NEXT GENERATION INTERNET

Distributed systems

Christian Grothoff

6.03.2026

Learning objectives

What should we think about when building distributed systems?

- What are typical attacks on decentralized systems?

Distributed Hash Tables

- CAN
- Chord
- Kademlia

Routing in the Dark

- Pitch Black

What are impossibly hard problems in distributed systems security?

Offline payments

Part I: What should we think about when building distributed systems?

The 8 Fallacies of Distributed Computing[1]

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology does not change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

[1] According to Peter Deutsch and James Gosling

Self stabilization (Dijkstra 1974) [1]

- A system is self-stabilizing, if starting from any state, it is guaranteed that the system will eventually reach a correct state (convergence).
- Given that the system is in a correct state, it is guaranteed to stay in a correct state, provided that no fault happens (closure).
- Self-stabilization enables a distributed algorithm to recover from a transient fault **regardless of its nature**.

Example: Spanning-tree Protocol from Networking!

What are typical attacks on decentralized systems?

Sybils

Background:

- Ancient Greece: Sybils were prophetesses that prophesized under the devine influence of a deity. Note: At the time of prophecy not the person but a god was speaking through the lips of the sybil.
- 1973: Flora Rheta Schreiber published a book “Sybil” about a woman with 16 separate personalities.

The Sybil attack

The Sybil attack [3]:

- Insert a node multiple times into a network, each time with a different identity
- Position a node for next step on attack:
  - Attack connectivity of the network
  - Attack replica set
  - In case of majority votes, be the majority!

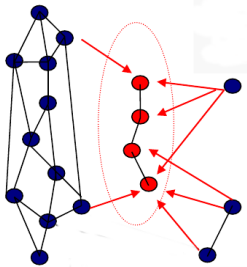
Defenses against Sybil Attacks

- Use authentication with trusted party that limits identity creation
- Use “external” identities (IP address, MAC, e-mail)
- Use “expensive” identities (solve computational puzzles, require payment)

Douceur [3]: Without trusted authority to certify identities, no realistic approach exists to completely stop the Sybil attack.

Eclipse attack: Goal

- Separate a node or group of nodes from the rest of the network
- isolate peers (DoS, surveillance) or isolate data (censorship)



Eclipse Attack: Techniques

- Use Sybil attack to increase number of malicious nodes
- Take over routing tables, peer discovery

⇒ Details depend on overlay structure

Eclipse Attack: Defenses

- Large number of connections
- Aggressive discovery (“continuous” bootstrap)
- Prefer long-lived connections / old peers
- Replication
- Diverse neighbour selection (different IP subnets, geographic locations)
- Audit neighbour behaviour (if possible)

Poisoning Attacks

Nodes provide false information:

- wrong routing tables [4]
- wrong meta data
- wrong performance measurements

Timing Attacks [13]

Nodes can:

- measure latency to determine origin of data
- delay messages
- send messages using particular timing patterns to aid correlation
- include wrong timestamps (or just have the wrong time set...)

Part II: Distributed Hash Tables

Distributed Hash Tables (DHTs)

- Distributed **index**
- GET and PUT operations like a hash table
- JOIN and LEAVE operations (internal)
- Trade-off between JOIN/LEAVE and GET/PUT costs
- Typically use exact match on cryptographic hash for lookup
- Typically require overlay to establish particular connections

DHTs: Key Properties

To know a DHT, you must know (at least) its:

- routing table structure
- lookup procedure
- join operation process
- leave operation process

... including expected costs (complexity) for each of these operations.

A trivial DHTs: The Clique

- routing table: hash map of all peers
- lookup: forward to closest peer in routing table
- join: ask initial contact for routing table, copy table, introduce us to all other peers, migrate data we're closest to to us
- leave: send local data to remaining closest peer, disconnect from all peers to remove us from their routing tables

Complexity?

A trivial DHTs: The Circle

- routing table: left and right neighbour in cyclic identifier space
- lookup: forward to closest peer (left or right)
- join: lookup own peer identity to find join position, transfer data from neighbour for keys we are closer to
- leave: ask left and rigt neighbor connect directly, transfer data to respective neighbour

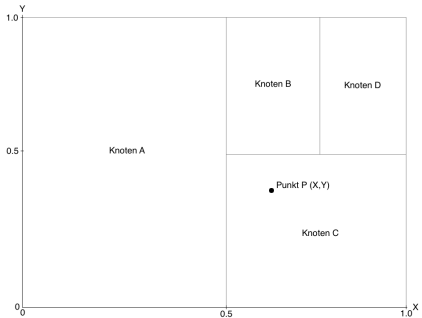
Complexity?

Additional Questions to ask

- Security against Eclipse attack?
- Survivability of DoS attack?
- Maintenance operation cost & required frequency?
- Latency? ($\neq$ number of hops!)
- Data persistence?

Content Addressable Network: CAN

- routing table: neighbours in *d*-dimensional torus space
- lookup: forward to closest peer
- join: lookup own peer identity to find join position, split quadrant (data areas) with existing peer
- leave: assign quadrant space to neighbour (s)

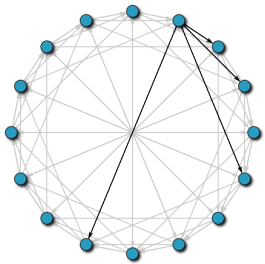


Interesting CAN properties

- CAN can do range queries along $\leq n$ dimensions
- CAN's peers have $2d$ connections (independent of network size)
- CAN routes in $O(d\sqrt[d]{n})$

Chord

- routing table: predecessor in circle and at distance 2^i, plus r successors
- lookup: forward to closest peer (peer ID after key ID)
- join: lookup own peer identity to find join position, use neighbor to establish finger table, migrate data from respective neighbour
- leave: join predecessor with successor, migrate data to respective neighbour, periodic stabilization protocol takes care of finger updates

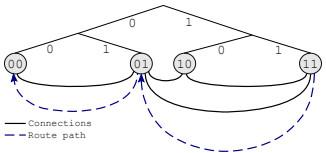


Interesting Chord properties

- Simple design
- $\log_2 n$ routing table size
- $\log_2 n$ lookup cost
- Asymmetric, inflexible routing tables

Kademlia

- routing table: 2^{160} buckets with k peers at XOR distance 2^i
- lookup: iteratively forward to α peers from the “best” bucket, selected by latency
- join: lookup own peer identity, populate table with peers from iteration
- maintenance: when interacting with a peer, add to bucket if not full; if bucket full, check if longest-not-seen peer is live first
- leave: just drop out



Interesting Kademlia properties

- XOR is a symmetric metric: connections are used in both directions
- α replication helps with malicious peers and churn
- Iterative lookup gives initiator much control,
- Lookup helps with routing table maintenance
- Bucket size trade-off between routing speed and table size
- Iterative lookup is a trade-off:
  - good UDP (no connect cost, initiator in control)
  - bad with TCP (very large number of connections)

Part III: Routing in the Dark

Motivation

- Efficient fully decentralized routing in restricted-route topologies is important:
  - Friend-to-friend (F2F) networks (“darknets”)
  - WiFi ad-hoc and sensor networks
  - Unstructured networks
- Clarke & Sandberg claim to achieve $O(\log n)$ routing in the dark (Freenet 0.7)
- Is this new routing protocol reasonably resistant against attacks?

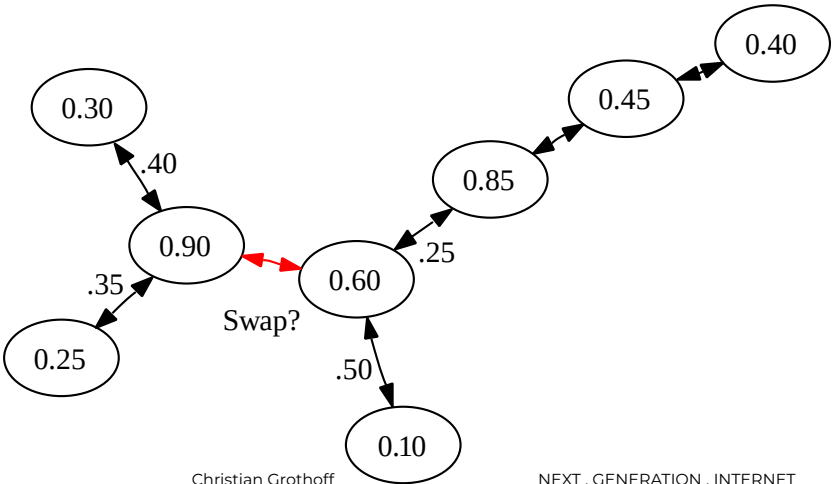
Freenet 101

- Freenet is a 'anonymous' peer-to-peer network
- Overlay based on cyclic address space of size 2^{32}
- Nodes have a constant set of connections (F2F)
- All data identified by a key (modulo 2^{32})
- Data assumed to be stored at closest node
- Routing uses depth-first traversal in order of proximity to key

