HS18

DISTRIBUTED SYSTEMS ADVANCED & BLOCKCHAIN

Protocols

T. Bocek

thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Distributed Systems - In the *News*

- **01.11.2018** **Tether Confirms That It Is Banking With Bahamas-Based Deltec**
  - Scanned PDF: “without any liability, however arising, on the part of Deltec Bank & Trust Limited, its officers, directors, employees and shareholders”
- **05.11.2018** **Deltec Chairman Says Tether Letter on Bank Relationship Is 'Authentic'**
- **02.11.2018** **Proof of Stake or Proof of Work, What's the Difference?**
  - Ethereum: total mining costs: $2,277,959,012, annual rewards: $1,378,876,829
  - Bitcoin: total mining costs: $3,656,073,069, annual rewards: $4,769,978,010
  - “Overall PoS seems to be a better solution as it will make the blockchain safer, drastically reduce its power consumption, and reduce the time it takes to make transactions.”

DELTEC
DELTEC BANK & TRUST LIMITED

November 1, 2018

Tether Limited
17F-1, No. 266, Sec.1
Wenhua Rd., Banqia Dist.
New Taipei City 22041
Taiwan
Republic of China

Dear Sirs:

We hereby confirm that, at the close of business on October 31, 2018, the portfolio cash value of your account with our bank was US$1,831,322,828.

This letter is provided without any liability, however arising, on the part of Deltec Bank & Trust Limited, its officers, directors, employees and shareholders, and is solely based on the information currently in our possession.

Yours faithfully,

Deltec Bank & Trust Limited

Deltec House, Lyford Cay
P.O. Box N-3229
Nassau, Bahamas
www.deltecbank.com
Tel: +1 (242) 302-4100
Fax: +1 (242) 362-4623

https://tether.to/wp-content/uploads/2018/11/Tether-Letter.pdf

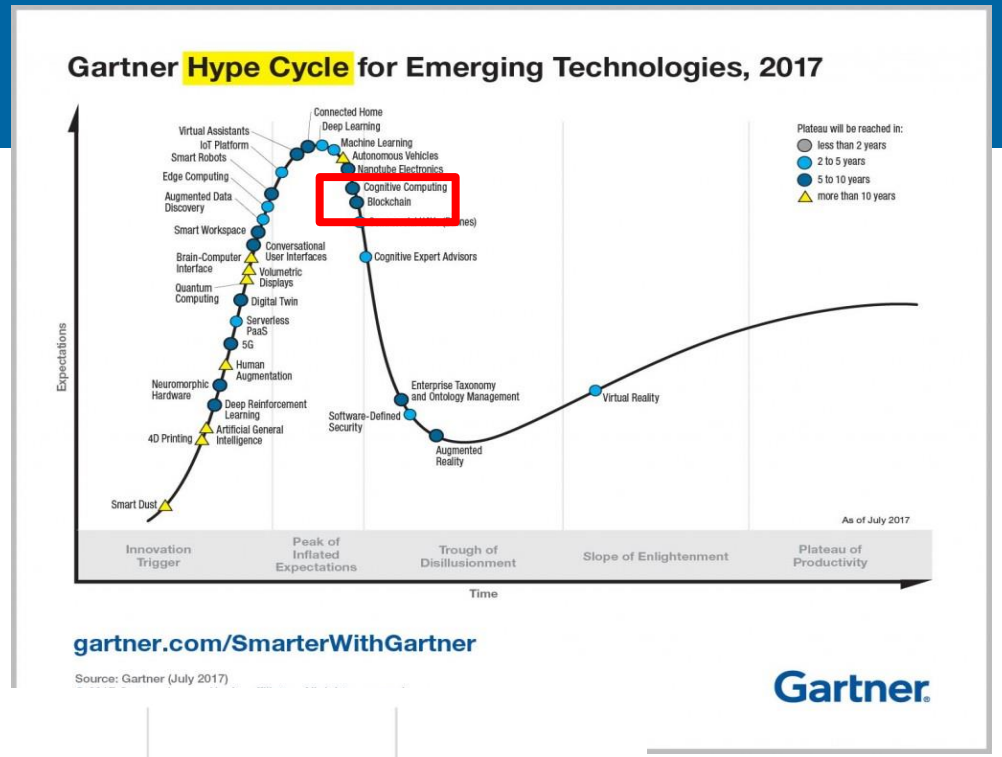
Distributed Systems - In the *News*

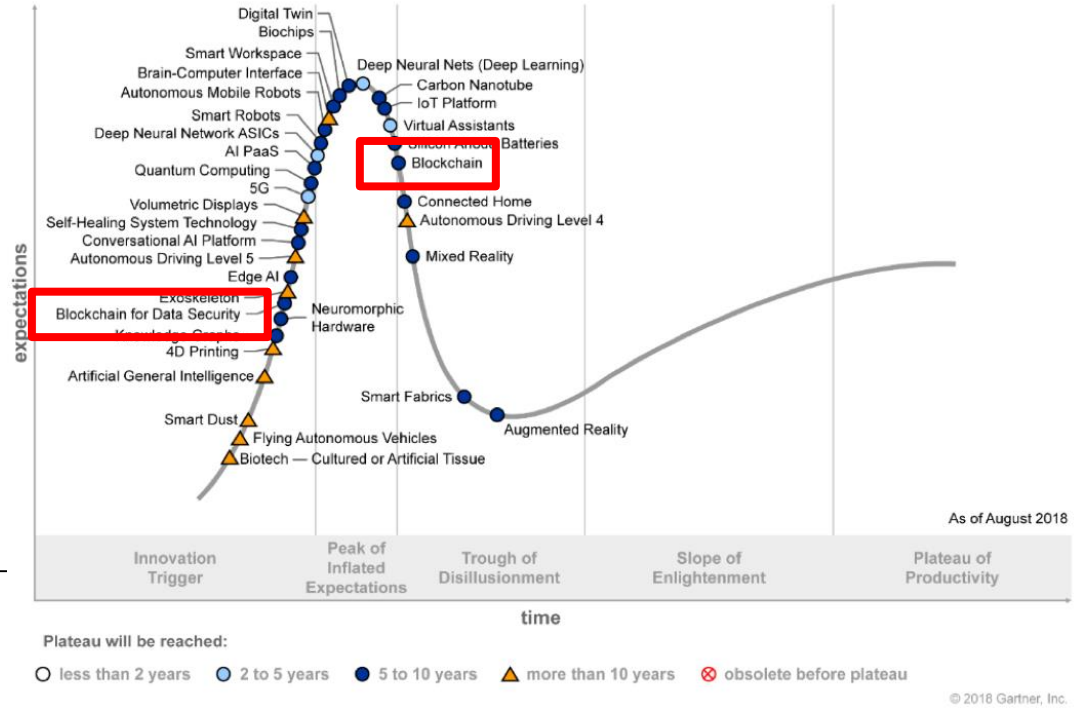
- **02.11.2018 Swedish ISP Protests 'Site Blocking' by Blocking Rightsholders Website Too**
  - Elsevier requested Sci-Hub to be blocked, ISP Bahnhof now also blocks Elsevier
- **05.11.2018 How does gas work? (Infographic)**
- **NEO Hackathon results:**
  1. NEOKey – Integrate NEO into AirGap (offline wallet)
  2. BlockSelfie – Prove your identify with a selfie together with a random person

Distributed Systems - In the *News*

- **Gartner Hype Cycle – 2016, 2017, 2018**

Source: Gartner (July 2016)



