The Architecture of the GNUnet: 45 Subsystems in 45 Minutes

Christian Grothoff

Inria Rennes Bretagne Atlantique

10.12.2015

“Never doubt your ability to change the world.” –Glenn Greenwald
The Internet is Broken

- Network generally learns too much
- Insecure defaults and high system complexity
- Centralized Internet infrastructure requires administration:
  - Number resources (IANA)
  - Domain Name System (Root zone)
  - X.509 CAs (HTTPS certificates)
- Administrators have power, and power attracts attackers
- Self-organizing systems aka P2P systems offer a way forward!
## Our Vision (Simplified)

### Internet

<table>
<thead>
<tr>
<th>Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
</tr>
<tr>
<td>DNS/X.509</td>
</tr>
<tr>
<td>TCP/UDP</td>
</tr>
<tr>
<td>IP/BGP</td>
</tr>
<tr>
<td>Ethernet</td>
</tr>
<tr>
<td>Phys. Layer</td>
</tr>
</tbody>
</table>

*GNUnet, Applications, GNU Name System, CADET (Axolotl+SCTP), R5N, N DHT, CORE (OTR), HTTPS/TCP/WLAN/…*
### Our Vision (Simplified)

#### Internet

<table>
<thead>
<tr>
<th>Google</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS/X.509</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP/UDP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP/BGP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethernet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phys. Layer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...HTTPS/TCP/WLAN/...
## Our Vision (Simplified)

### Internet

<table>
<thead>
<tr>
<th>Google</th>
<th>CORE (OTR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS/X.509</td>
<td></td>
</tr>
<tr>
<td>TCP/UDP</td>
<td></td>
</tr>
<tr>
<td>IP/BGP</td>
<td></td>
</tr>
<tr>
<td>Ethernet</td>
<td>HTTPS/TCP/WLAN/...</td>
</tr>
<tr>
<td>Phys. Layer</td>
<td></td>
</tr>
</tbody>
</table>

---

**Décentralisé**

*Institut National de Recherche en Informatique et en Automatique*
Our Vision (Simplified)

Internet

| Google |  
| DNS/X.509 |  
| TCP/UDP |  
| IP/BGP |  
| Ethernet |  
| Phys. Layer |  

|  |  
|  |  
|  |  
|  |  
|  |  
|  |  
| $R^5N$ DHT |  
| CORE (OTR) |  
| HTTPS/TCP/WLAN/... |
Our Vision (Simplified)

### Internet

<table>
<thead>
<tr>
<th>Google</th>
<th>CADET (Axolotl+SCTP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS/X.509</td>
<td>$R^5_N$ DHT</td>
</tr>
<tr>
<td>TCP/UDP</td>
<td>CORE (OTR)</td>
</tr>
<tr>
<td>IP/BGP</td>
<td>HTTPS/TCP/WLAN/...</td>
</tr>
<tr>
<td>Ethernet</td>
<td></td>
</tr>
<tr>
<td>Phys. Layer</td>
<td></td>
</tr>
</tbody>
</table>
Our Vision (Simplified)

### Internet

<table>
<thead>
<tr>
<th>Google</th>
<th>GNU Name System</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS/X.509</td>
<td>CADET (Axolotl† + SCTP)</td>
</tr>
<tr>
<td>TCP/UDP</td>
<td>$R^5 N$ DHT</td>
</tr>
<tr>
<td>IP/BGP</td>
<td>CORE (OTR)</td>
</tr>
<tr>
<td>Ethernet</td>
<td>HTTPS/TCP/WLAN/...</td>
</tr>
<tr>
<td>Phys. Layer</td>
<td></td>
</tr>
</tbody>
</table>
## Our Vision (Simplified)