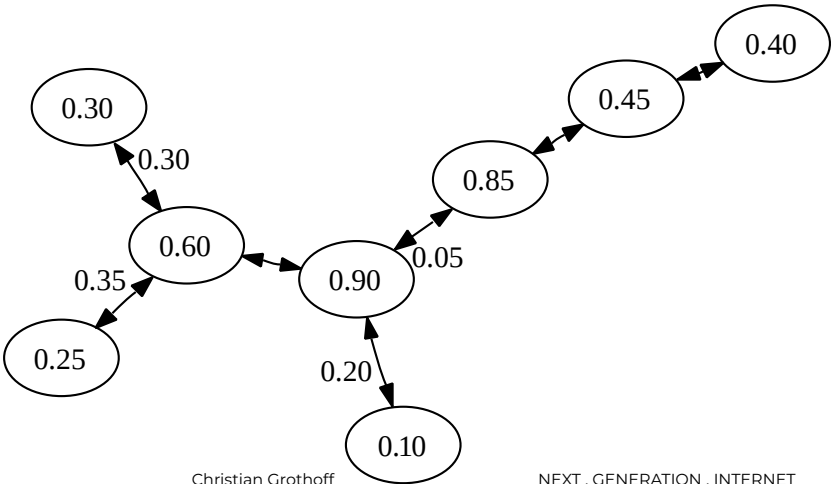
Routing in the Dark

- Small world network assumption
  - Sparsely connected graph
  - There exists a short path (O(log N)) between any pair of nodes
  - Common real world phenomenon (Milgram, Watts & Strogatz)
- Freenet's routing algorithm attempts to find short paths
  - Uses locations of nodes to determine proximity to target
  - Uses swapping of locations to structure topology

Swap Example



Result of Swap



Location Swapping

- Nodes swap locations to improve routing performance
- Each connected pair of nodes (a, b) computes:

$$P_{a,b} := \frac{\prod_{(a,o)\in E} |L_a - L_o| \cdot \prod_{(b,p)\in E} |L_b - L_p|}{\prod_{(a,o)\in E} |L_b - L_o| \cdot \prod_{(b,p)\in E} |L_a - L_p|} \tag{1}$$

- If $P_{a,b} \geq 1$ the nodes swap locations
- Otherwise they swap with probability $P_{a,b}$

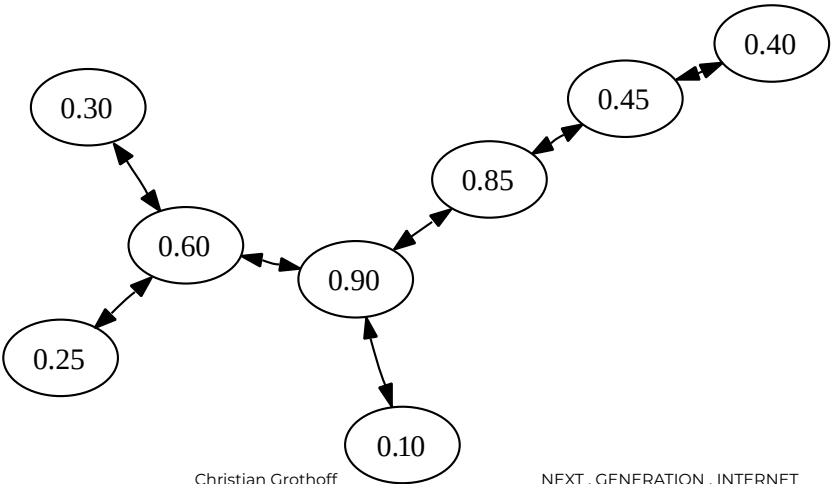
Routing of GET Requests

GET requests are routed based on peer locations and key:

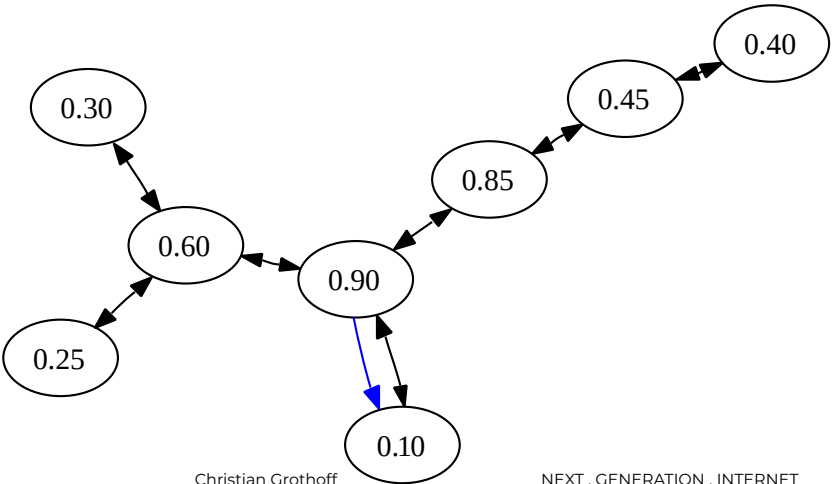
1. Client initiates GET request
2. Request routed to neighbor with closest location to key
3. If data not found, request is forwarded to neighbors in order of proximity to the key
4. Forwarding stops when data found, hops-to-live reaches zero or identical request was recently forwarded (to avoid circular routing)

$\Rightarrow$ Depth-first routing in order of proximity to key.