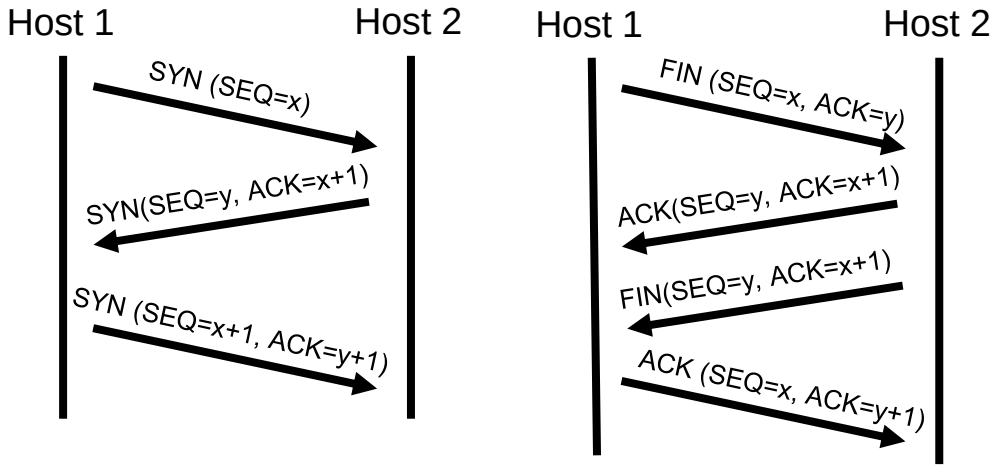


Challenge Task

- **Web3j**
  - compile 'org.web3j:core:3.4.0'
  - `Web3j web3 = Web3j.build(new HttpService("https://morden.infura.io/your-token"));`
  - Smart contract, compile with solc:
    - `solc <contract>.sol --bin --abi --optimize -o <output-dir>/`
  - Generate Java Wrappers
    - `web3j solidity generate /path/to/<smart-contract>.bin /path/to/<smart-contract>.abi -o /path/to/src/main/java -p com.your.organisation.name`
    - `YourSmartContract contract = YourSmartContract.finishMinting(…`
- **Testonator**
  - I don’t like generating a Java wrapper
  - compile 'io.iconator:testonator:1.0.28’
  - `File f = Paths.get(ClassLoader.getSystemResource("DOS.sol").toURI()).toFile();`
  - `Map<String, Contract> contracts = compile(f);`
  - `DeployedContract deployed = blockchain.deploy(CREDENTIAL_0, contracts.get("YourSmartContract"));`
  - `List events = blockchain.call(deployed, new FunctionBuilder("finishMinting"));`
- **Today deadline milestone**
  06.11.2018 - 23:59

Protocol, Serialization, VSS recap: Layer 4 - TCP

- **Connection establishment**
  - SYN, SYN-ACK, ACK (three way)
  - Initiates TCP session: initial sequence number is ~random
  - Congestion control
- **Connection termination**
  - FIN, ACK + FIN, ACK (three/four way)
  - 3-way handshake, when host 1 sends a FIN and host 2 replies with a FIN & ACK
- **Sequences and ACKs**
  - Identification each byte of data
  - Order of the bytes → reconstruction
  - Detecting lost data: RTO, DupACK:



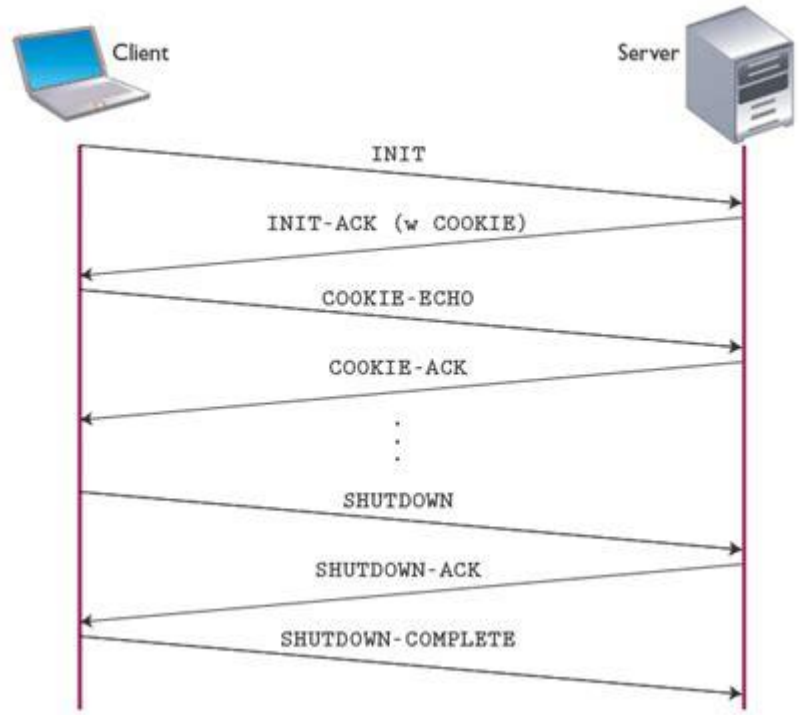
- **Retransmission timeout**
  - If no ACK is received aftert timout (e.g. 2xRTT), resend.
- **Duplicate cumulative acknowledgements**
  - ACKs for last consecutive packets
  - 3 times same ACK → retransmit (fast retransmit)

Protocol, Serialization, VSS recap: Layer 4 - Transport

- **User Datagram Protocol (UDP)**
  - UDP is used for DNS, streaming audio and video
  - Simple connectionless communication model
  - No guarantee
    - Delivery
    - Ordering
    - Duplicate protection
- **Simple Client/Server (wireshark)**

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                data                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

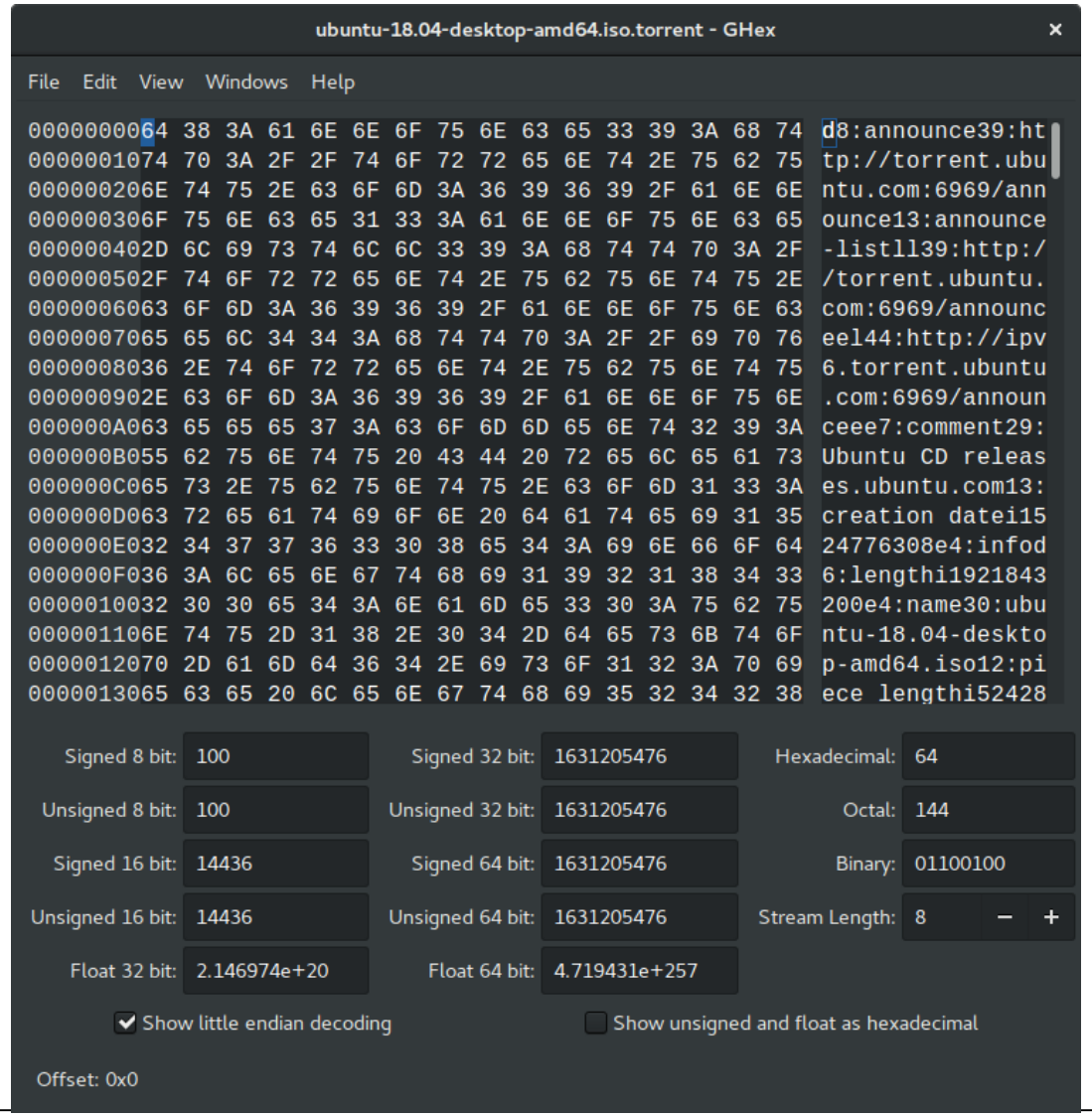
- **SCTP (Stream Control Transmission Protocol)**
  - Message-based
  - Allows data to be divided into multiple streams
  - Syn cookies - SCTP uses a four-way handshake with a signed cookie.



