FS18

DISTRIBUTED APPLICATIONS II

Tor, Rsync, and others

T. Bocek

thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

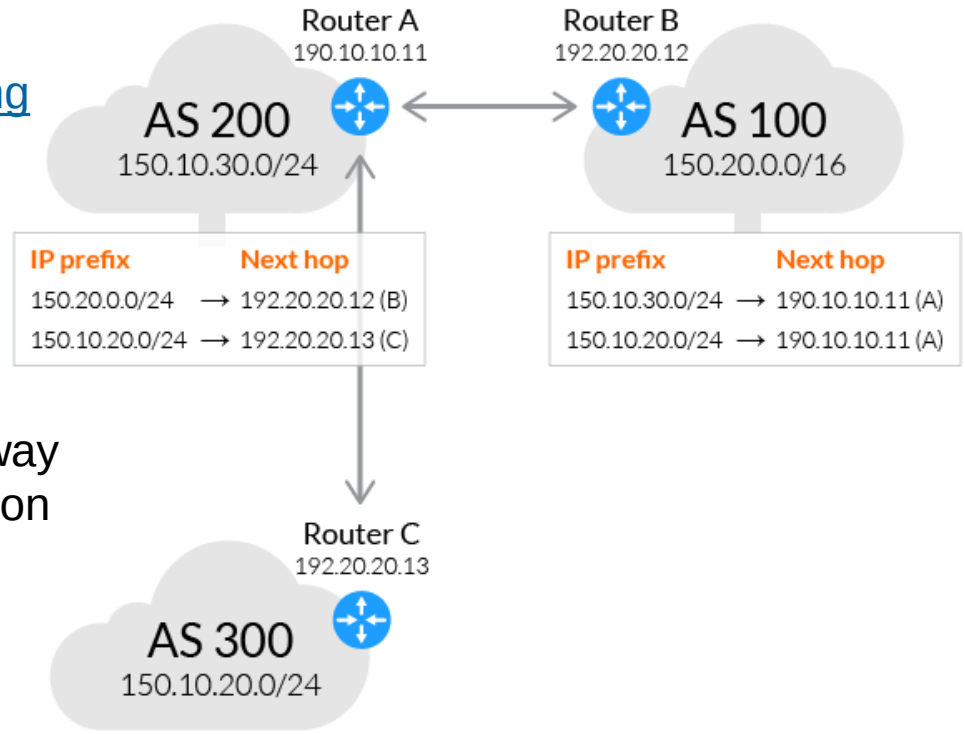
Agenda

- **Distributed Systems in the News**
- **Consistency**
- **Rsync**
- **Tor**
- **Message Queues / Apache Kafka**

Distributed Systems - In the *News*

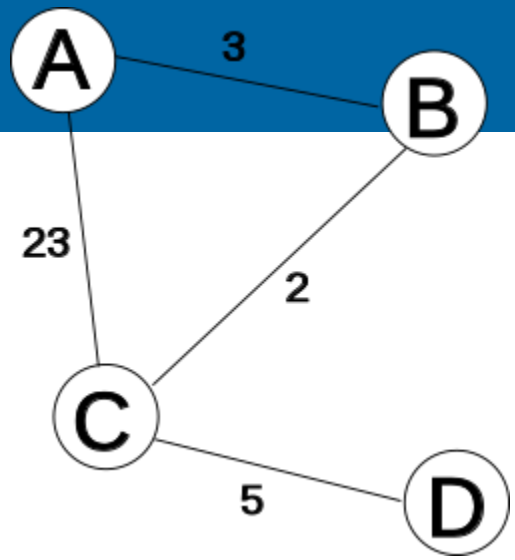
- **24.04.2018: MyEtherWallet Warns That A "Couple" Of Its DNS Servers Have Been Hacked**
  - More than 200 ETH were stolen
  - Another headline: Hackers emptied Ethereum wallets by breaking the basic infrastructure of the internet
  - What happened?

- **Hackers stole cryptocurrencies using a BGP leak**
  - Border Gateway Protocol (BGP) is a standardized exterior gateway protocol designed to exchange routing and reachability information among autonomous systems (AS) on the Internet (HSR)
  - Current version BGP4 used since 1994 (TCP)
  - BGP peer exchanges routing info with neighbor peers
    - Prefix announcements, keep alive messages
  - Each peer has a table with all routes – exchanges this information



https://www.incapsula.com/blog/bgp-routing-explained.html

Distributed Systems - In the *News*



- **AS559, maintained by switch, list**
  - 130.60.156.1 – UZH, also maintained by switch
- **~ distance-vector protocol**

T=1

from A	via A	via B	via C	via D
to A				
to B		3		
to C			23	
to D				

from B	via A	via B	via C	via D
to A	3			
to B				
to C			2	
to D				

from C	via A	via B	via C	via D
to A	23			
to B		2		
to C				
to D				5

from D	via A	via B	via C	via D
to A				
to B				
to C			5	
to D				

T=2

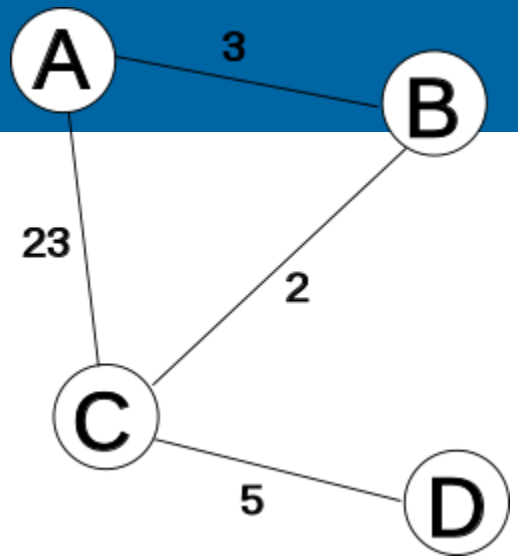
from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D			28	

from B	via A	via B	via C	via D
to A	3		25	
to B				
to C	26		2	
to D			7	

from C	via A	via B	via C	via D
to A	23	5		
to B	26	2		
to C				
to D				5

from D	via A	via B	via C	via D
to A			28	
to B			7	
to C			5	
to D				

Distributed Systems - In the *News*



- **T4: no new information**

T=3

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		7	
to B				
to C	8		2	
to D	31		7	

from C	via A	via B	via C	via D
to A	23	5		33
to B	26	2		12
to C				
to D	51	9		5

from D	via A	via B	via C	via D
to A			10	
to B			7	
to C			5	
to D				

T=4

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		7	
to B				
to C	8		2	
to D	13		7	

from C	via A	via B	via C	via D
to A	23	5		15
to B	26	2		12
to C				
to D	33	9		5

from D	via A	via B	via C	via D
to A			10	
to B			7	
to C			5	
to D				

Distributed Systems - In the News

- **BGP leak: announced by somebody not allowed by the owner**

- **Between approximately 11:05:00 UTC and 12:55:00 UTC on 24.4.2018**

  BGP4MP|04/24/18 11:05:42|A|205.251.199.0/24|10297
  BGP4MP|04/24/18 11:05:42|A|205.251.197.0/24|10297
  BGP4MP|04/24/18 11:05:42|A|205.251.195.0/24|10297
  BGP4MP|04/24/18 11:05:42|A|205.251.193.0/24|10297
  BGP4MP|04/24/18 11:05:42|A|205.251.192.0/24|10297
  ...
  BGP4MP|04/24/18 11:05:54|A|205.251.197.0/24|4826,6939,10297

- **E.g., 205.251.199.0 belongs to Amazon DNS services (AS16509), announced by eNet (AS10297)**
- **Rerouted DNS traffic to servers at Equinix, pointed to address in Russia**
  - Only affected site: myetherwallet.com, other sites got a SERVFAIL

Distributed Systems - In the News

- **Only some regions affected:**
  - Hurricane Electric (AS6939) has a strong presence Australia. Chicago was affected because AS10279 has a physical presence resulting in direct peering.
- **Who is to blame?**
  - The ISP that announced a subnet it did not own.
  - The transit providers that did not check the announcement before relaying it.
  - The ISPs that accepted the route.
  - The lack of protection on the DNS resolvers and authority.
  - The phishing website hosted on providers in Russia.
  - The website that did not enforce legitimate TLS certificates.
  - The user that clicked continue even though the TLS certificate was invalid.
- **How to fix?**
  - DNSSEC / DANE / DNS over HTTPS
  - HTTPS / HSTS – webservers declare to only use HTTPS only