### Internet

<table>
<thead>
<tr>
<th>Google</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS/X.509</td>
<td>GNU Name System</td>
</tr>
<tr>
<td>TCP/UDP</td>
<td>CADET (Axolotl+SCTP)</td>
</tr>
<tr>
<td>IP/BGP</td>
<td>( R^5N ) DHT</td>
</tr>
<tr>
<td>Ethernet</td>
<td>CORE (OTR)</td>
</tr>
<tr>
<td>Phys. Layer</td>
<td>HTTPS/TCP/WLAN/...</td>
</tr>
</tbody>
</table>
Our Vision (Simplified)

<table>
<thead>
<tr>
<th>Internet</th>
<th>GNUUnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>Applications</td>
</tr>
<tr>
<td>DNS/X.509</td>
<td>GNU Name System</td>
</tr>
<tr>
<td>TCP/UDP</td>
<td>CADET (Axolotl+SCTP)</td>
</tr>
<tr>
<td>IP/BGP</td>
<td>$R^5N$ DHT</td>
</tr>
<tr>
<td>Ethernet</td>
<td>CORE (OTR)</td>
</tr>
<tr>
<td>Phys. Layer</td>
<td>HTTPS/TCP/WLAN/...</td>
</tr>
</tbody>
</table>
Today: 45 things to do with GNUnet

- A fast tour-de-force through GNUnet’s features
- Features for users, developers and researchers
- What you can do, **not** how it is done
C library

- Safer C: GNUNET_malloc(), GNUNET_asprintf(), ...
- Containers: multi hash map, Bloom filter, heap, ...
- Networking: event loop, socket abstraction (client, server)
- Initialization: find paths, parse configuration, parse options
- Disk: buffered and unbuffered IO, endianess conversion, logging
Cryptographic primitives

- RNG, permutation
- AES, Twofish
- SHA-512, SHA-256, SCRYPT, HKDF, CRC32, CRC16
- Curve25519 point addition, Curve25519 point multiplication, small-scalar Curve25519-DLOG
- EdDSA, ECDHE
- Paillier (homomorphic addition)
- RSA blind signatures
The Automated Restart Manager (ARM)

- Starts services on-demand (like systemd)
- Automatically restarts crashed services (like ARM on OS/360)
- Can provide performance data per service
- `gnunet-arm -e` only terminates after peer is fully down
- Simple API: `GNUNET_ARM_request_service_start()`, `GNUNET_ARM_request_service_stop()`, etc.
Transport

- Unreliable, out-of-order packet delivery semantics
- Over TCP, UDP, IPv4/IPv6, HTTP/HTTPS, WLAN or BT (pluggable)
- Enforces bandwidth quotas
- Enforces connection restrictions (F2F)
- Supports NAT traversal
- Supports bootstrap via broadcast/multicast
- Measures network latency
- UDP/WLAN/BT: Fragments large messages (including ACKs and selective retransmission)
Distance-Vector Routing (WiP)

- Transport plugin
- Bounded (i.e. \( \leq 3 \) hops) distance-vector routing
- Provides “illusion” of direct connections
Automated Transport Selection (ATS)

- Decides which connections to establish
- Selects “best” transport plugin to use
- Allocates bandwidth to peers by network technology (LO, LAN, WAN, WLAN)
- Allows other subsystems to specify preferences:
  - Which peers?
  - Minimize latency?
  - Maximize bandwidth?
CORE

- Off-the-record link encryption between peers
- Multiplexes inbound messages by type to higher-level subsystems
- Hides connections from/to peers that do not speak same higher-level protocol
HOSTLIST

- Allows download of known peer addresses for bootstrapping
- HTTP client and HTTP server provided
- URLs from configuration or learned via gossip among peers
The Network Size Estimate (NSE)

- Gives estimate of $\log n$ where $n$ is number of active peers (with reasonable lifetime)
- All peers converge to the same network size estimate
- Extremely cheap (bandwidth, storage & amortized CPU cost)
- Byzantine fault-tolerant
- Malicious attacker can only slightly increase size estimate
- Trivial API: GNUNET_NSE_connect()
Distributed Hash Table (DHT)