https://doi.ieeecomputersociety.org/cms/Computer.org/dl/mags/ic/2009/05/figures/mic20090500811.gif

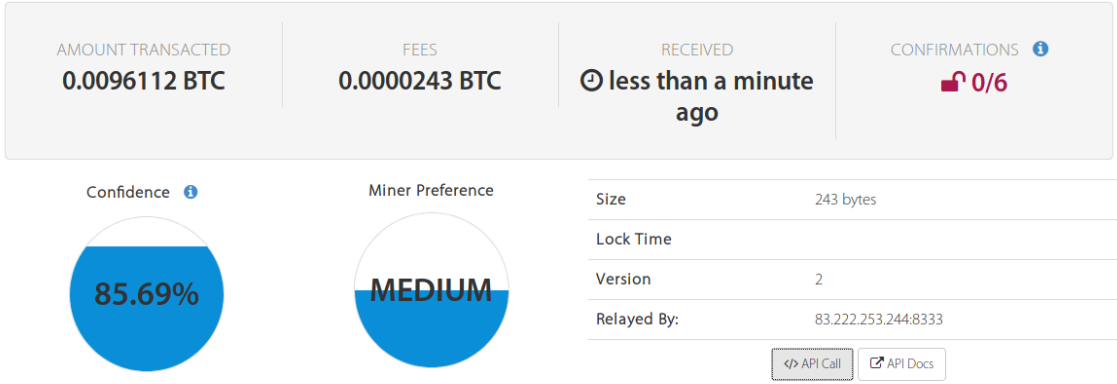
Protocol, Serialization, VSS recap: Protocols

- **Avro / gRPC / ASN1**
- **JSON encoding, e.g., JSON-RPC**
- **Benconding**
  - Integers: i42e
  - Byte string: 4:test
  - List: l4:testi42ee
  - Map/dictionary: d4:test3:hsr3:tomi42ee



The smaller the better

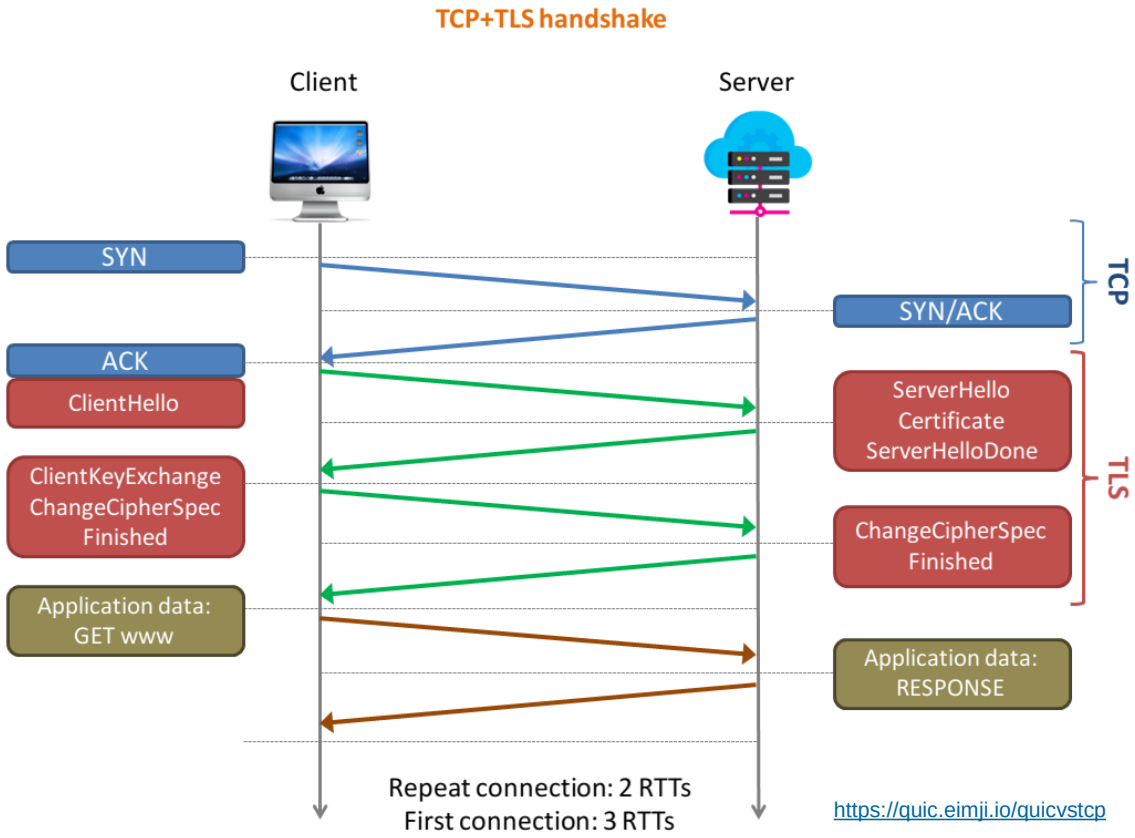
- **Bitcoin TX, random: 243 bytes**
  - In JSON: 2081 bytes
- **Largest safe UDP packet**
  - IPv4: 508 bytes payload
  - IPv6: 1212 bytes payload
- **~Best size: <1500 bytes?**
  - TomP2P uses 1400, MTU discovery, adds complexity
- **How reliable is UDP?**



	Receiver				
	NJ 1	**NJ 2**	**LA**	**NLD**	**JPN**
NJ 1	-	100	99.97	100	99.97
NJ 2	100	-	99.97	99.97	100
LA	98.64	98.55	-	99.86	100
NLD	100	99.97	100	-	100
JPN	99.96	100	100	100	-

Protocol, Serialization

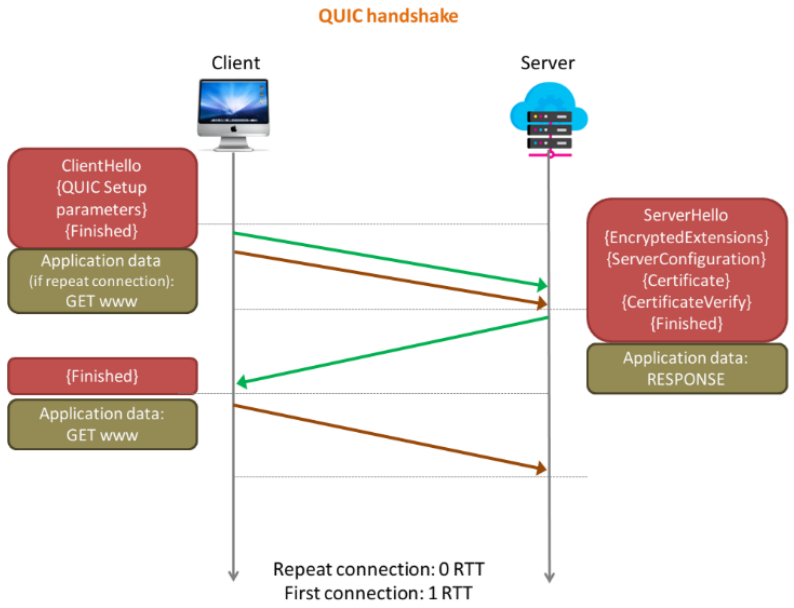
- **Security: Transport Layer Security (TLS)**
  1. "client hello" lists cryptographic information, TLS version, ciphers/keys
     - Forward secrecy: ephemeral keys
  2. "server hello" chosen cipher, the session ID, random bytes, digital certificate (checked by client), optional: "client certificate request"
  3. Key exchange using random bytes, now server and client can calc secret key
  4. "finished" message, encrypted with the secret key

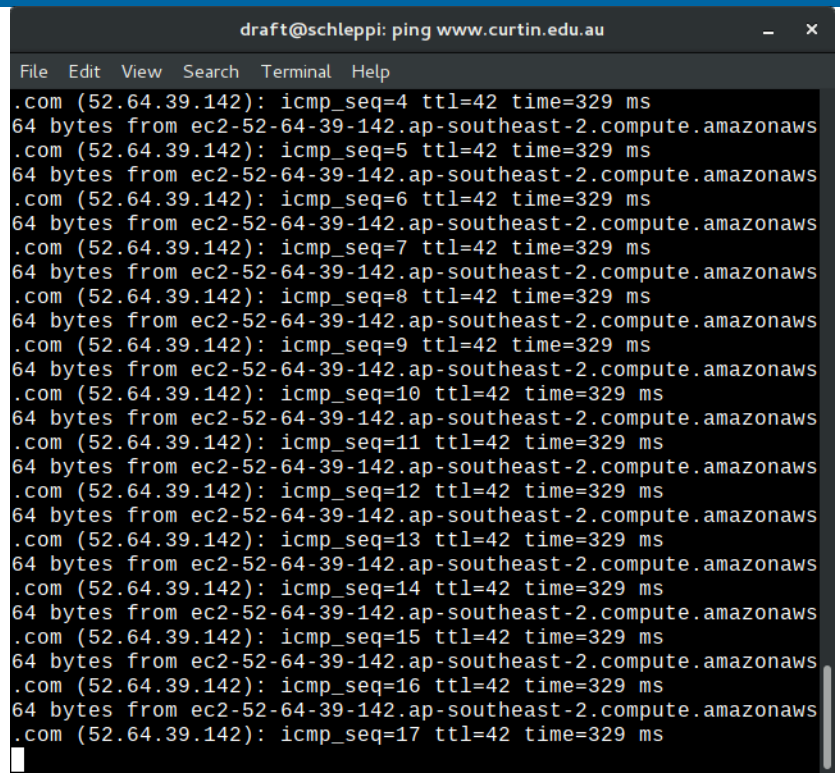


https://www.ibm.com/support/knowledgecenter/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660_.htm

Protocol considerations

- **Ping to Australia: 329ms**
  - One way ~ 165ms
- **TCP + TLS handshake:**
  - 3RTT = 987ms! No data sent yet
- **Last lecture - QUIC: 1RTT**





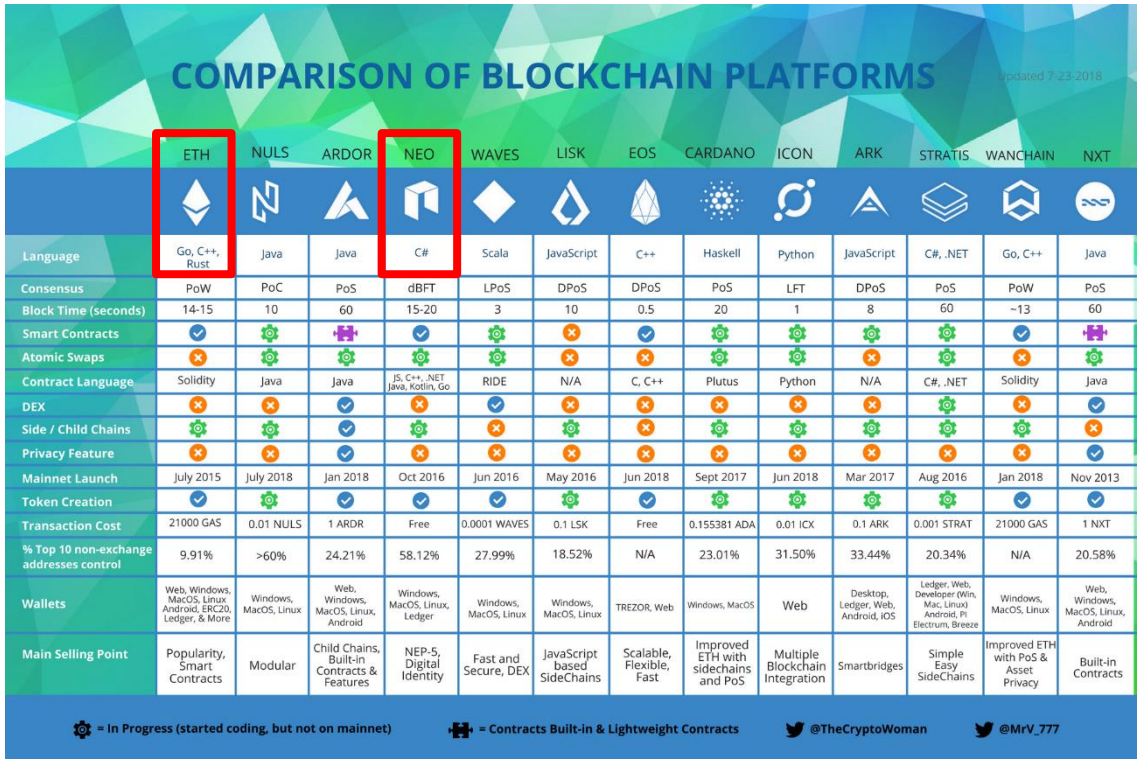
URL: b.socrative.com/student/

Room: HSR

QUIC

- **Experimental transport layer protocol**
  - Implemented 2012, announced 2013
- **Reduce RTT: 1 RTT**
  - 1 second connection establishment annoying with web browsing
  - CDNs: ping www.australia.gov.au
- **Main goal: “improve perceived performance of connection-oriented web applications”**
- **IETF standardization, multipath and FEC**
- **2018: ~1.2% of webservers support QUIC**
- **Client stores a synchronization cookie: 0 RTT**
- **Chrome since 29, Opera 16**

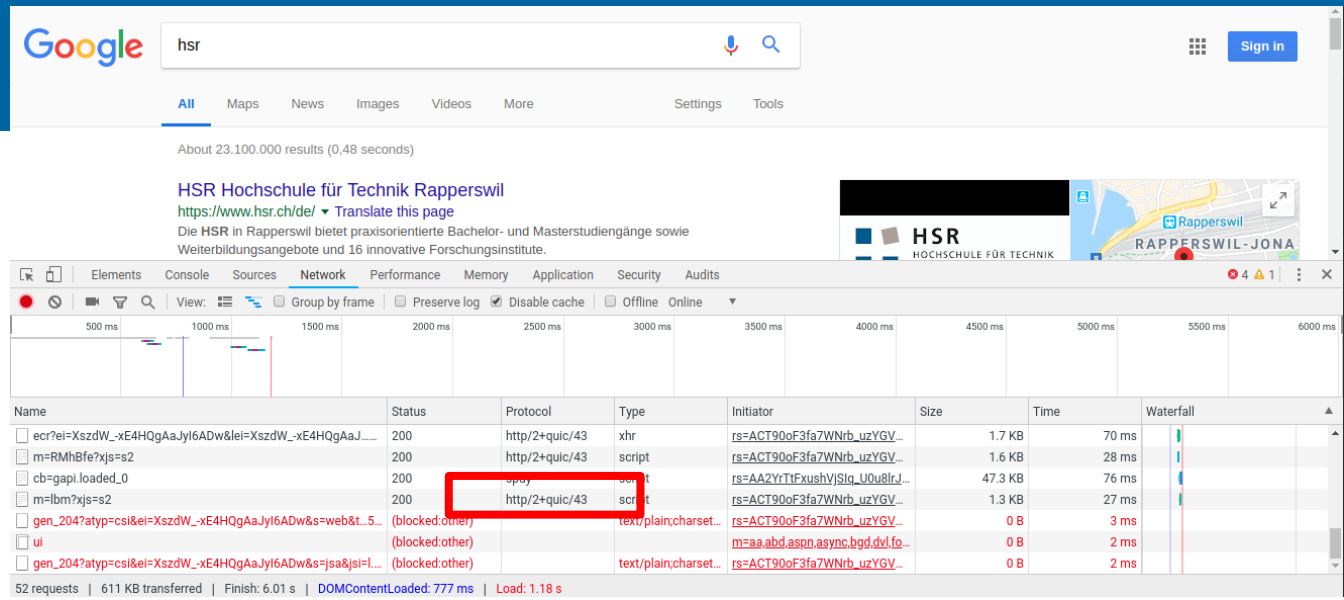
- **Block time Bitcoin 10min**
- **Blockchains today 60s - 0.5s**

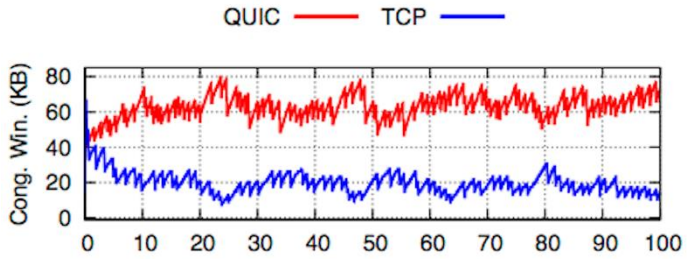


https://twitter.com/thecryptowoman/status/1021385599088054273

QUIC

- **Chrome desktop, not enabled by default, mobile, enabled by default (check if enabled)**
- **Mobile operators not happy: "And because it is encrypted, MNOs can't see the traffic that is flowing on their networks. In a nutshell, mobile networks are "going dark"."**
- **QUIC more aggressive than TCP**
  - Larger congestion window
  - Is it fair?
- **QUIC, issue around Dec 2015, enabled on mobile on Sep. 2016.**
  - Usage google: 34% in 2016





https://blog.apnic.net/2018/01/29/measuring-quic-vs-tcp-mobile-desktop/

SIGCOMM '17, August 21-25, 2017, Los Angeles, CA, USA

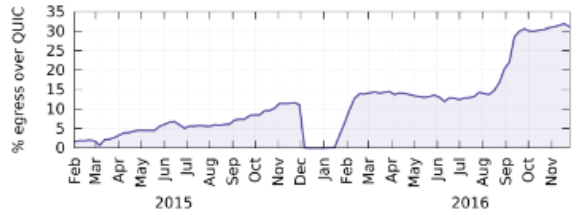
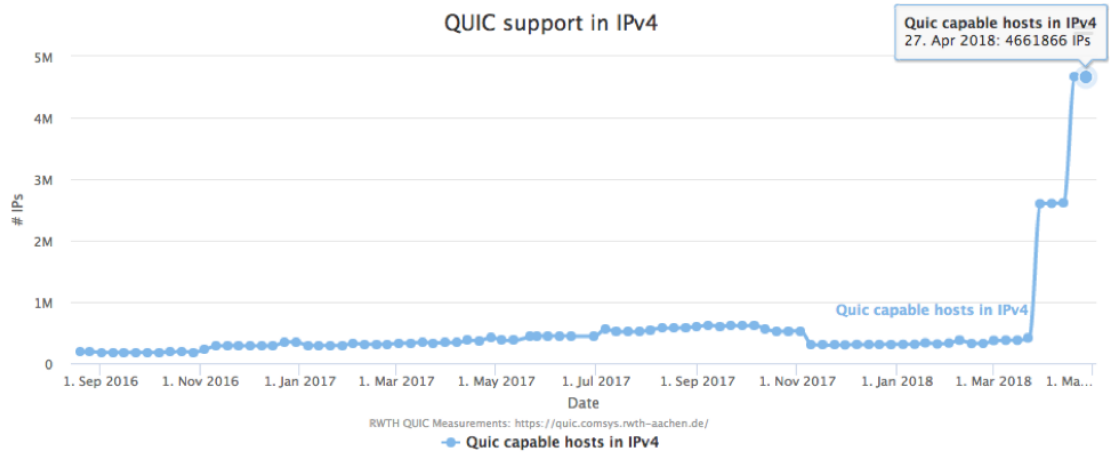


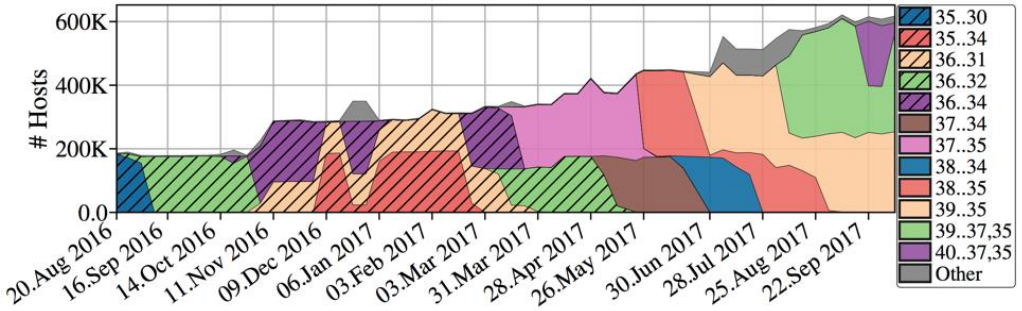
Figure 2: Timeline showing the percentage of Google traffic served over QUIC. Significant increases and decreases are described in Section 5.1.

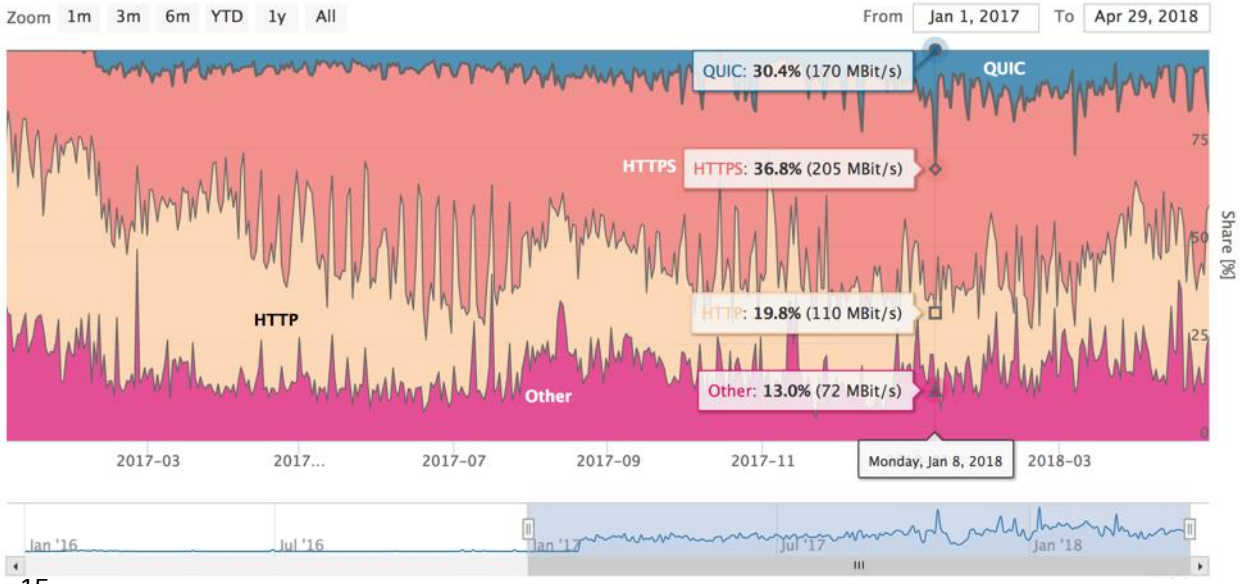
https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46403.pdf

QUIC

- **QUIC in the wild**
  - Akamai support in May 2018
  - University uplink: max 30.4% of traffic!
- **UDP on port 443**
- **IETF standard not finalized, excepted end of 2018**
  - Different versions in the wild