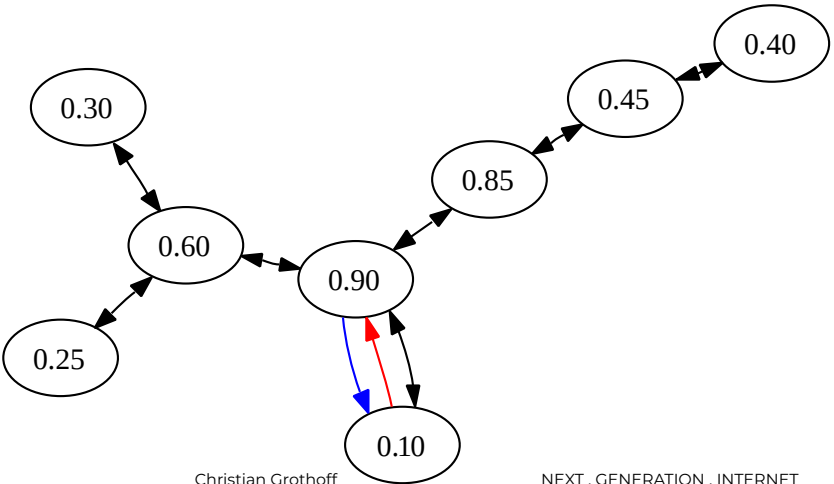
GET 1/7



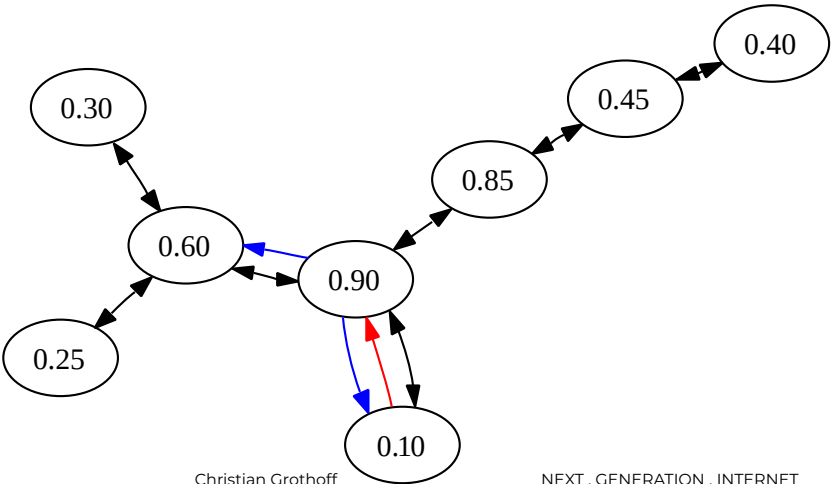
GET 2/7



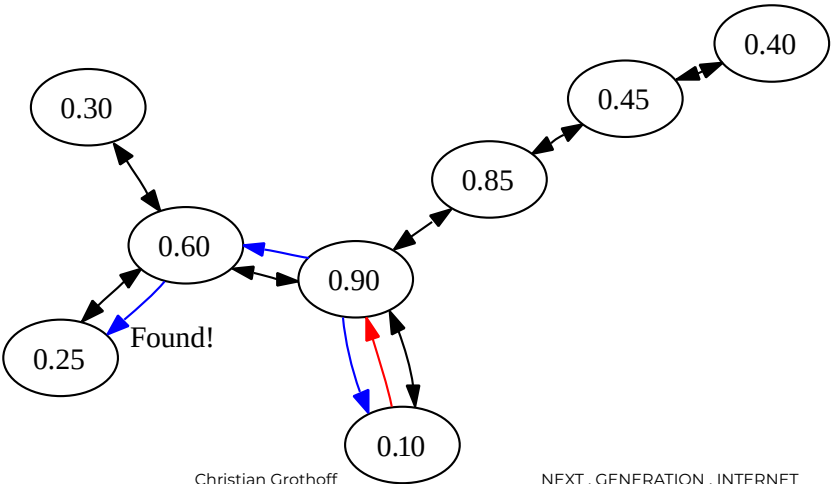
GET 3/7



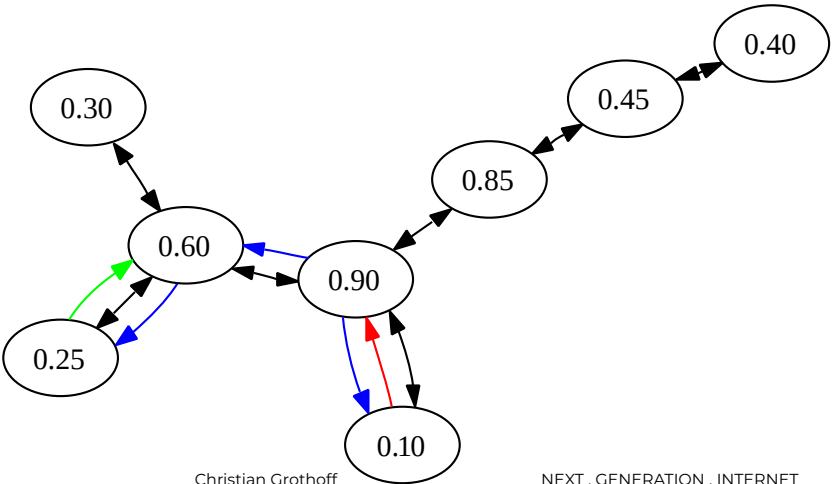
GET 4/7



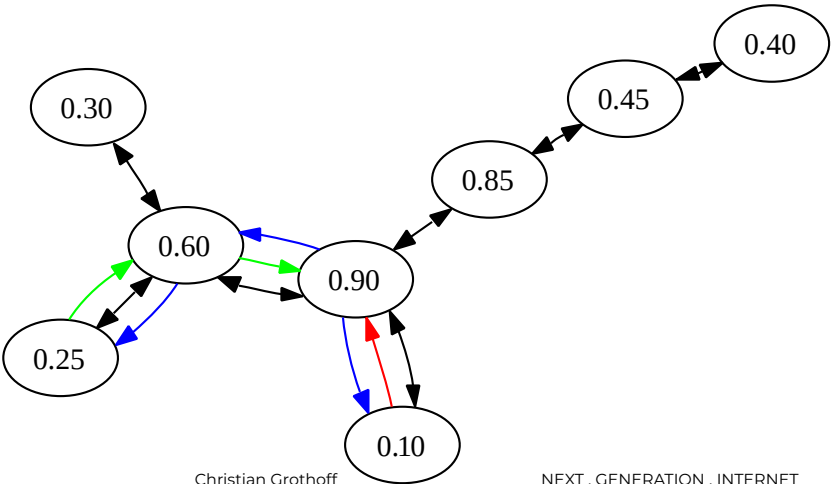
GET 5/7



GET 6/7



GET 7/7

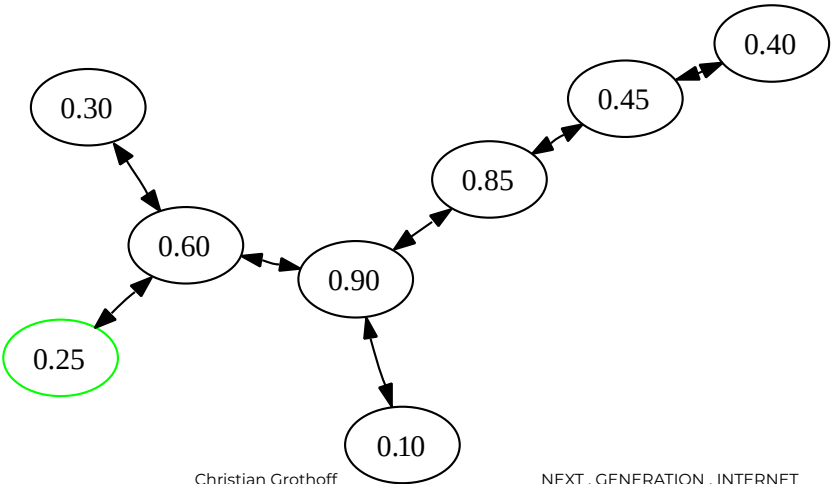


PUT Requests

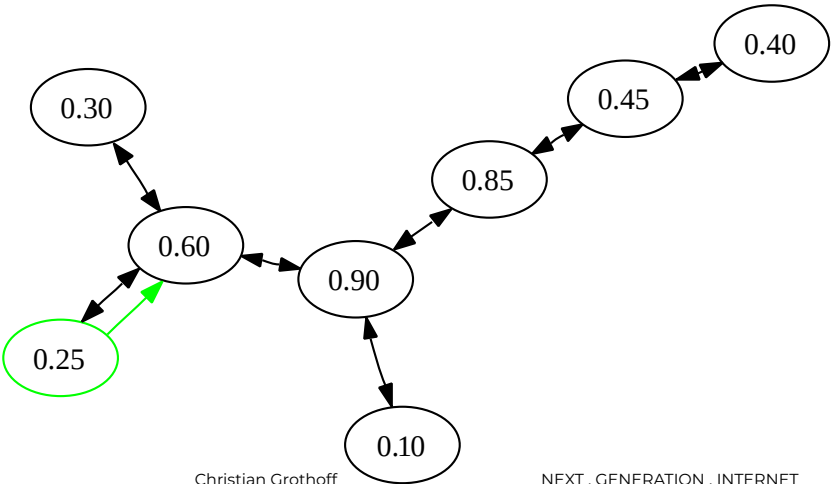
PUT requests are routed the same as GET requests:

1. Client initiates PUT requests
2. Request routed to neighbor closest to the key
3. If receiver has any peer whose location is closer to the key, request is forwarded
4. If not, the node resets the hops-to-live to the maximum and sends the put request to all of its' neighbors
5. Routing continues until hops-to-live reaches zero (or node has seen request already)

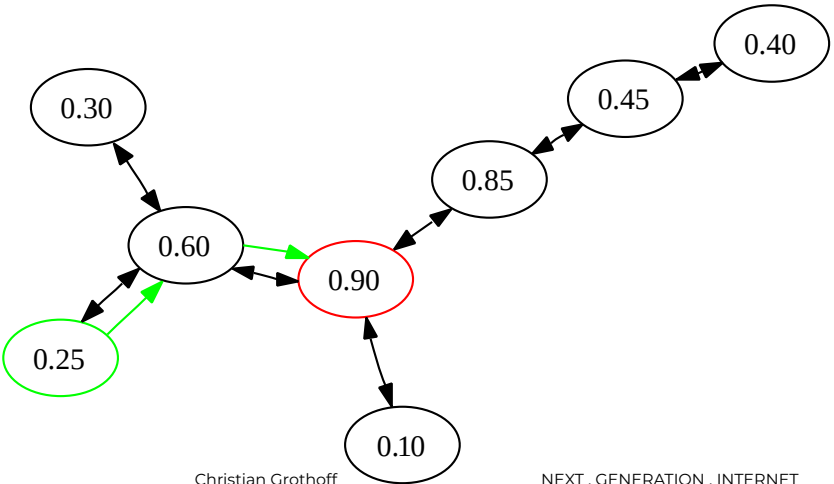
Put Example



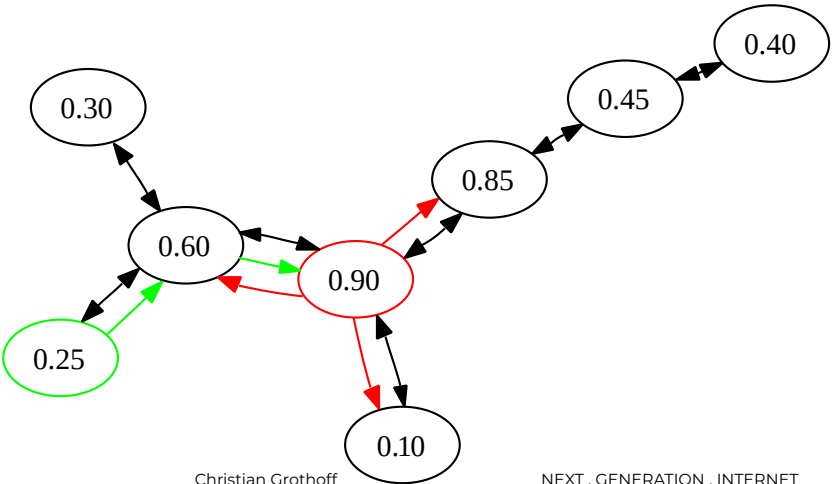
Put Example 1/3



Put Example 2/3



Put Example 3/3



How to attack this?

Basic Idea for the Attack

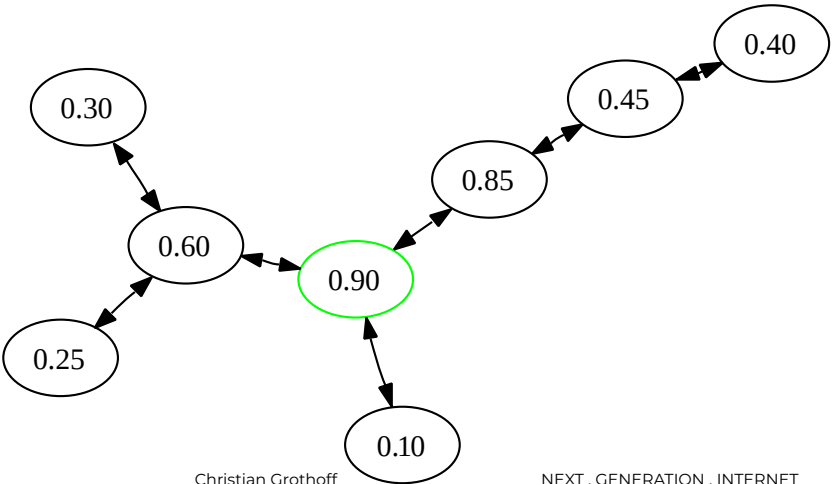
- Freenet relies on a balanced distribution of node locations for data storage
- Reducing the spread of locations causes imbalance in storage responsibilities
- Peers cannot verify locations in swap protocol, including location(s) they may receive

$\Rightarrow$ use swap protocol to reduce spread of locations!

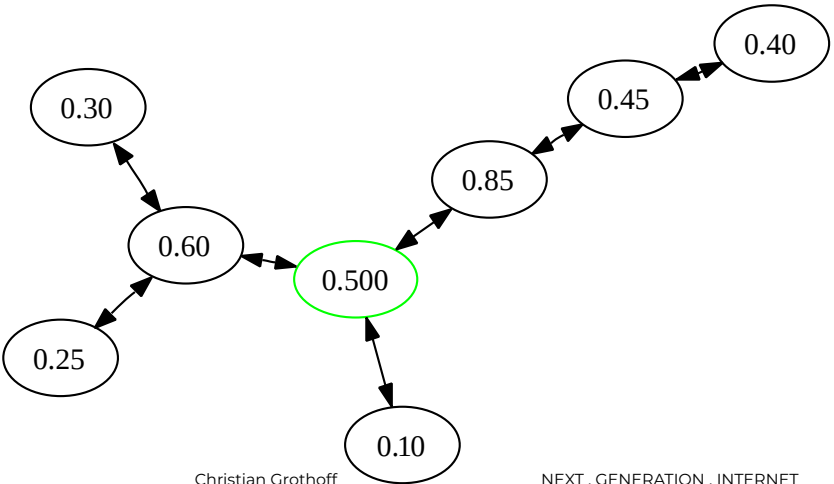
Attack Details

- Initialize malicious nodes with a specific location
- If a node swaps with the malicious node, the malicious node resets to the initial location (or one very close to it)
- This removes the “good” node location and replaces it with one of the malicious nodes choosing
- Each time any node swaps with the malicious node, another location is removed and replaced with a “bad” location
- Bad location(s) spread to other nodes through normal swapping behavior
- Over time, the attacker creates large clusters of nodes around a few locations