- Store key-value pairs in overlay network
- Replication in the network
- Multiple values per key possible
- Duplicate/known replies not transmitted repeatedly
- Tolerates small-world underlay topology
- Can optionally track path key-values took in the network
- $O(\sqrt{n} \log n)$ lookup complexity, $O(\log n)$ hops
- Plugins provide custom logic to verify integrity of key-value pairs in DHT
- Simple API: `GNUNET_DHT_get()`, `GNUNET_DHT_put()`, `GNUNET_DHT_monitor_start()`
Confidential Ad-Hoc Decentralised End-to-End Transport

- AXOLOTL-encrypted end-to-end communication
- Reliable or unreliable
- In-order or out-of-order
- Low-latency or buffered
- Multiple streams duplexed over one authenticated encrypted channel
- Encrypted channel multiplexed over multiple, redundant paths
- Easy API: GNUNET_CADET_connect(), GNUNET_CADET_channel_create(), GNUNET_CADET_notify_transmit_ready()
Identity (management)

- Public key pairs as “egos” to identify users
- Each user can have many alter-egos (or pseudonyms)
- Separate from peer identities (network addresses)
The GNU Name System (GNS)

- Decentralized name system with secure memorable names
- Delegation used to achieve transitivity
- Also supports globally unique, secure identifiers
- Achieves query and response privacy
- Provides alternative public key infrastructure
- Interoperable with DNS
- Trivial API: GNUET_GNS_connect(), GNUET_GNS_lookup()
(Key) revocation

- Instant revocation at all peers that the network allowed to receive it
- Highly efficient protocol
- Revocation messages can be prepared and stored off-line if desired
- Trivial API: `GNUNET_REVOCATION_revoke()`, `GNUNET_REVOCATION_query()`
Set

- Compute set union or set intersection
- Surprisingly low bandwidth required
- Few round trips, but non-deterministic
Scalarproduct (SMC)

- Given private maps \( a : A \rightarrow \mathbb{Z} \) and \( b : B \rightarrow \mathbb{Z} \), calculates scalar product

\[
\prod_{e \in A \cap B} a(e) b(e)
\]  

(1)

- Bandwidth-efficient at \( \approx 100 \) bytes/element
- CPU-efficient with runtime in milliseconds/element
- Only leaks information derivable from final result and prior knowledge
- Result only disclosed to one party
- Assumes honest-but-curious adversary model
- Trivial API: GNUNET_SCALARPRODUCT_start_computation(), GNUNET_SCALARPRODUCT_accept_computation()
Random Peer Sampling (WiP)

- Selects a random peer, or sequence of random peers
- Fully decentralised
- Byzantine fault-tolerant
Multicast (WiP)

- Source controls membership in multicast group
- End-to-end encrypted
- Source does not have to KX with each group member
- Members that left really can no longer read messages
Extensible messaging format: syntax and semantics
Stateful protocol with state updates using deltas
Efficient encoding and decoding (in bandwidth and CPU)
Runs on top of Multicast
Social (Network Applications)

- Combines PSYC2 and GNS to build social networking applications
- Key concepts:
  - **nym**: pseudonym of another user in the network
  - **place**: where social interactions happen
  - **host**: owner of a place
  - **guest**: visitor of a place
- API then offers vocabulary: enter, leave, host eject, host entry decision, host announce, guest talk, place history replay, place look at
SecuShare (WiP)

- Social networking application using SOCIAL API
- GUI written with Qt
Statistics

- Collects numeric run-time information from a peer
- Used for primarily for diagnostic monitoring and performance evaluation
- Trivial API: GNUNET_STATISTICS_set(), GNUNET_STATISTICS_update(), GNUNET_STATISTICS_get()
The Testbed

- Run controlled experiments
- Detect available ports, generate configurations
- Share services across peers for higher efficiency (i.e. DNS resolver)
- Connect peers into custom network topologies
- Run peers with non-uniform configurations
- Run multiple peers on one host
- Run testbed across multiple hosts
- Control large-scale execution with hierarchy of testbed controllers
- Launch thousands of peers per second
Conversation