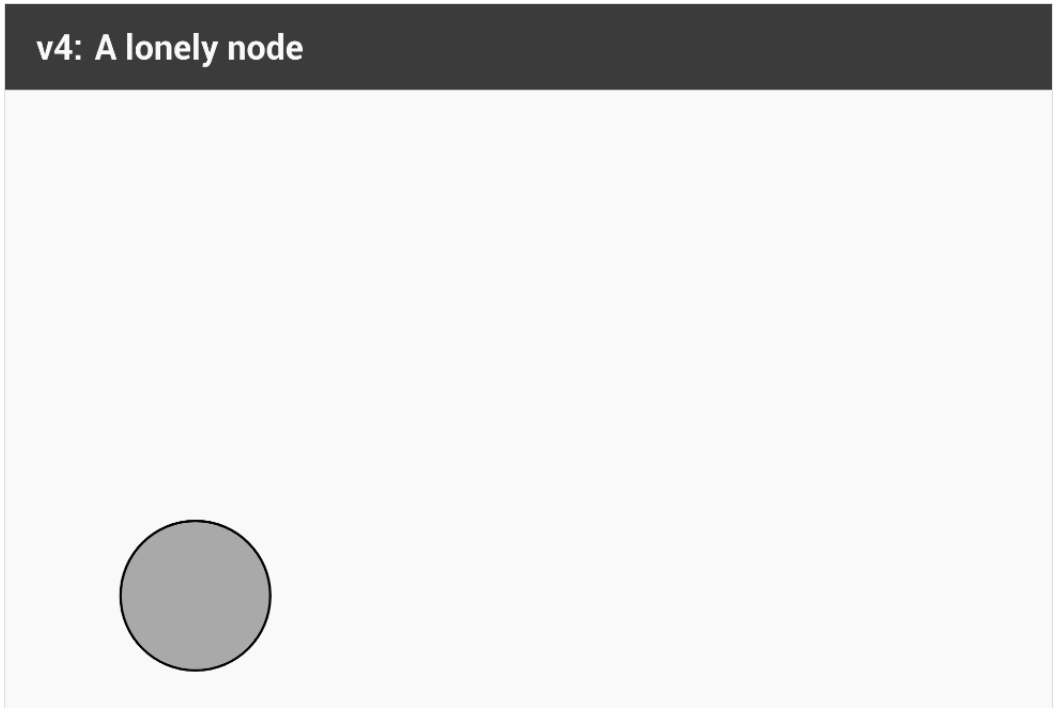
URL: b.socrative.com/student/

Room: HSR

Ethereum Wire Protocol

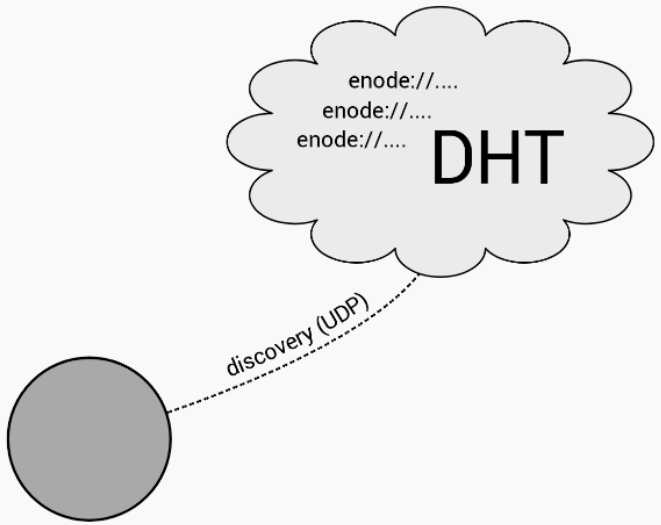
- **Peer-to-peer communications between Ethereum clients:**
  - Node Discovery
    - Ethereum uses a Kademlia-like protocol for node discovery
    - 256-bit cryptographic hashes of the nodes' public keys
  - Transport (RLPx)
  - Devp2p (subprotocols: eth, les, shh, bzz)
    - 0x00 – 0x10 for devp2p, above for subprotocols
    - Hello, Disconnect, Ping, Pong
- **ETH**
  - Status, NewBlockHashes, Transactions, GetBlockHashes, BlockHashes, GetBlocks, Blocks, NewBlock
  - Extensions: PV61, PV62, PV63, PoC-X

- **Ethereum connection: (discovery v4)**

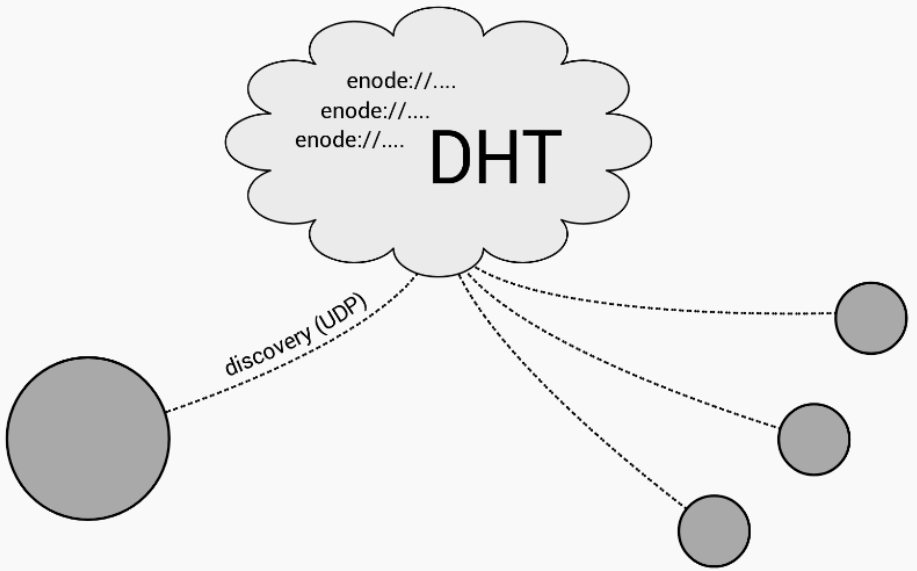


Ethereum Wire Protocol

v4: Connecting to the DHT

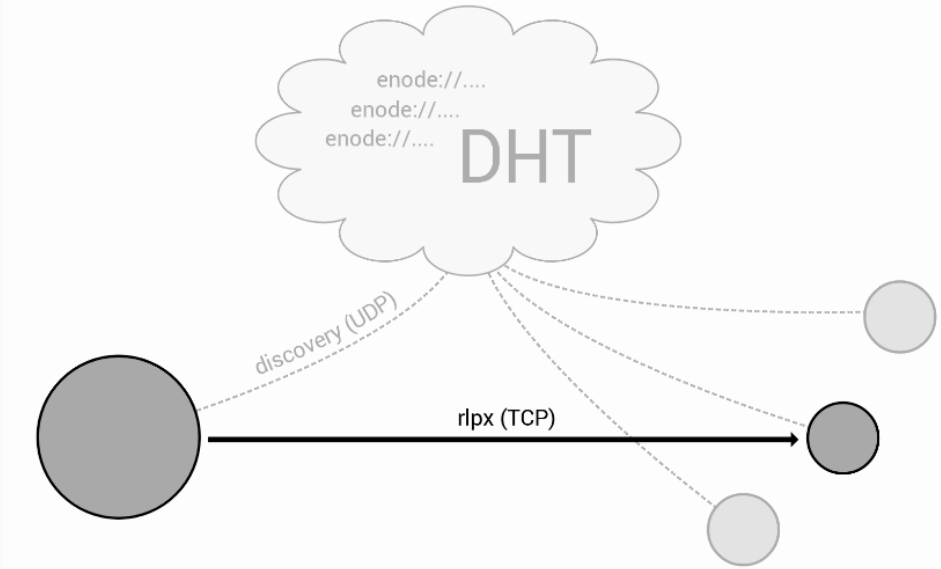


v4: Finding other nodes

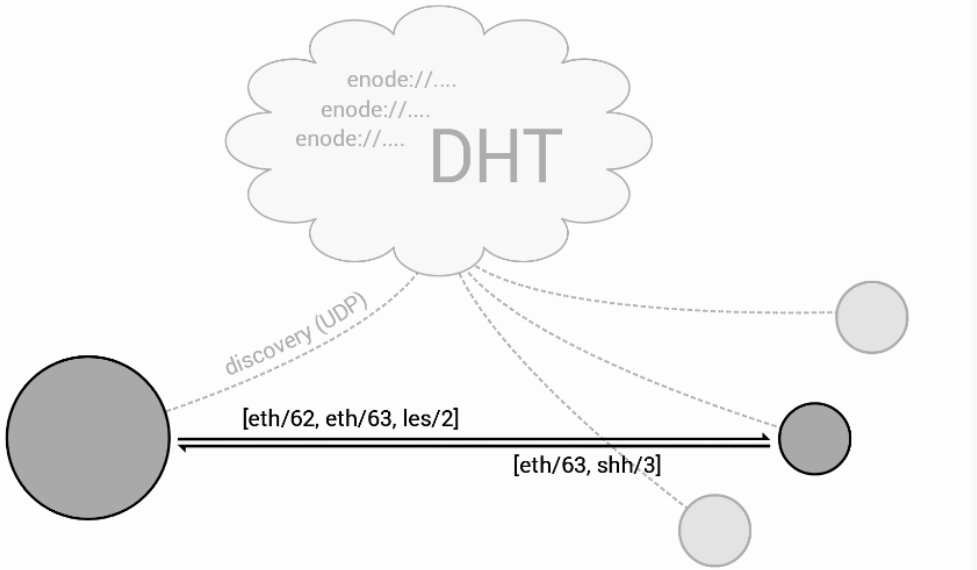


Ethereum Wire Protocol

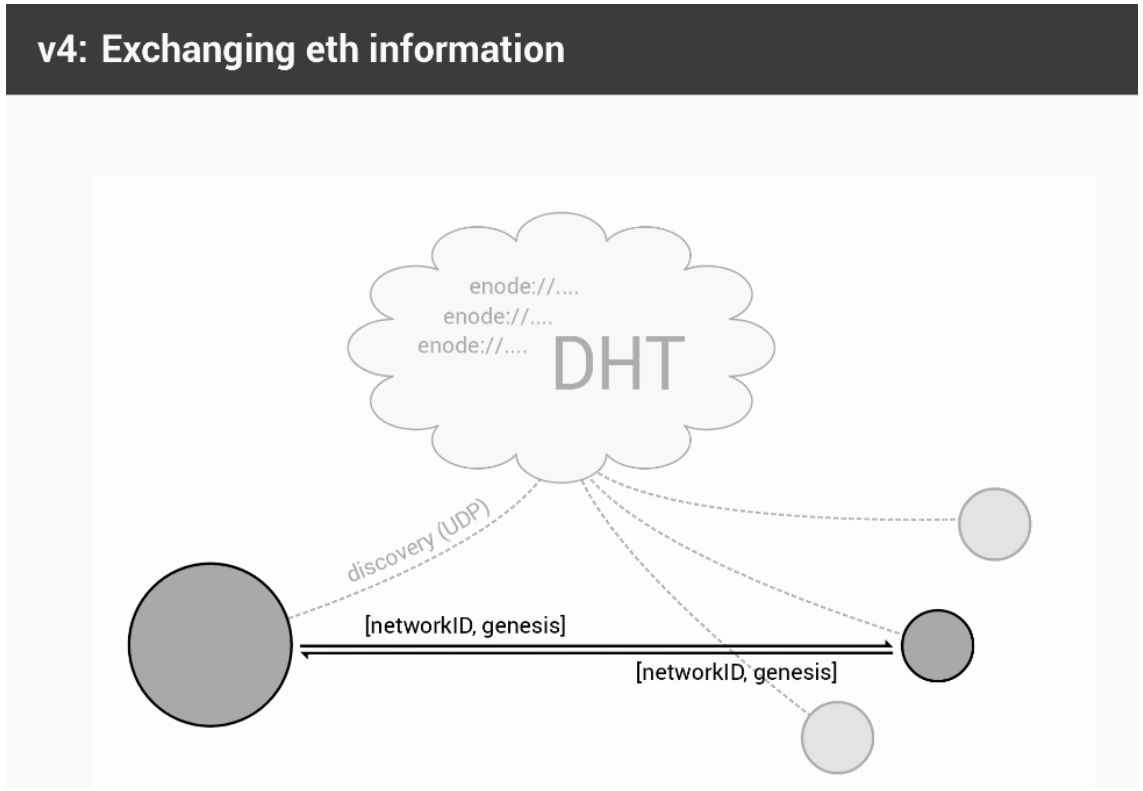
v4: Connecting via TCP



v4: Negotiating devp2p capabilities



Ethereum Wire Protocol



- **Lots of RTT**
  - Ethereum Node Records (ENR) enhancement
    - Node discovery only used to find nodes, IP+port
    - Improvement: flexible format, contain e.g., public keys
    - At most 300 bytes
    - “We can try ifps/libp2p transports”
  - DHT spam prevention
- **SHH: Whisper for multi-casting and broadcasting with topics using DHT information**
  - POC 2 stage (alpha)

The RLPx Transport Protocol

- **RLPx is a cryptographic peer-to-peer network and protocol suite which provides a general-purpose transport and interface for applications to communicate via a p2p network.**
  - All cryptographic operations are based on the secp256k1 elliptic curve (same as Bitcoin)
- **ECIES Encryption**
  - Shared key (derivation), KDF
  - Authentication, MAC
  - Symmetric encryption, AES
- **Messages: auth, auth-ack – + 1RTT (TLS + 2RTT)**
- **Framing**
  - multiplexing multiple protocols
  - frame:
    - frame = frame-size || header-data || header-mac || frame-data || frame-mac

Cryptographic primitive	Hash	MAC	Digital signature
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Kind of keys	none	symmetric	asymmetric

source

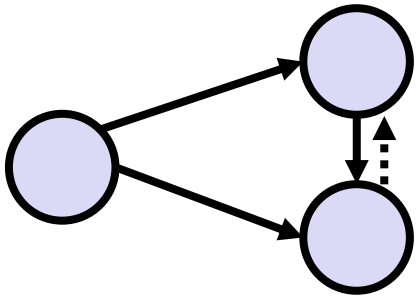
URL: b.socrative.com/student/

Room: HSR

Kademlia Broadcasting / Flooding in P2P

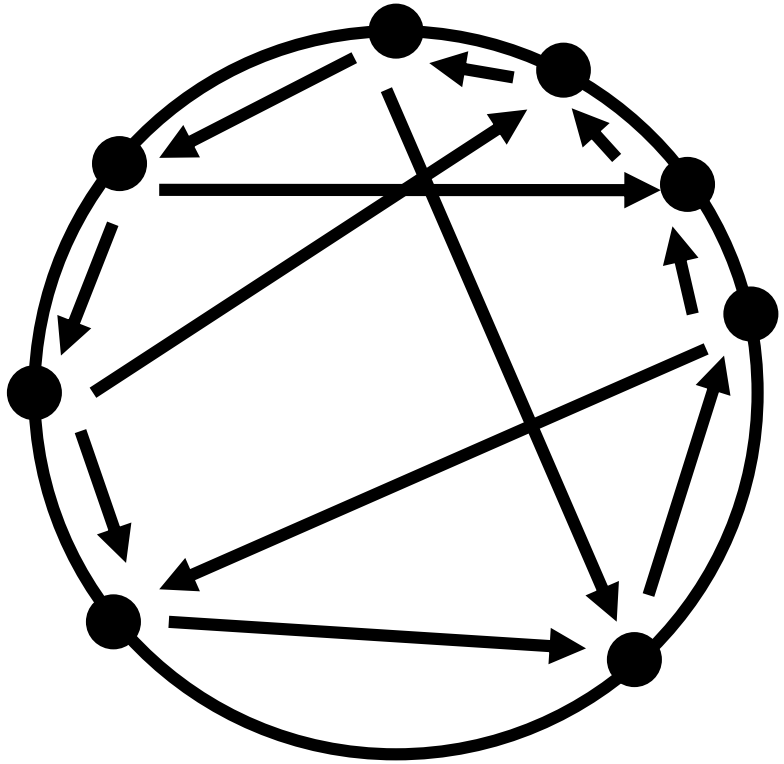
- **Flooding scales poorly**
  - Unnecessary traffic
  - 5 peers, fully meshed: 4 msg, 4 x 3 msg → 16 msg
- **Reduce Messages**
  - Don’t send to all neighbors (random walk)
  - Coupon collector's problem
    - Number of trials grow O(n log n),
    - Reach 50 peers, it takes about 225 trials to reach all 50 peers
    - Tradeoff: all peers vs. most peers / many msgs vs. reduce msgs
- **Broadcasting use-cases**
  - Broadcast global parameters, global events (maintenance)
  - Hierarchical system: broadcast root inode (e.g. / in a file system)

- **Ethereum Wispher: TTL, “probabilistic message forwarding”**



Broadcasting in Structured Systems

- **5 peers fully meshed: How to send only 4 messages?**
- **Structured vs. unstructured:**
  - Knowing the direction
- **Broadcasting in ring-based Overlay**
  - Main idea: tell other peers in what direction so send
- **Example:**
  - Send to neighbor (successor)
  - For faster spreading / redundancy send to far away peer (overhead)
  - Overhead: 4 messages
- **Kademlia? Tree-based structure**



Broadcasting in Kademlia

- **Tree structure – no direction, example 7 peers**
- **Main idea: send to random peer in bag X**
  - Send along in which bag this peer was, only send to smaller bags

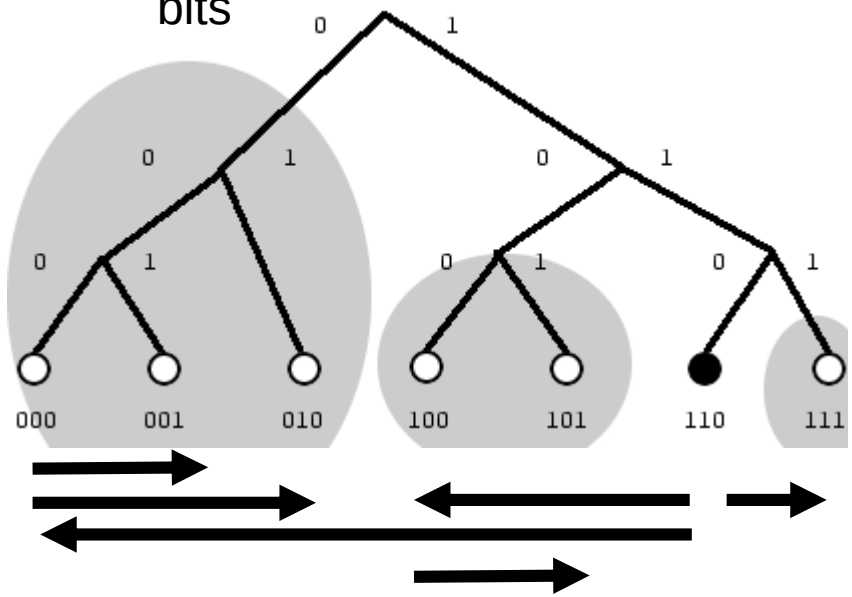
Peer 6 (110) broadcast to b1,b2,b3		
7 (b1)	4 (or 5) (b2)	0 (or 1, 2) (b3)

