Distributed Systems

Christian Grothoff

Berner Fachhochschule

April 26, 2019

Learning objectives

- Fallacies of distributed computing
- Boyd's theorem
- The CAP theorem and the Blockchain trilemma
- Zooko's triangle
- Self stabilization
- Attacks and solutions
- Routing with distributed hash tables (DHTs)

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

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) [4]

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.

CAP Theorem [2]

No distributed system can *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

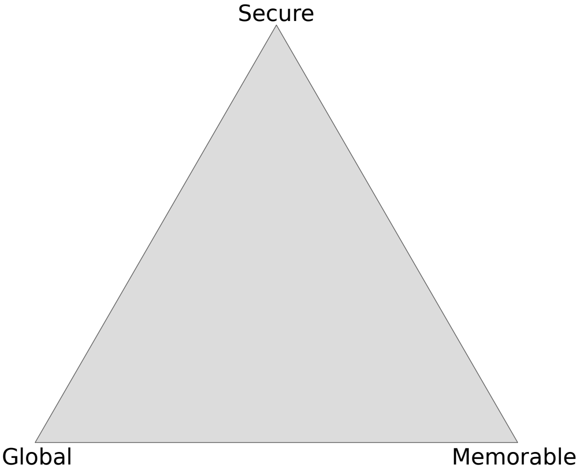
Blockchain Trilemma

Blockchains claim to achieve three properties:

- Decentralization: there are many participants, and each participant only needs to have a small amount of resources, say $O(c)$
- Scalability: the system scales to $O(n) > O(c)$ transactions)
- Security: the system is secure against attackers with $O(n)$ resources

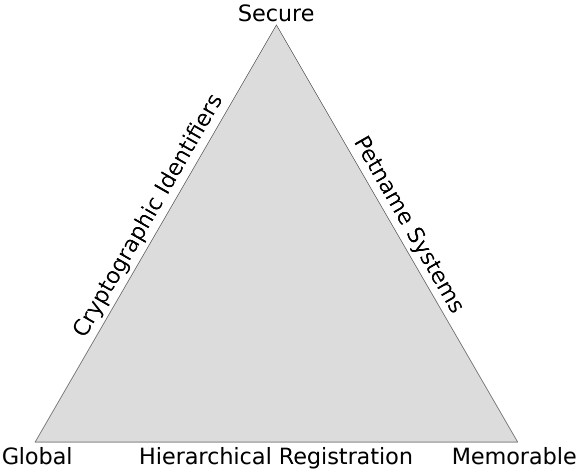
The Blockchain trilemma is that one can only have two of the three.

Zooko's Triangle



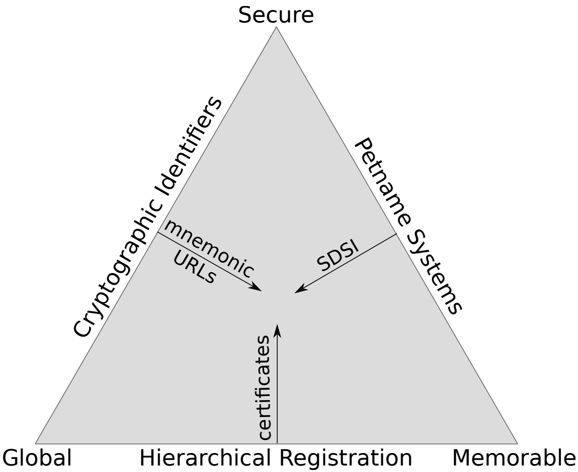
A name system can only fulfill **two**!

Zooko's Triangle



DNS, ".onion" IDs and `/etc/hosts/` are representative designs.

Zooko's Triangle



DNSSEC security is broken by design (adversary model!)

Self stabilization (Dijkstra 1974)

- 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 7071!

Sybil Attack

Background:

- Ancient Greece: Sybils were prophetesses that phrophesized 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.

Sybil Attack

Background:

- Ancient Greece: Sybils were prophetesses that phrophesized 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 [1]:

- 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.

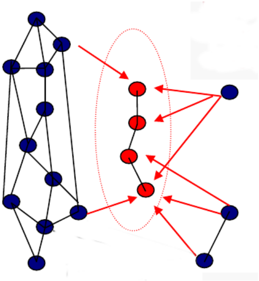
Defenses against Sybil Attacks

- 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: 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

$\Rightarrow$ Details depend on overlay structure

Eclipse Attack: Defenses

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

Poisoning Attacks

Nodes provide false information:

- wrong routing tables
- wrong meta data
- wrong performance measurements

Timing Attacks [3]

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...)

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

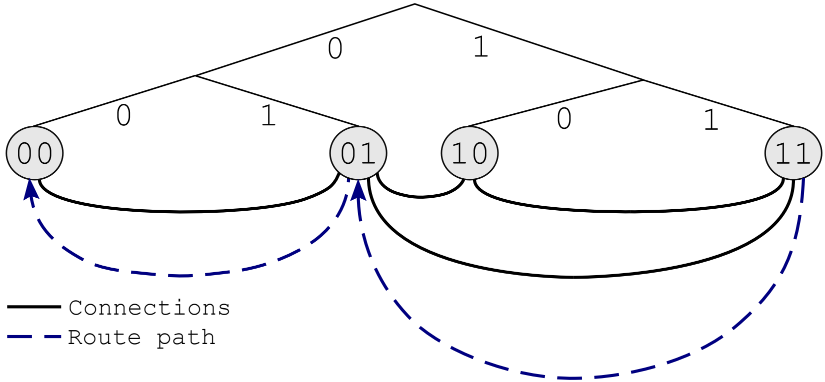
Kademlia

- 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)

Kademlia



Exam reminder

1. Submit your code ≈ 1 week before the oral exams
2. In that case, you will be asked questions about the project:
   - What your project does (explain to co-examiners!)
   - Examination on how it works in depth
   - Critical discussion based on my code review prior to the exam
3. Otherwise, any theory that was taught is fair game

References I

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

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.

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.

References II

Laurianne McLaughlin.
Philip zimmermann on what's next after pgp.
IEEE Security & Privacy, 4(1):10–13, 2006.