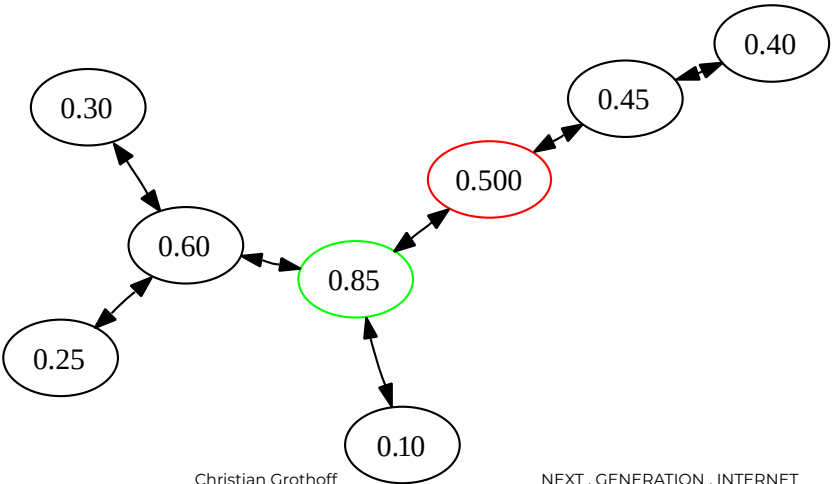
Attack Example 1/11



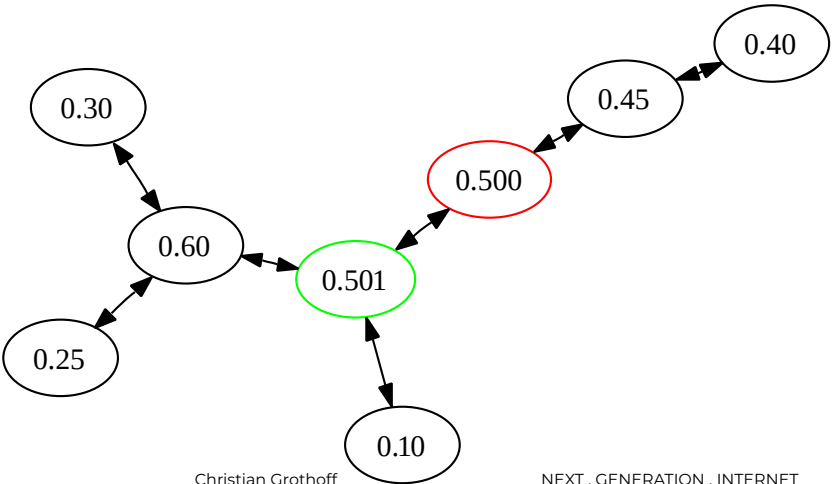
Attack Example 2/11



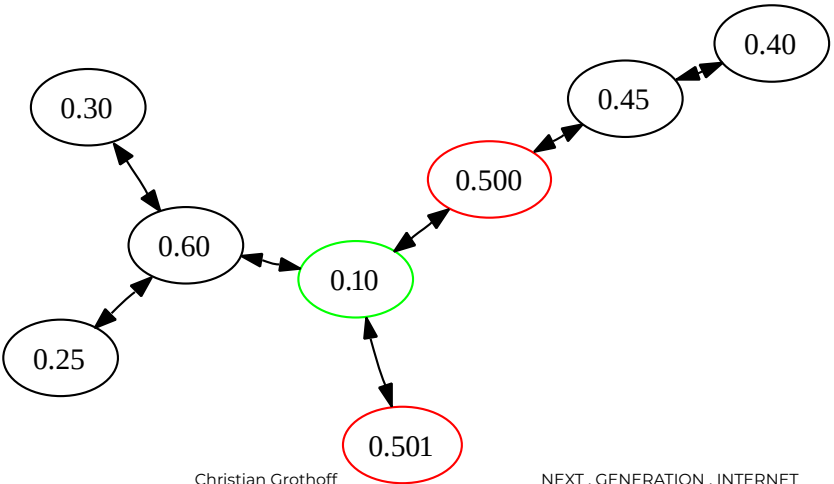
Attack Example 3/11



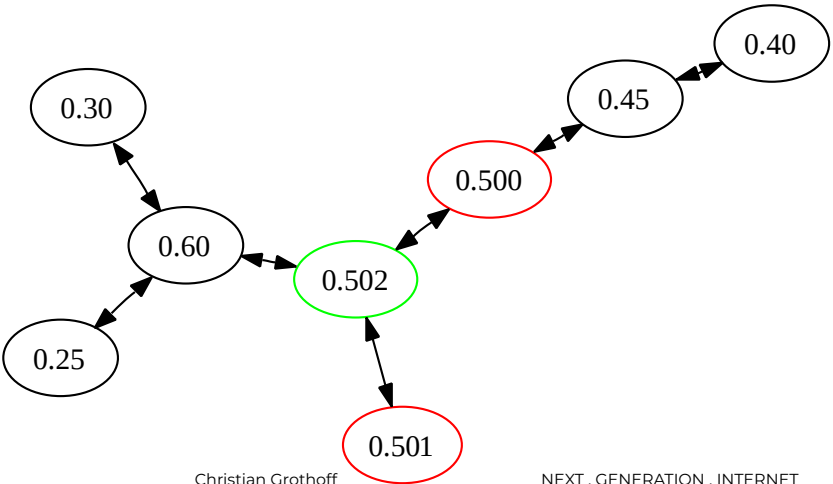
Attack Example 4/11



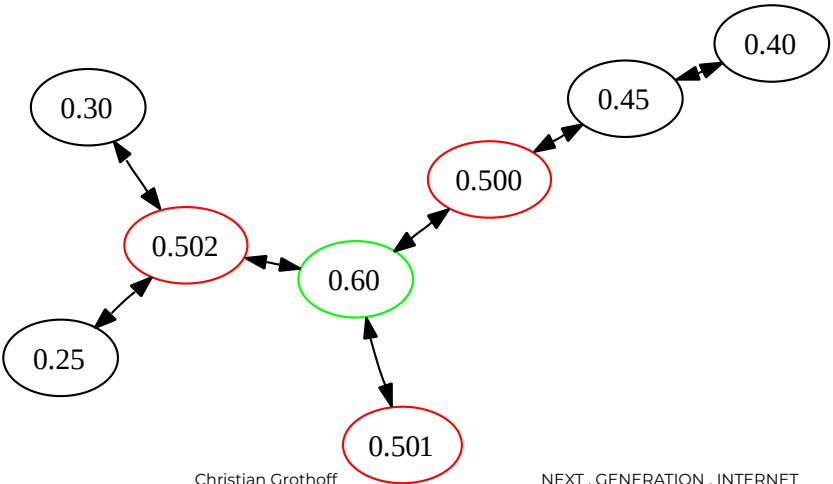
Attack Example 5/11



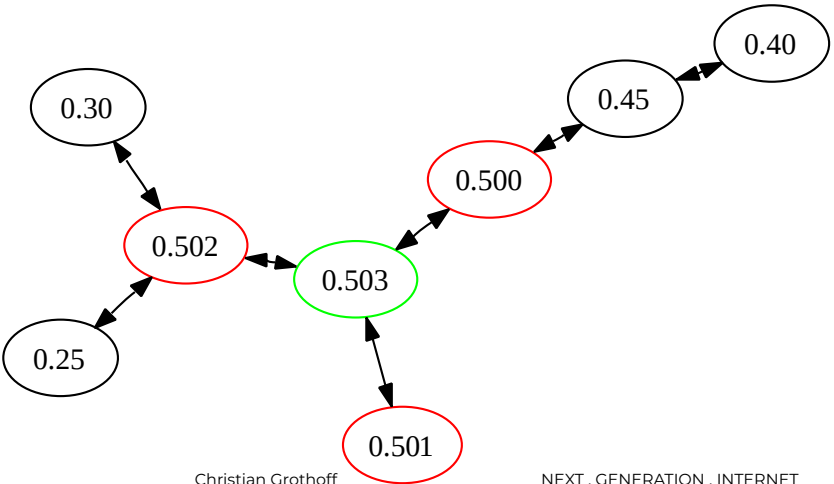
Attack Example 6/11



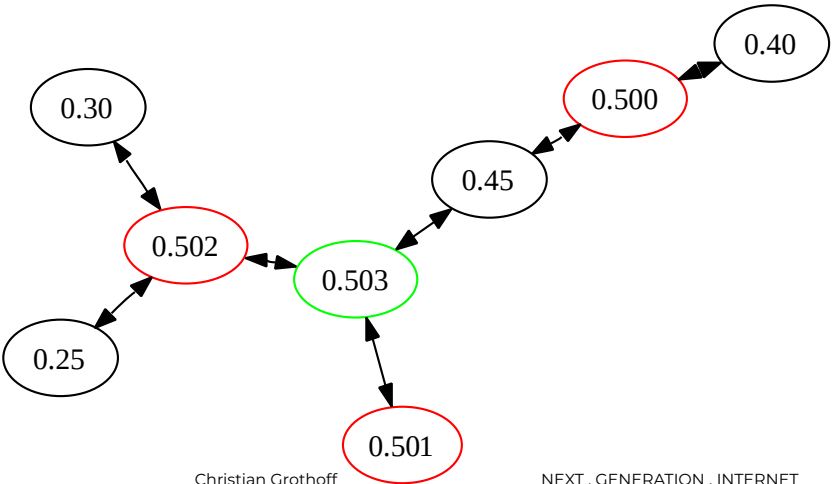
Attack Example 7/11



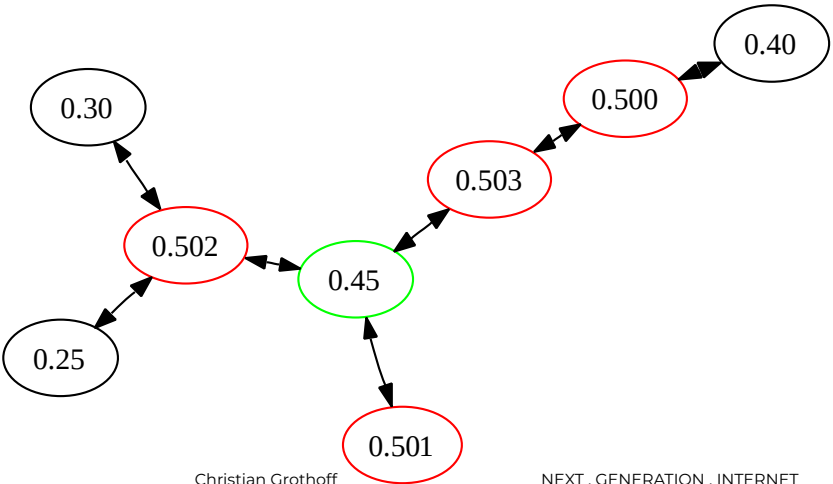
Attack Example 8/11



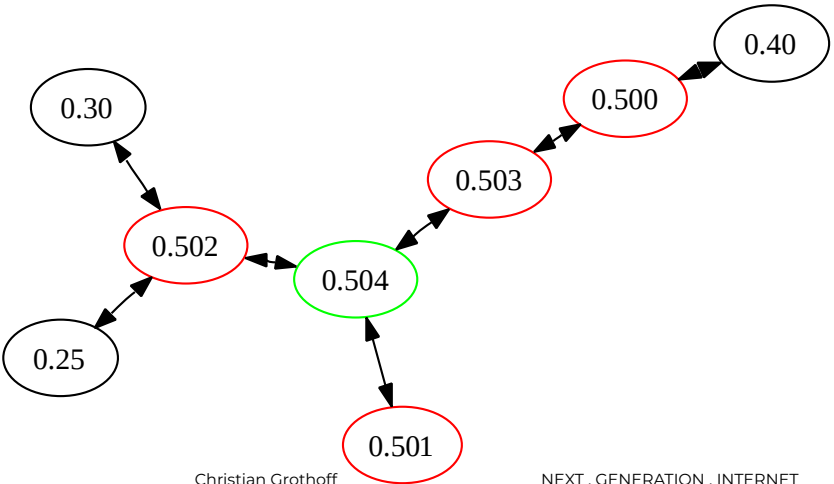
Attack Example 9/11



Attack Example 10/11



Attack Example 11/11



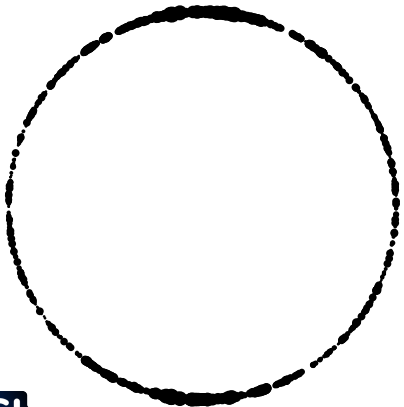
Attack Implementation

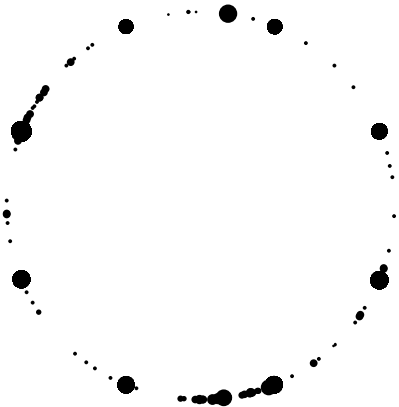
- Malicious node uses Freenet's codebase with minor modifications
- Attacker does not violate the protocol in a detectable manner
- Malicious nodes behave as if they had a large group of friends
- Given enough time, a single malicous node can spread bad locations to most nodes
- Using multiple locations for clustering increases the speed of penetration

Experimental Setup

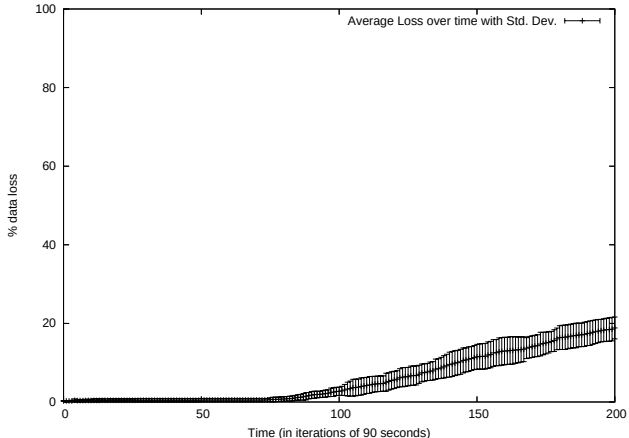
- Created testbed with 800 Freenet nodes
- Topology corresponds to Watts & Strogatz small world networks
- Instrumentation captures path lengths and node locations
- Content is always placed at node with closest location
- Nodes have bounded storage space

Dispersion Example with 800 Nodes

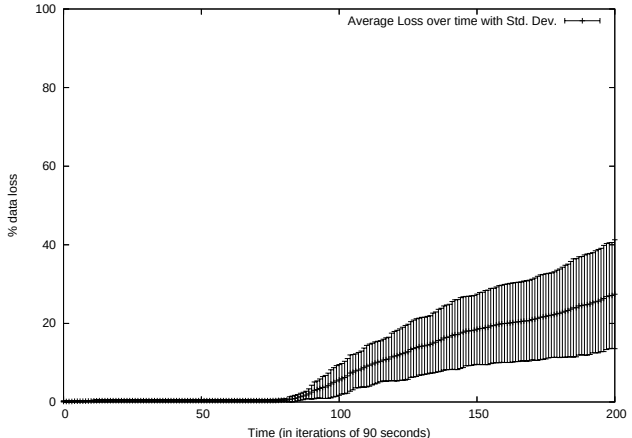




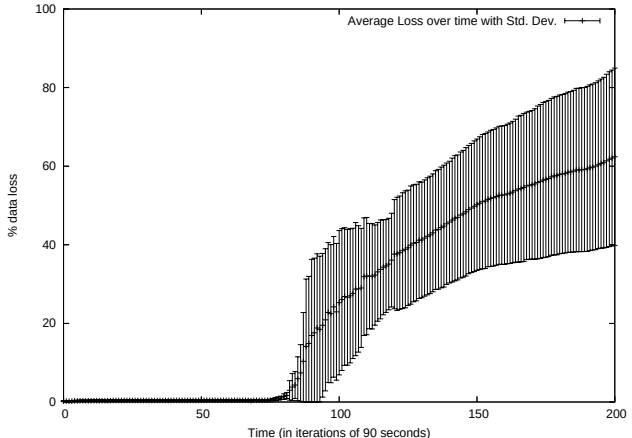
Data Loss Example (2 attack nodes)



Data Loss Example (4 Attack nodes)



Data Loss Example (8 Attack nodes)



How to protect against this?

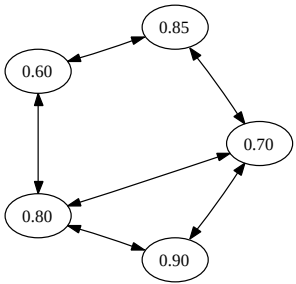
- Check how frequently a node swaps similar locations?
- Limit number of swaps with a particular peer?
- Determine a node is malicious because its' location is *too* close?
- Periodically reset all node locations?
- Secure multiparty computation for swaps?

In F2F networks, you can never be sure about the friends of your friends!

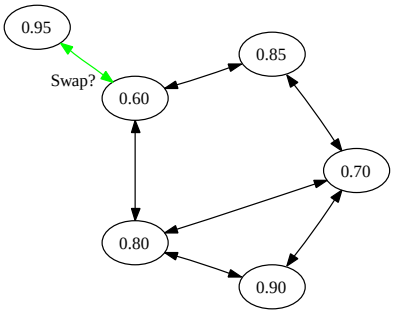
Churn

- Leave join churn
  - Nodes are not constantly in the network
  - They leave for some period of time and then come back into the network
- Join leave churn
  - Nodes join the network for a time, then disconnect permanently
- This also causes load imbalance similar to our attack

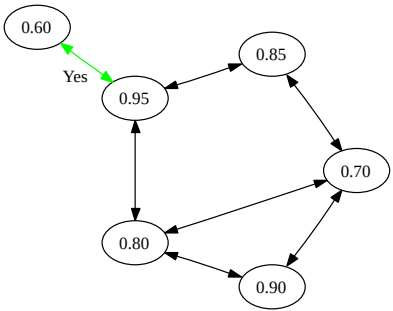
Churn Example 1/13



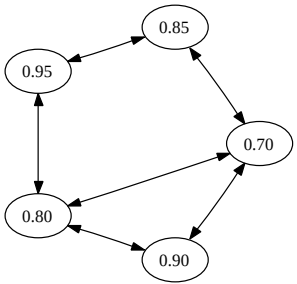
Churn Example 2/13



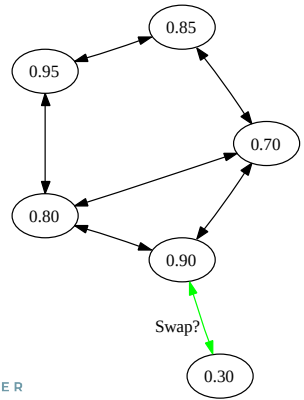
Churn Example 3/13



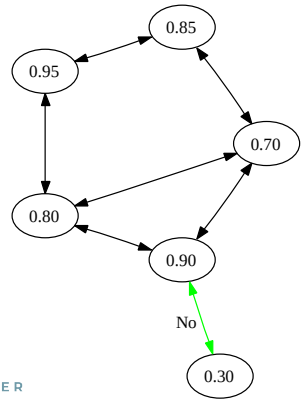
Churn Example 4/13



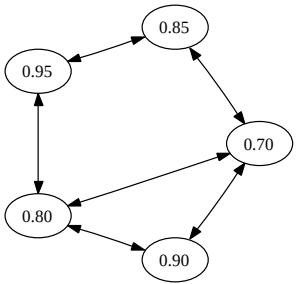
Churn Example 5/13



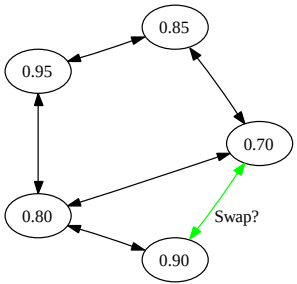
Churn Example 6/13



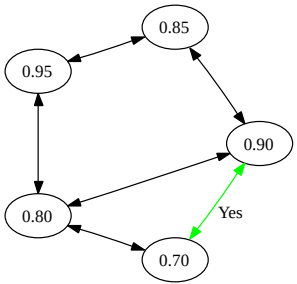
Churn Example 7/13



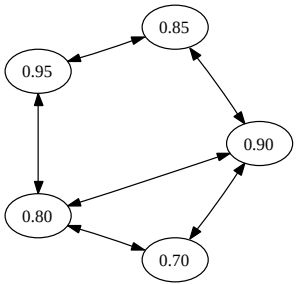
Churn Example 8/13



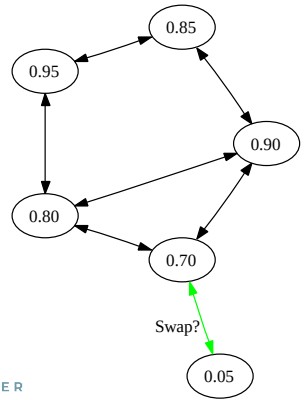
Churn Example 9/13



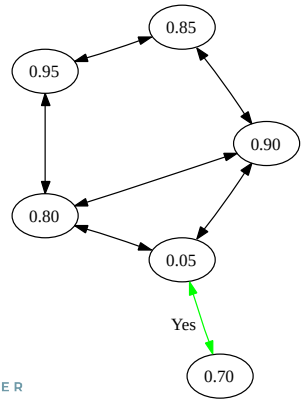
Churn Example 10/13



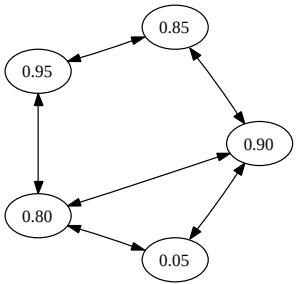
Churn Example 11/13



Churn Example 12/13



Churn Example 13/13



Churn Simulation

- Created stable core of nodes
- Simulated join-leave churn, let network stabilize
- Ran exactly the native swap code
- Repeat *n* number of times
- Revealed drastic convergence to single location

Conclusion

- Freenet's routing algorithm is not robust
- Adversaries can easily remove most of the content
- Attack exploits location swap, where nodes trust each other
- Swap is fundamental to the routing algorithm
- Natural churn causes similar results

Part IV: Hard Problems

Ryge's Triangle

Ryge's Triangle postulates three key management goals for a system associating cryptographic keys with addresses or names:

- Non-interactive: the system should require no user interface
- Flexible: addresses/names can be re-used by other participants
- Secure: the system is secure against active attackers

Ryge's triangle says that one can only have two of the three.

Limits on authentication

Theorem (Boyd's Theorem I)

"Suppose that a user has either a confidentiality channel to her, or an authentication channel from her, at some state of the system. Then in the previous state of the system such a channel must also exist. By an inductive argument, such a channel exists at all previous states."

Theorem (Boyd's Theorem II)

"Secure communication between any two users may be established by a sequence of secure key transfers if there is a trusted chain from each one to the other."

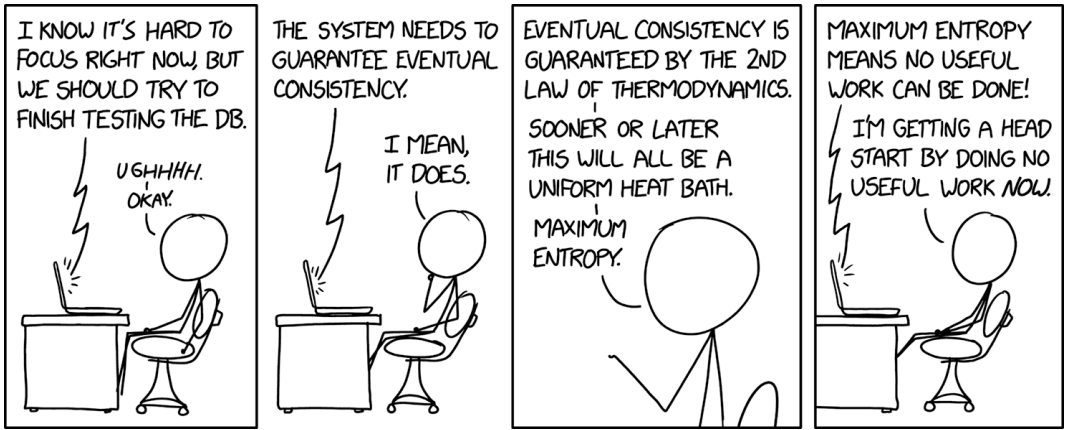
Solution space: Zfone Authentication (ZRTP) [18]

Idea: combine human interaction proof and baby duck approach:

- *A* and *B* perform Diffie-Hellman exchange
- Keying material from previous sessions is used (duckling)
- Short Authentication String (SAS) is generated (hash of DH numbers)
- Both users read the SAS to each other, recognize voice

$\Rightarrow$ ZRTP foils standard man-in-the-middle attack.

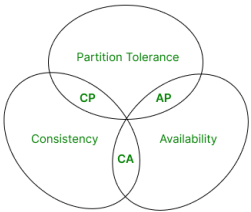
https://xkcd.com/2315/



The CAP theorem [7]

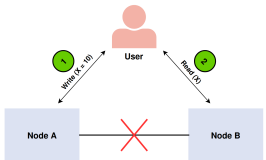
No distributed system can be *consistent*, *available* and *partition tolerant* at the same time.

- Consistency: A *read* sees the changes made by all previous *writes*
- Availability: *Reads* and *writes* always succeed
- Partition tolerance: The system operates even when network connectivity between components is broken



An informal proof

Consider two nodes of a partitioned system storing a variable *X*. The system processes write (1) and read (2) requests:



In this case, the distributed system has two options:

- It can fail at one of the requests, breaking the system's availability, or
- It can execute both requests, returning a stale value from the read request and breaking the system's consistency

PARELC

The PACELC theorem extends the CAP theorem:

- In case of network partitioning (P) in a distributed computer system one has to choose between
  - availability (A) and
  - consistency (C),
- else (E), even when the system is running normally in the absence of partitions, one has to choose between
  - latency (L) and
  - consistency (C)

Basically, even in the absence of partitioning, a trade-off between consistency and latency exists.

Digital offline payments ...

... are incompatible with the CAP theorem

- Offline capabilities are often cited as a requirement for digital payments by central banks
- All implementations must either use restrictive hardware elements and/or introduce counterparty risk.

⇒ Permanent offline features weaken a digital payment solution (privacy, security)

Digital offline payments …

… are incompatible with the CAP theorem

- Offline capabilities are often cited as a requirement for digital payments by central banks
- All implementations must either use restrictive hardware elements and/or introduce counterparty risk.

⇒ Permanent offline features weaken a digital payment solution (privacy, security)

Nevertheless, the ECB *claims* that the offline Digital Euro will work:

- Offline (partitioned from the Internet), and
- Transitive (*A* pays *B*, then *B* pays *C*, etc. — all while offline)
- with full cash-like privacy/anonymity for participants

Heise Forum: Posting 43921013

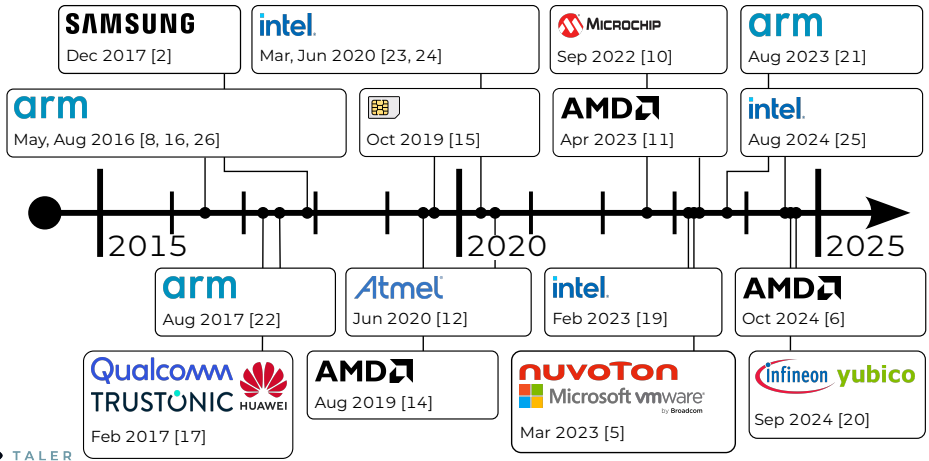
"EZB: Digitaler Euro benötigt Secure Element des iPhones" –Heise.

"Ohne Lösung des Single-Spending-Problems wird das nichts":

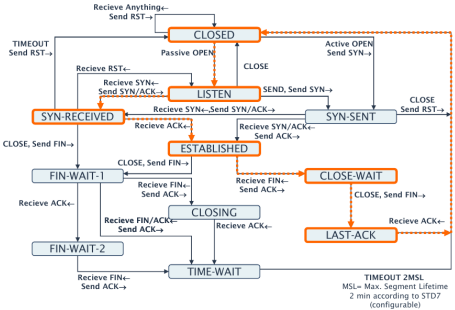
> *"Im klassischen Geldverkehr gibt es ja das Problem, dass man jeden Euro nur einmal ausgeben kann. Das ist einer der großen Nachteile von Bargeld. Eine digitale Währung hätte den großen Vorteil, dass man jeden Euro mehrfach ausgeben könnte, und somit mehr Geld erschaffen könnte. Das wäre der große Vorteil gegenüber Bargeld und Kartenzahlungen. So lange das nicht geht, ist es im Prinzip wie DRM, ein Versuch, mit viel Aufwand, irgendwie die Zeit zurück zu drehen um alte Geschäftsmodelle 1:1 am Leben zu erhalten." –Casandro*

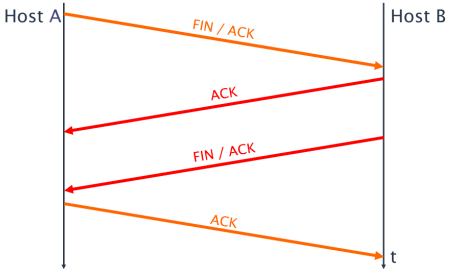
Hardware to the rescue?

Only if history has nothing to teach us!

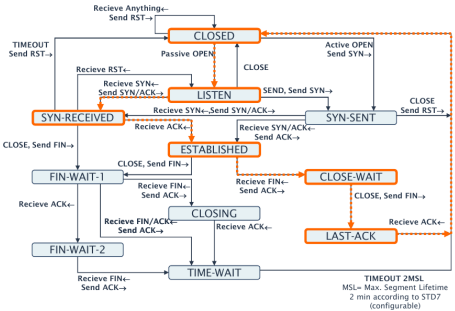


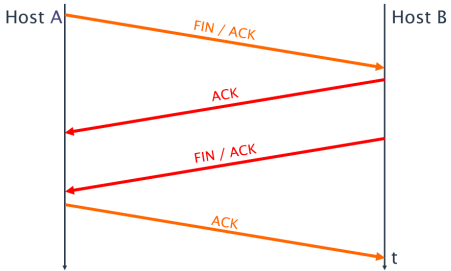
Would secure hardware suffice?





Would secure hardware suffice?

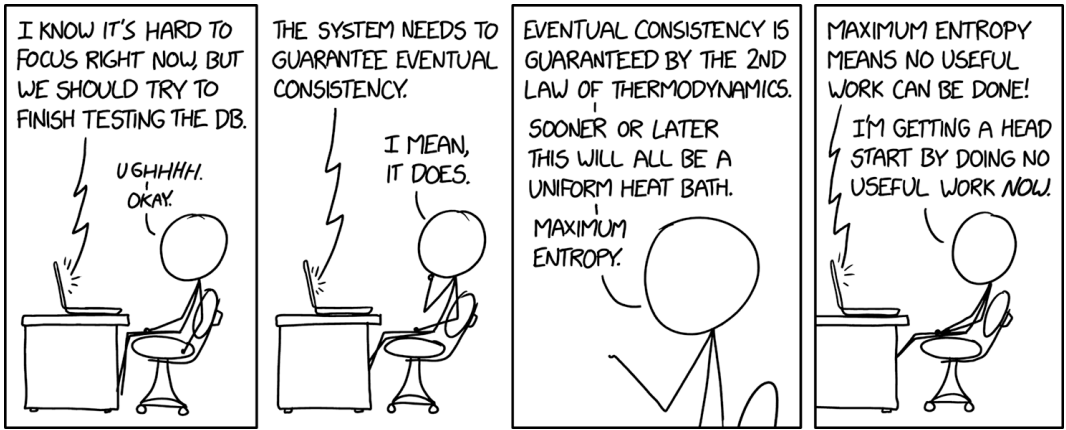




Not only TCP teaches us:

In any communication, someone must have the last word.

https://xkcd.com/2315/



GNU Taler vs. Twint

When can we fix this?