- GNU Name System PKI:
  - Address book ≡ GNS zone
  - make calls to phone.alice.bob.gnu
- OPUS-encoded voice streams
- CADET end-to-end encryption
- Clean API, command-line and GTK user interfaces
- put calls on hold, etc.
- still lacks ringtones!
File-"Sharing"

- Anonymous, pseudonymous and non-anonymous *publishing*
- Files broken up into blocks (Merkle tree)
- Peers caching blocks cannot view contents (encrypted queries and replies)
- Multi-source download
- Contributing peers rewarded with better performance
- Keyword search
- File meta-data available as part of search result
- Can share directories, can mount shared directories via FUSE
- API, command-line and GTK GUIs
Search by REgular EXpression

- Service publisher advertises regular expression (!)
- Client searches using string
- Services where the RegEx matches string are returned
- Fully decentralised, uses $R^5N$ DHT
- Trivial API: GNUNET_REGEX_announce(), GNUNET_REGEX_search()
- Warning: non-trivial theory. Read up about RegEx prefixes before using.
DNS Integration

Intercept DNS queries using iptables to:

▷ Observe DNS activity
▷ Drop DNS queries
▷ Supply “alternative” DNS replies
▷ Can be used to support GNS instead of NSS, proxies or GNS-specific resolution APIs
IP-over-GNUnet

- Open TUN interface to receive inbound IP traffic ("VPN")
- Open TUN interface to forward IP traffic to Internet ("EXIT")
- Translate between IPv4 and IPv6 as needed and implement NAT-PT for DNS ("PT")
- Also allows routing IP traffic to a particular GNUnet peer
- Integrates with the GNU Name System
Byzantine Fault-tolerant Consensus

- Given a set of $n$ peers with at most $k$ malicious participants
- And a deadline (synchronous protocol!) and enough bandwidth for honest participants
- Compute the global *union* over a set of initial elements distributed across the $n - k$ honest participants
- Malicious participants may add additional (well-formed) elements
- All honest participants end up with exactly the same set
- Final set is super-set of union of initial elements at honest peers
Electronic Voting (SMC)$^1$

- Implements Cramer’97-style electronic voting:
  - **correctness**: votes are counted correctly, one vote per voter
  - **secrecy**: voter’s votes remain secret
  - **indi. verif.**: each voter can verify
  - **univ. verifi.**: third parties can verify
  - **fairness**: will not leak partial outcomes
  - **robustness**: a threshold faction of officials may be corrupt
  - **coercion-res.**: *Not* offered! Adversary could verify that voter complied with his demands

- Three types of participants:
  - **supervisor**: affirms list of eligible voters, selects authorities
  - **authorities**: collect & verify ballots, tally results, provide audit data
  - **voter**: registers to vote, votes, submits ballot

---

$^1$Implemented in gnuet-java
RESTful APIs (WiP)

- Access GNUnet services via HTTP
- Plugin architecture
- Data encoded using JSON
- Used to build Web service with JS for identity management
Taxable Anonymous Libre Electronic Reserves

- Payment system, not a new currency
- Client-server architecture, not peer-to-peer
- HTTP RESTful protocol (JSON over HTTP/HTTPS)
- Supposed to be used initially over Tor for anonymity
- Payer remains anonymous
- Payee easily identifiable by the government (“taxable”)
- Affero GPL server, GPL wallet, LGPL merchant logic
- Cheap transactions, can give change, supports refunds
GNUnet dependencies (generated by GNU Guix)

Compile time:

Runtime:

Close inspection shows: Guix didn’t build all of it.
Future Work

- Improve all of the above, in particular the WiPs
- Onion routing
- Asynchronous messaging
- Secure auctions
- News distribution / timeline construction
- Collaborative editing
- Multiparty linear programming
Conclusion

- GNUnet provides foundations for an alternative network stack
- More work needs to be done: SMTP 2.0, Web 3.0, Tor 2.0, ...
- If what you need is not there, help us add it!
Do you have any questions?

References:


