool `gnunet-stats` to display node status information
- access control for the trusted TCP port, no more need to firewall it!
- DNS lookup for NAT-boxes that change their IPs (thanks to David Hansen)
- bounded exponential backoff for TTLs (improves resuming of long-standing download requests once content becomes available again)

08/06/02: And another one: v0.4.1

This is mostly a bugfix release, but we have also new features:

- automated download of the inital hostlist via http on startup
- mime-type and filename used by the GTK GUI
- support for libextractor 0.0.3 which is now highly recommended.

Bugfixes include segfaults in gnunet-insert with multiple keywords, a CRC problem in the GTK GUI and some minor efficiency improvements.

02/06/02: v0.4.0 released

The new version comes with more changes than ever, but you should also see significant improvements:

GNUnet resources

- FAQ
- Mailinglists
- Mantis
- README
- Sources
- WWW page

References

[1] E. Adar and B. Huberman. Free riding on gnutella. Technical report, Xerox Parc, Aug. 2000.

[2] Adam Back. Hash cash - a denial of service counter-measure, 1997.

[3] Adam Back, Ulf Moeller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems.

[4] S. Bellovin. Security aspects of napster and gnutella, 2000.

[5] K. Bennett and C. Grothoff. gap - practical anonymous networking. 2002.

[6] K. Bennett, C. Grothoff, T. Horozov, and I. Patrascu. Efficient sharing of encrypted data. In *Proceedings of ASCIP 2002*, 2002.

[7] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting network proximity in peer-to-peer overlay networks.

[8] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Crypto '88*, pages 319–327, 1988.

[9] I. Clarke. A distributed decentralised information storage and retrieval system, 1999.

[10] G. Danezis, R. Dingledine, D. Hopwood, and N. Mathewson. Mixminion: Design of a type iii anonymous remailer, 2002.

[11] Wei Dei. Pipenet.

[12] Roger Dingledine, Michael J. Freedman, and David Molnar. *Accountability*. 2001.

[13] Friedrich Engels. *Umrisse zu einer Kritik der Nationalökonomie*. 1844.

[14] C. Grothoff. An excess based economy. 2002.

[15] Aviel D. Rubin Marc Waldman and Lorrie Faith Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proc. 9th USENIX Security Symposium*, pages 59–72, August 2000.

[16] David Mazieres Marc Waldman. Tangler: A censorhip-resistant publishing system based on document entanglements. 2001.

[17] Mojo Nation. Technology overview, Feb. 2000.

[18] George Orwell. *1984*. 1949.

[19] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA, 2000.

[20] Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[21] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems.

[22] R. Sherwood and B. Bhattacharjee. P5: A protocol for scalable anonymous communication. In *IEEE Symposium on Security and Privacy*, 2002.

[23] Clay Shields and Brian Neil Levine. A protocol for anonymous communication over the internet. In *ACM Conference on Computer and Communications Security*, pages 33–42, 2000.

[24] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. pages 149–160.

[25] P F Syverson, D M Goldschlag, and M G Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 4–7 1997.