Distributed Systems - In the News

- **25.04.2018:** [Multiple Exchanges Suspend ERC20 Token Trading Due To Potential BatchOverflow Bug]
  - Integer overflows in multiple contracts
  - Poloniex suspended all ERC20 token deposits and withdrawals,
  - OKEX's decision to halt ERC20 deposits

```
255   function batchTransfer(address[] _receivers, uint256 _value) public whenNotPaused returns (bool) {
256     uint cnt = _receivers.length;
257     uint256 amount = uint256(cnt) * _value;
258     require(cnt > 0 && cnt <= 20);
259     require(_value > 0 && balances[msg.sender] >= amount);
260
261     balances[msg.sender] = balances[msg.sender].sub(amount);
262     for (uint i = 0; i < cnt; i++) {
263         balances[_receivers[i]] = balances[_receivers[i]].add(_value);
264         Transfer(msg.sender, _receivers[i], _value);
265     }
266     return true;
267   }
268 }
```

From the Exercises

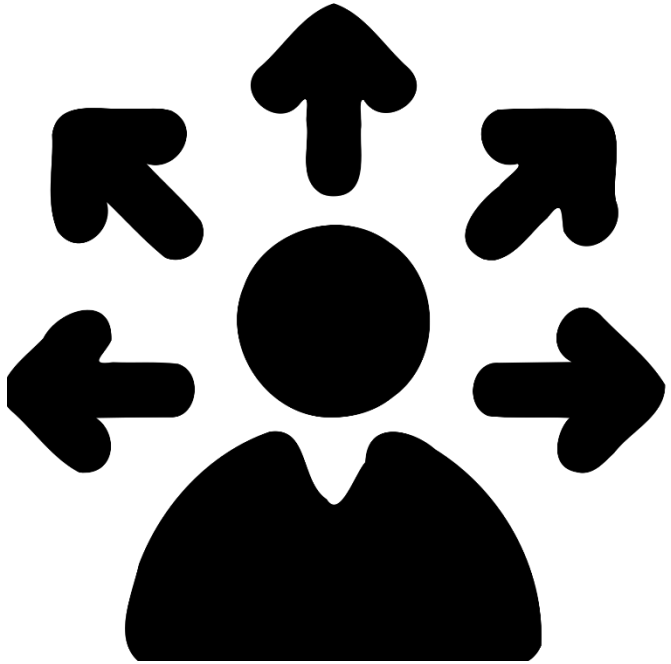
- **Attacking the DHT**
  - Create a key for a data item close to the target:
    Number160.createHash(data).xor(new Number160(0)) – distance 0, perfect match
    Number160.createHash(data).xor(new Number160(1)) – distance 1
    Number160.createHash(data).xor(new Number160(2)) – distance 2
    …
  - Or create key of node close to the target
    new PeerBuilder(new Number160(RND)).ports(port).start(), where RND is
    Number160.createHash(data).xor(new Number160(0))
    Number160.createHash(data).xor(new Number160(1))
    …
  - Peer can then answer there is no data
  - For previously known values / peers (known public key)
    - Cannot change data, but make it disappear
- **Previous exams: only from UZH, uploaded to moodle**

URL: b.socrative.com/student/

Room: HSR

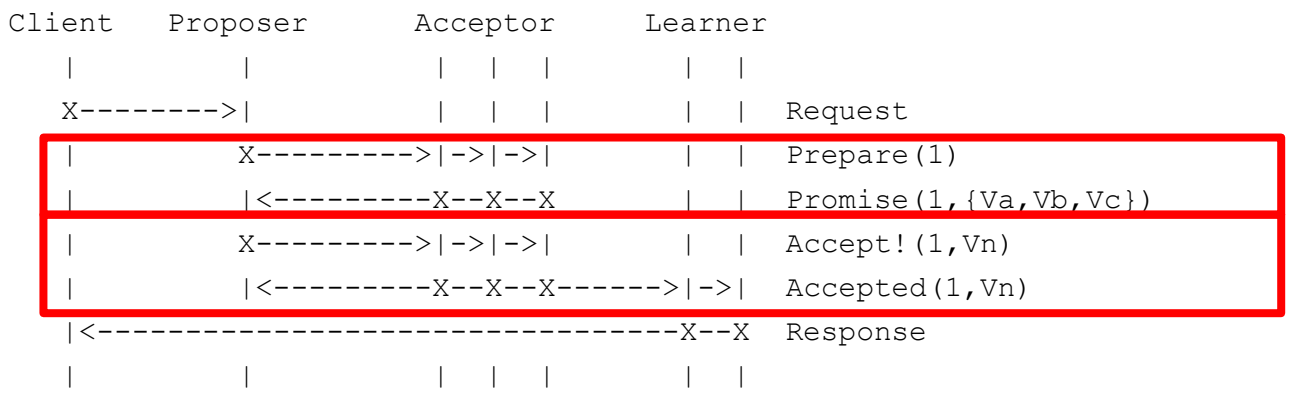
Consistency

- **DHTs have weak consistency**
  - Peer A put X.1, Peer B gets X.1 modifies it puts B.2
  - Same time: Peer C gets X.1 modifies it puts C.2
    - Which one is stored B.2 of B or C.2 of C?
- **Consistency generic issue in distributed systems**
  - Coordinator required:
    - easy solution: centralized
    - Interesting solution: decentralized, in case failed peer, pick another peer
- **Coordinator needs to be defined**
  - Election, example Paxos



Consistency – Paxos – Leader Election

- **Paxos**
  - Protocol family for consensus (multi, cheap, fast, generalized, …)
  - Roles: Client/Proposer (requester), Acceptor (voter), Leader (coordinator), Learner (responder)
    - Client sends requests to a proposer
    - Proposer send proposal acceptor, send back promise
    - If majority promises, send value to acceptor, acceptor sent to learner
    - Learner sent result to client
- **2 Phases**
  - Phase 1: prepare / promise
  - Phase 2: accept / accepted



http://en.wikipedia.org/wiki/Paxos_%28computer_science%29

Consistency – Raft/Other – Leader Election

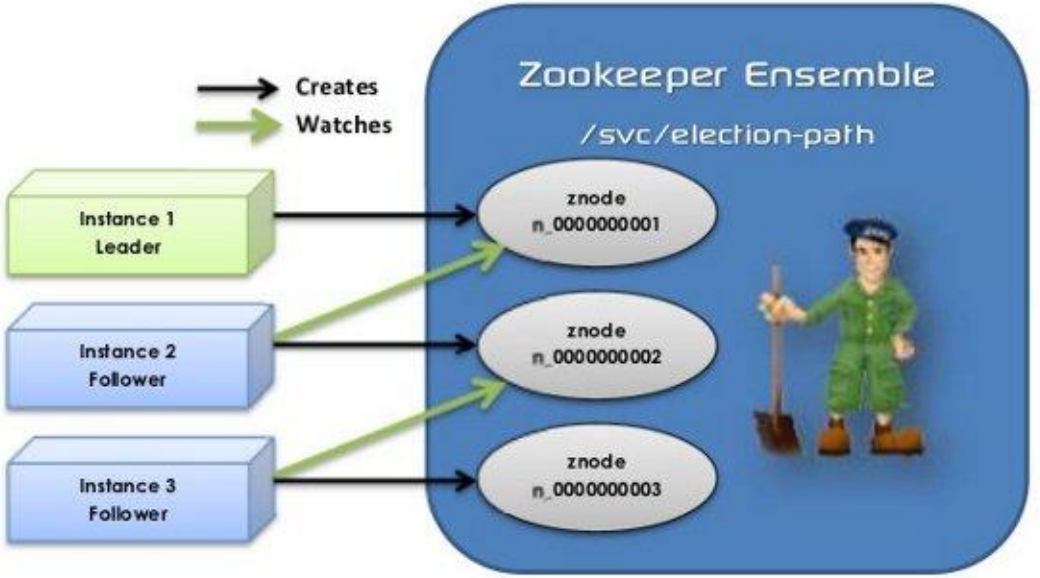
- **Raft – Alternative to Paxos (easier), three roles: leader, follower, candidate**
  - Paxos has a lot of roles: client, proposer, learner, acceptor, leader, follower. Implement Paxos → figure out all of those roles, and how to implement them.
  - Paxos (not multi paxos) is single-consensus protocol – designed to consensus once
  - Raft applies randomized timers to the heartbeat process to achieve leader election.
  - In short: “Paxos has more complexity than it needs, and despite that, it needs to be tweaked to be really useful, and getting those tweaks right is hard.”
- **Chubby is based on Multi-Paxos**
- **etcd is built on top of Raft (using gRPC)**

		Usage Patterns								
Project	**Consensus System**	**SR**	**LR**	**SS**	**BO**	**SD**	**GM**	**LE**	**MM**	**Q**
GFS	Chubby			✓				✓	✓	
Borg	Chubby/Paxos	✓				✓		✓		
Kubernetes	etcd						✓		✓	
Megastore	Paxos		✓							
Spanner	Paxos	✓								
Bigtable	Chubby						✓	✓	✓	
Hadoop/HDFS	ZooKeeper	✓						✓		
HBase	ZooKeeper	✓		✓			✓		✓	
Hive	ZooKeeper			✓					✓	
Configerator	Zeus								✓	
Cassandra	ZooKeeper					✓		✓	✓	
Accumulo	ZooKeeper		✓	✓					✓	
BookKeeper	ZooKeeper						✓		✓	
Hedwig	ZooKeeper						✓		✓	
Kafka	ZooKeeper						✓	✓	✓	
Solr	ZooKeeper							✓	✓	✓
Giraph	ZooKeeper		✓		✓				✓	
Hama	ZooKeeper				✓					
Mesos	ZooKeeper							✓		
CoreOS	etcd					✓				
OpenStack	ZooKeeper					✓				
Neo4j	ZooKeeper			✓				✓		

http://container-solutions.com/raft-explained-part-1-the-consenus-problem/

Consistency – ZooKeeper – Leader Election

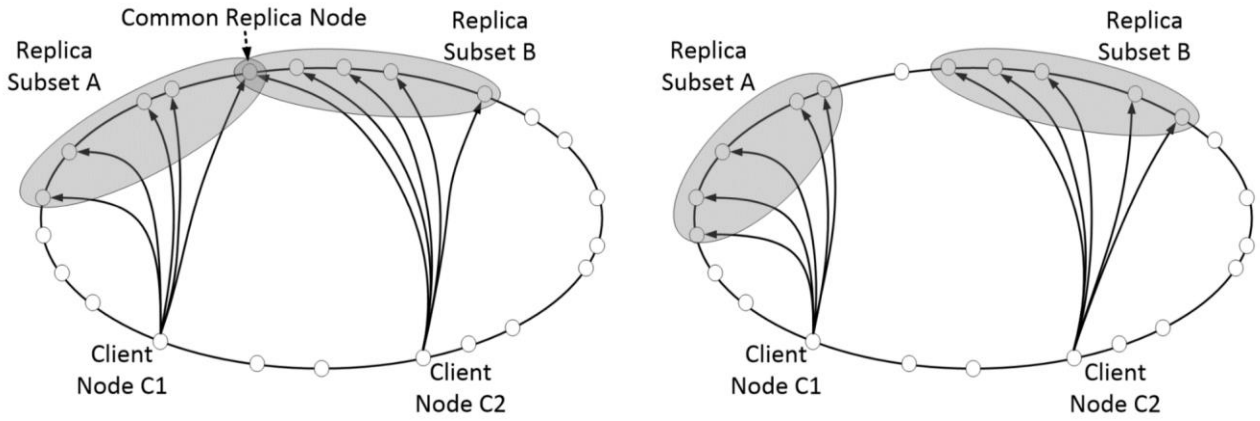
- **Apache Kafka - Leader election**
  - All nodes create a sequential node with path: /app/leader_election/guid_
  - Append the 10-digit sequence number to the path: /app/leader_election/guid_0000000001, /app/leader_election/guid_0000000002
  - Smallest number becomes the leader
  - Each follower node watches the next smallest number. E.g., the node /app/leader_election/guid_0000000008 will watch /app/leader_election/guid_0000000007
  - If leader goes down, corresponding /app/leader_election gets deleted
  - Next in line follower node will get the notification through watcher about the leader removal
  - Next in line follower node will check if there are any smaller number. If none, then it will assume the role of the leader.



https://www.outbrain.com/techblog/2011/07/leader-election-with-zookeeper/

Consistency – DHT – Leader Election

- **Consistency in DHTs – vDHT**
  - Paxos and DHTs [1], [2]
  - vDHT: CoW, versions, 2PC, replication, software transactional memory (STM) → for consistent updates. Works for light churn
    - No locking, no timestamps (replication time may have an influence)
    - Every update – new version
      1. get latest version, check if all replica peers have latest version, if not wait and try again
      2. put prepared with data and short TTL, if status is OK on all replica peers, go ahead, otherwise, remove the data and go to step 1.
      3. put confirmed, don't send the data, just remove the prepared flag
- **In case of heavy churn, API user needs to resolve**



URL: b.socrative.com/student/

Room: HSR

Protocols

- **Avro / gRPC / ASN1**

```
@namespace("ch.hsr.dsl")

protocol MyProtocol{
  record AMessage {
     string request;
     int code;
  }
  record BMessage {
     string reply;
  }

  BMessage GetMessage(AMessage msg);
}
```

Protocols

- **Avro / gRPC / ASN1**

```
30 13 02 01 05 16 0e 41 6e 79 62 6f 64 79 20 74 68 65 72 65 3f


30 — type tag indicating SEQUENCE
13 — length in octets of value that follows
  02 — type tag indicating INTEGER
  01 — length in octets of value that follows
    05 — value (5)
  16 — type tag indicating IA5String
     (IA5 means the full 7-bit ISO 646 set, including
variants,
      but is generally US-ASCII)
  0e — length in octets of value that follows
    41 6e 79 62 6f 64 79 20 74 68 65 72 65 3f — value
("Anybody there?")


<FooQuestion>
    <trackingNumber>5</trackingNumber>
    <question>Anybody there?</question>
</FooQuestion>
```

Protocols

- **Avro / gRPC / ASN1**
- **Custom encoding/decoding**

```
115     public static boolean decodeHeader(final ByteBuf buffer, final InetSocketAddress recipientSocket,
116             final InetSocketAddress senderSocket, final Message message) {
117         LOG.debug("Decode message. Recipient: {}, Sender:{}.", recipientSocket, senderSocket);
118         final int versionAndType = buffer.readInt();
119         message.version(versionAndType >>> 4);
120         message.type(Type.values()[(versionAndType & Utils.MASK_0F)]);
121         message.protocolType(ProtocolType.values()[versionAndType >>> 30]);
122         message.messageId(buffer.readInt());
123         final int command = buffer.readUnsignedByte();
124         message.command((byte) command);
125         final Number160 recipientID = Number160.decode(buffer);
126
127         //we only get the id for the recipient, the rest we already know
128         final PeerAddress recipient = PeerAddress.builder().peerId(recipientID).build();
129         message.recipient(recipient);
130
131
132         final int contentTypes = buffer.readInt();
133         message.hasContent(contentTypes != 0);
134         message.contentTypes(decodeContentTypes(contentTypes, message));
```

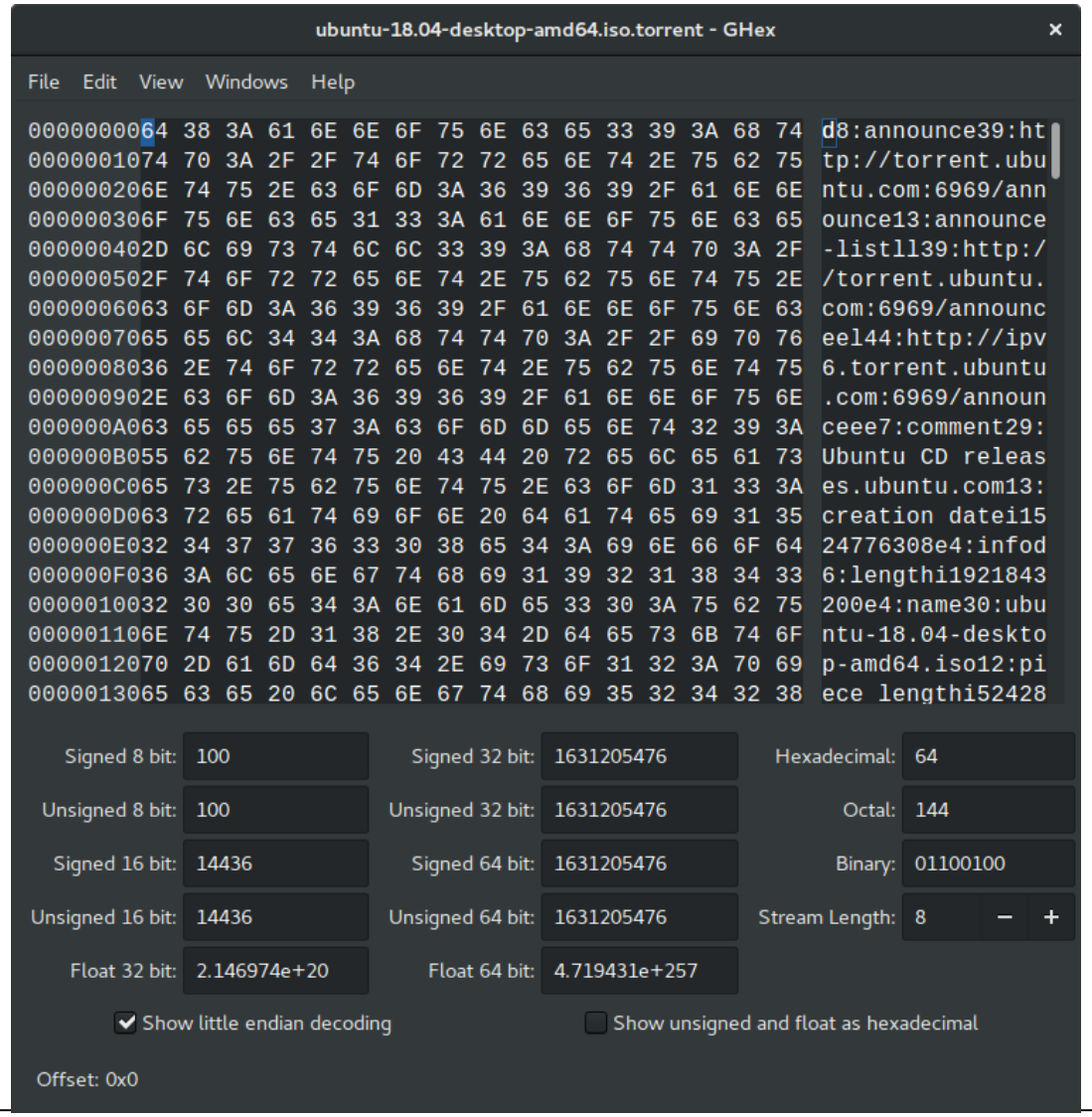
Protocols

- **Avro / gRPC / ASN1**
- **Custom encoding/decoding**
- **JSON encoding**

```
[
    {
        "id": "bitcoin",
        "name": "Bitcoin",
        "symbol": "BTC",
        "rank": "1",
        "price_usd": "9324.08",
        "price_btc": "1.0",
        "24h_volume_usd": "9039300000.0",
        "market_cap_usd": "158560288125",
        "available_supply": "17005462.0",
        "total_supply": "17005462.0",
        "max_supply": "21000000.0",
        "percent_change_1h": "0.46",
        "percent_change_24h": "-0.27",
        "percent_change_7d": "4.5",
        "last_updated": "1525011874"
    }, ...
]
```

Protocols

- **Avro / gRPC / ASN1**
- **Custom encoding/decoding**
- **JSON encoding**
- **Benconding**
  - Integers: i42e
  - Byte string: 4:test
  - List: l4:testi42ee
  - Map/dictionary: d4:test3:hsr3:tomi42ee
- **What if (partial) data is already on the other side?**