Peer 4 (100) broadcast to b1		
5 (b1)	-	-

Peer 0 (000) broadcast to b1,b2		
1 (b1)	2 (b2)	-

- **Peer 4 and 0 send simultaneously**

Routing with XOR, with 3-bits



Broadcasting in Kademlia

- **7 peers, 6 messages**
  - Works only with perfect routing, churn?
- **Redundancy: send to R random peer in bag X**
  - Previous example: R=1, TomP2P: R=3
- **Demo: StructuredBroadcastHandler**
  - 1000 peers
  - FROM_EACH_BAG = 3, ~3800 messages
  - No redundancy: 999 messages
- **Testcase/showcase:**
  - net.tomp2p.p2p.TestBroadcast.testBroadcastTCP()

Serialization in Ethereum

- **RLP (Recursive Length Prefix): encode arbitrarily nested arrays of binary data**
  - Integers represented in big endian
  - String 0x8[len]: [0x83, 'd', 'o', 'g']
    - Data is string if in range [0x80, 0xb7]
    - Larger data, 0xb8, next byte is len
    - 0xb9, next 2 bytes len, … 0xbf 8 bytes len
    - At most 64bit length
  - Single byte below 55: 15 = 0x0f
  - List 0xc[len]: [0xc8, 0x83, 'c', 'a', 't', 0x83, 'd', 'o', 'g']
- **Parameter encoding:**

```
contract Foo {
  function bar(bytes3[2] memory) public pure {}
  function baz(uint32 x, bool y) public pure returns (bool r) { r = x > 32 || y; }
  function sam(bytes memory, bool, uint[] memory) public pure {}
}
```

- **Method ID: first 4 bytes of hash of method signature**
  - baz(uint32,bool) → 0xcdcd77c0
- **`web3.eth.abi.encodeFunctionSignature('myMethod(string,uint256)')`**
- **`new FunctionBuilder("myMethod")`**
  `.addInput("string", "hallo")`
  `.addInput("uint256", new BigInteger("10000"))`
  - Uint32, e.g. 45: 0x0000000000000000000000000000000000000000000000000000000000000045
  - Boolean, e.g. true: 0x0000000000000000000000000000000000000000000000000000000000000001
  - Onchain: 0xcdcd77c0000000000000000000000000000000000000000000000000000000000000004500000000000000000000000000000000000000000000000000000000000000001
  - Efficiency?

Serialization in Ethereum

- **Dynamic types**
  - `sam(bytes,bool,uint256[])`
  - Location of the data (offset):
    `0x0000000000000000000000000000000000000000000000000000000000000060`
  - `Boolean:`
    `0x0000000000000000000000000000000000000000000000000000000000000001`
  - `Locatoin of data (offset):`
    `0x00000000000000000000000000000000000000000000000000000000000000a0`
  - `At position 60, length of byte array:`
    `0x0000000000000000000000000000000000000000000000000000000000000004`
  - `“dave” (left padded):`
    `0x6461766500000000000000000000000000000000000000000000000000000000`

- **Number of items in array at position 0xa0**
  - 0x0000000000000000000000000000000000000000000000000000000000000003
  - 3 times data item:
    - 0x0000000000000000000000000000000000000000000000000000000000000001
    - 0x0000000000000000000000000000000000000000000000000000000000000002
    - 0x0000000000000000000000000000000000000000000000000000000000000003

- **In total 292 bytes, 16 bytes non-zero**
  - Efficiency? Example

URL: b.socrative.com/student/

Room: HSR

Noise Protocol

- **The Noise Protocol Framework**
  - Used e.g., by Aeternity blockchain
    - Public blockchain based in Lichtenstein, PoW consensus, PoS for governance, account-based, ASIC resistant (Cuckoo Cycle), built-in oracles, off-chain (state channels), runs Ethereum smart contracts
  - ~ Signal and Whatsapp
- **Handshake**
  - Key exchange protocol, static key pair and/or an ephemeral key pair
- **Not to be confused with https://github.com/perlin-network/noise**
  - Noise: An opinionated, P2P networking stack for decentralized protocols in Go.
  - More specific than Noise Protocol: Ed25519 signatures, KCP, glog, and protobufs
  - “libp2p, which feels oddly modular and far too verbose in all the wrong ways as though it was ripped out of IPFS.”

Noise Protocol

- **Supports various handshake patterns**
  - s static key
  - e ephemeral key
  - ee, se, es, ss – shared key exchange based on s and e – se initiator key, e responder key
- **The first character refers to the initiator's static key:**
  - N = No static key for initiator
  - K = Static key for initiator Known to responder
  - X = Static key for initiator Xmitted ("transmitted") to responder
  - I = Static key for initiator Immediately transmitted to responder, despite reduced or absent identity hiding
- **The second character refers to the responder's static key:**
  - N = No static key for responder
  - K = Static key for responder Known to initiator
  - X = Static key for responder Xmitted ("transmitted") to initiator
- **Patterns:**

```
NN:
  -> e
  <- e, ee

NK:
  <- s
  ...
  -> e, es
  <- e, ee

NX:
  -> e
  <- e, ee, s, es

XN:
  -> e
  <- e, ee
  -> s, se

XK:
  <- s
  ...
  -> e, es
  <- e, ee
  -> s, se

XX:
  -> e
  <- e, ee, s, es
  -> s, se

KN:
  -> s
  ...
  -> e
  <- e, ee, se

KK:
  -> s
  <- s
  ...
  -> e, es, ss
  <- e, ee, se

KX:
  -> s
  ...
  -> e
  <- e, ee, se, s, es

IN:
  -> e, s
  <- e, ee, se

IK:
  <- s
  ...
  -> e, es, s, ss
  <- e, ee, se

IX:
  -> e, s
  <- e, ee, se, s, es
```

TomP2P protocol definition

- **IK:**

  <- s

  ...

  -> e, es, s, ss

  <- e, ee, se

- **Which one to choose?**
  - ss: sender authentication vulnerable to key-compromise impersonation (KCI)
  - es/se: sender authentication resistant to key-compromise impersonation (KCI)
  - ee: this payload has forward secrecy, however, sender not authenticated recipient
  - Use both, ee and se – encrypt twice?
- **TomP2P, peerId is now curve25519 public key – 32 bytes - always 0 RTT**

```
2bit protocol type 00 is UDP, 01 is KCP, 10 and 11 planned to have different KCP

    If its KCP, then then the next

30bits are the session Id, followed by the rest of the KCP header.

    Otherwise:

30bit p2p version **4 bytes** //ignore if its wrong

public key of sender xored with public key of recipient with overlapping 28 bytes **36 bytes**

Ephemeral public key of sender, just for this message. **32 bytes sym key based on key pairs sender receiver **12 bytes** nonce overhead from ChaCha20

    now starts the encrypted part with ChaCha20-- - 32bytes sym key based on key pairs es

peeraddress of sender (without IP or id) **min. 2 bytes**

32bit message id             **4 bytes**

4bit message type            (request, ack, ok)

4bit message options  **1 byte**

8bit message command  **1 byte**

nbit data, can be 0 or up to packet size

64 bytes ED signature **64 bytes**
```

- **It total, the header is minimum of size 156b**

URL: b.socrative.com/student/

Room: HSR

Bitcoin Protocol

- **Efficient wire protocol**
  - (Challenge Task question: bootstrap via DNS seed)
- **Message structure**

Field Size	Description	Data type
4	magic	uint32_t
12	command	char[12]
4	length	uint32_t
4	checksum	uint32_t
?	payload	uchar[]

- **Variable length integer**

Value	Storage length	Format
< 0xFD	1	uint8_t
<= 0xFFFF	3	0xFD followed by the length as uint16_t
<= 0xFFFF FFFF	5	0xFE followed by the length as uint32_t
-	9	0xFF followed by the length as uint64_t

- **Block headers**

Field Size	Description	Data type
4	version	int32_t
32	prev_block	char[32]
32	merkle_root	char[32]
4	timestamp	uint32_t
4	bits	uint32_t
4	nonce	uint32_t
1	txn_count	var_

- **Transaction**

Field Size	Description	Data type
4	version	int32_t
0 or 2	flag	optional uint8_t[2]
1+	tx_in count	var_int
41+	tx_in	tx_in[]
1+	tx_out count	var_int
9+	tx_out	tx_out[]
0+	tx_witnesses	tx_witness[]
4	lock_time	uint32_t

Questions?

Name