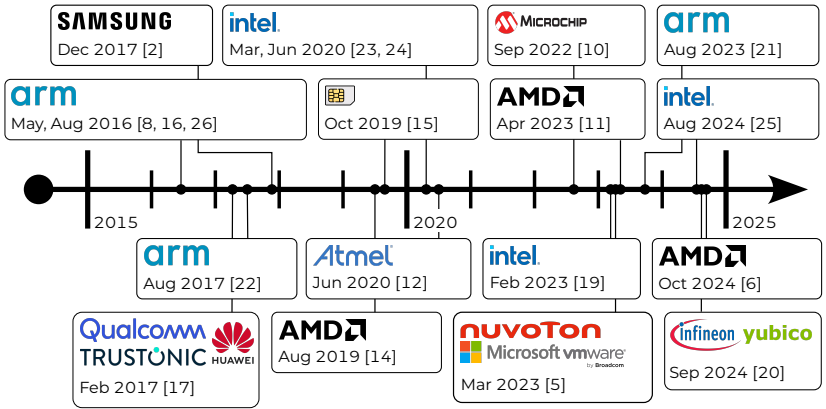
Offline payments

Digitaler Euro — Offline?

Many central banks today demand offline capabilities for CBDCs.

Digitaler Euro — Offline?

Many central banks today demand offline capabilities for CBDCs.



A Scenario

God is offline, but customer pays online

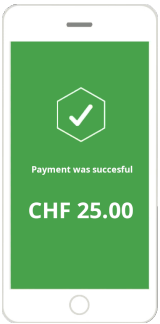
Typical Payment Process

All equivalent: Twint, PayPal, AliPay, PayTM

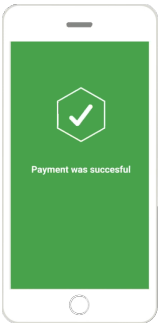
(C) Twint, 2023

Secure Payment ...

Everything green?

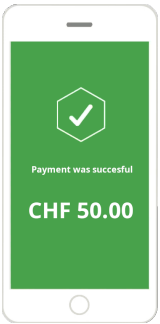


Exploit “Code”
Programming optional

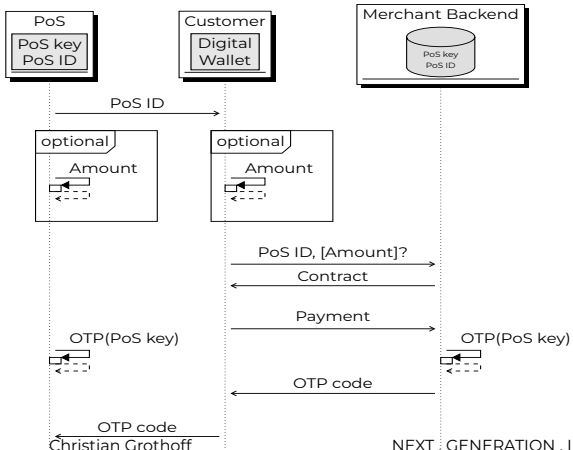


“Customers” *love* Twint …

Daily non-business for shops



Partially Offline Payments with GNU Taler [9]



References I

Edsger W. Dijkstra.
Self-stabilizing systems in spite of distributed control.
Commun. ACM, 17(11):643–644, nov 1974.

M. Dorjmyagmar, M. Kim, and H. Kim.
Security analysis of samsung knox.
In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 550–553, 2017.

References II

John Douceur.
The Sybil Attack.
In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, March 2002.

Nathan S. Evans, Chris GauthierDickey, and Christian Grothoff.
Routing in the dark: Pitch black.
In *23rd Annual Computer Security Applications Conference (ACSAC 2007)*, pages 305–314. IEEE Computer Society, December 2007.

References III

Francisco Falcon.
Vulnerabilities in the tpm 2.0 reference implementation code.
https://blog.quarkslab.com/vulnerabilities-in-the-tpm-20-reference-implementation-code.html, March 2023.

References IV

Stefan Gast, Hannes Weissteiner, Robin Leander Schröder, and Daniel Gruss.
Counterseveillance: Performance-counter attacks on amd sev-snp.
In *Network and Distributed System Security (NDSS) Symposium 2025*, February 2025.
Network and Distributed System Security Symposium 2025 : NDSS 2025, NDSS 2025 ; Conference date: 23-02-2025 Through 28-02-2025.

References V


Seth Gilbert and Nancy Lynch.
Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services.
SIGACT News, 33(2):51–59, June 2002.


R. Guanciale, H. Nemati, C. Baumann, and M. Dam.
Cache storage channels: Alias-driven attacks and verified countermeasures.
In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 38–55, May 2016.

References VI

- Priscilla Huang, Emmanuel Benoist, Christian Grothoff, and Sebastian Javier Marchano.
  Practical offline payments using one-time passcodes.
  SUERF Policy Briefs, (622), June 2023.
- Olivier Hériveaux.
  Triple exploit chain with laser fault injection on a secure element.
  In *2022 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)*, pages 9–17, 2022.

References VII

- Hans Niklas Jacob, Christian Werling, Robert Buhren, and Jean-Pierre Seifert.
  faultpm: Exposing amd ftpms' deepest secrets.
  In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 1128–1142. IEEE, 2023.
- Jan Jancar, Vladimir Sedlacek, Petr Svenda, and Marek Sys.
  Minerva: The curse of ECDSA nonces (systematic analysis of lattice attacks on noisy leakage of bit-length of ECDSA nonces).
  IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(4):281–308, 2020.

References VIII

Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright.
Timing attacks in low-latency mix-based systems.
In *Proceedings of Financial Cryptography (FC '04)*, pages 251–265, February 2004.

Mengyuan Li, Yinqian Zhang, Zhiqiang Lin, and Yan Solihin.
Exploiting unprotected i/o operations in amd's secure encrypted virtualization.
In *USENIX Security Symposium*, 2019.

References IX

- Adaptive Mobile Security Limited.
  Simjacker technical report.
  https://www.enea.com/info/simjacker/, 2019.
- Moritz Lipp, Daniel Gruss, Raphael Spreitzer, Clémentine Maurice, and Stefan Mangard.
  Armageddon: Cache attacks on mobile devices.
  In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 549–564, USA, 2016. USENIX Association.

References X

- Aravind Machiry, Eric Gustafson, Chad Spensky, Christopher Salls, Nick Stephens, Ruoyu Wang, Antonio Bianchi, Yung Ryn Choe, Christopher Kruegel, and Giovanni Vigna.
  Boomerang: Exploiting the semantic gap in trusted execution environments.
  In *NDSS*, 2017.
- Laurianne McLaughlin.
  Philip zimmermann on what's next after pgp.
  IEEE Security & Privacy, 4(1):10–13, 2006.

References XI

Joseph Nuzman.
Cve-2022-38090: Improper isolation of shared resources in some intel(r) processors when using intel(r) software guard extensions may allow a privileged user to potentially enable information disclosure via local access.
https://www.cve.org/CVERecord?id=CVE-2022-38090, February 2023.

Thomas Roche.
Eucleak: Side-channel attack on the yubikey 5 series—revealing and breaking infineon ecdsa implementation on the way.
https://ninjalab.io/eucleak/, September 2024.

References XII

Xhani Marvin Saß, Richard Mitev, and Ahmad-Reza Sadeghi.
Oops..! i glitched it again! how to Multi-Glitch the Glitching-Protections on ARM TrustZone-M.
In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6239–6256, 2023.

References XIII

Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo.
Clkscrew: Exposing the perils of security-oblivious energy management.
In *Proceedings of the 26th USENIX Conference on Security Symposium*, SEC'17, page 1057–1074, USA, 2017. USENIX Association.

References XIV

Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lipp, Marina Minkin, Daniel Genkin, Yarom Yuval, Berk Sunar, Daniel Gruss, and Frank Piessens.
LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection.
In *41th IEEE Symposium on Security and Privacy (S&P'20)*, March 2020.

References XV

Stephan van Schaik, Andrew Kwong, Daniel Genkin, and Yuval Yarom.

SGAxe: How SGX fails in practice.
https://sgaxeattack.com/, June 2020.

Luca Wilke, Florian Sieck, and Thomas Eisenbarth.
Tdxdown: Single-stepping and instruction counting attacks against intel tdx.
In *ACM CCS 2024*, 2024.

References XVI

Ning Zhang, Kun Sun, Deborah Shands, Wenjing Lou, and Y Thomas Hou.
Truspy: Cache side-channel information leakage from the secure world on arm devices.
IACR Cryptol. ePrint Arch., 2016:980, 2016.

Acknowledgements

Co-funded by the European Union (Project 101135475).

Co-funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
State Secretariat for Education,
Research and Innovation SERI

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union.

Neither the European Union nor the granting authority can be held responsible for them.