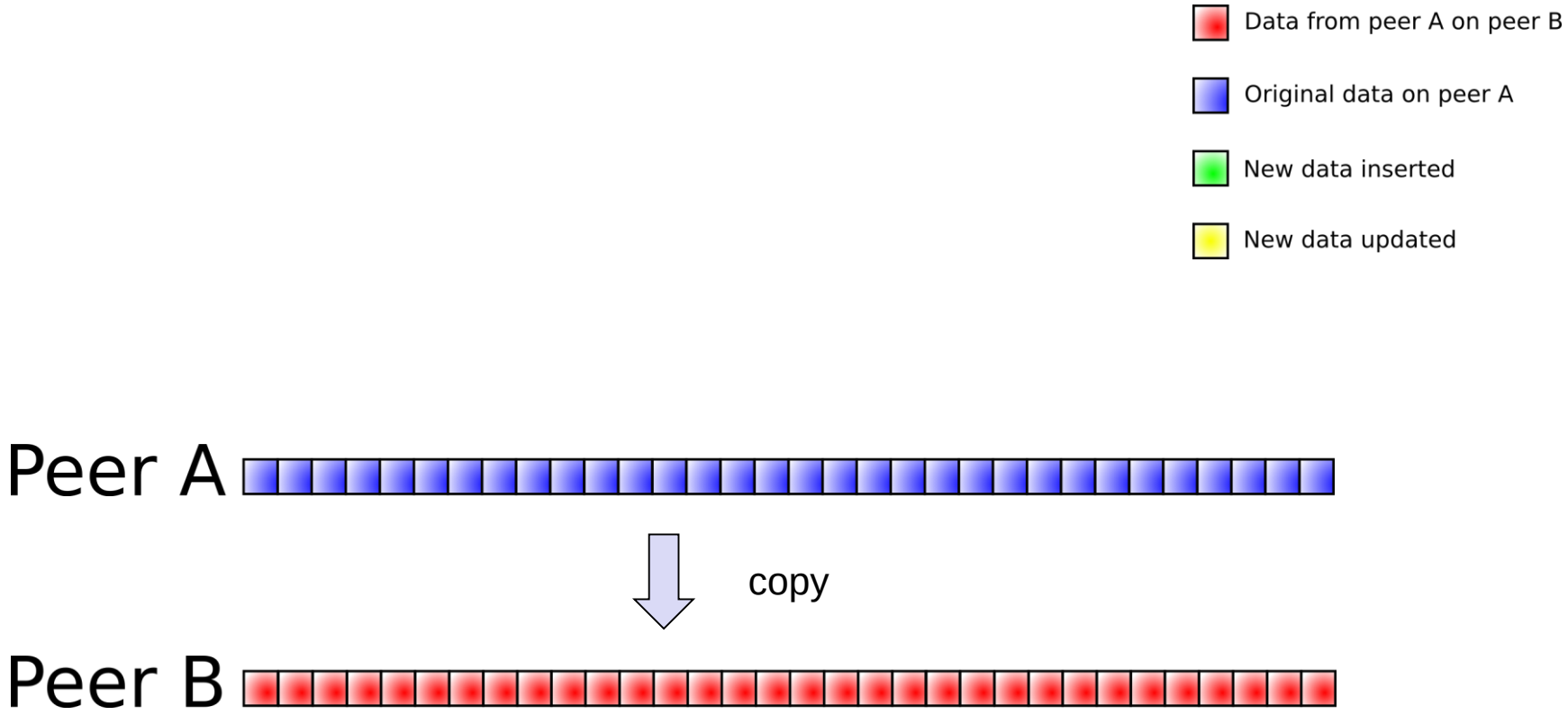


Rsync - Introduction

- **Rsync used to synchronize data over network**
  - Minimizing data transfer (delta)
- **Command line client (standard utility)**
  - E.g. `rsync -aP --link-dest=$HOME/Backups/current /path/to/important_files $HOME/Backups/back-$date`
  - Unchanged files are hard linked (`--link-dest`) → Can be used for incremental backups
- **Main idea**
  - Receiver compute two checksums (strong, weak) → sent to sender
  - Sender computes with weak checksum and checks for known blocks
  - Sender verifies with strong checksum → sends difference to receiver
- **Example with two peers:**

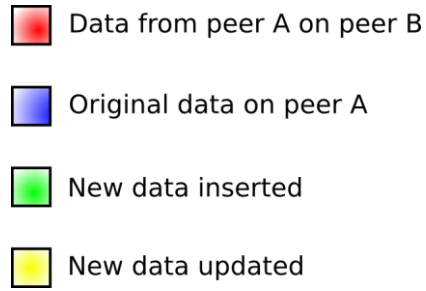
Rsync - Example

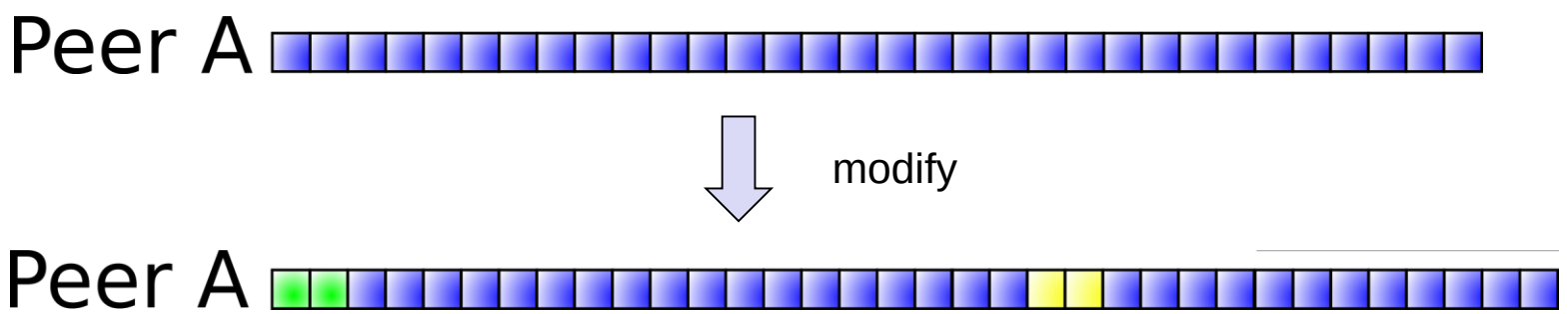
- **Peer B does not have the data → peer A copies it to peer B, no need for rsync**



Rsync - Example

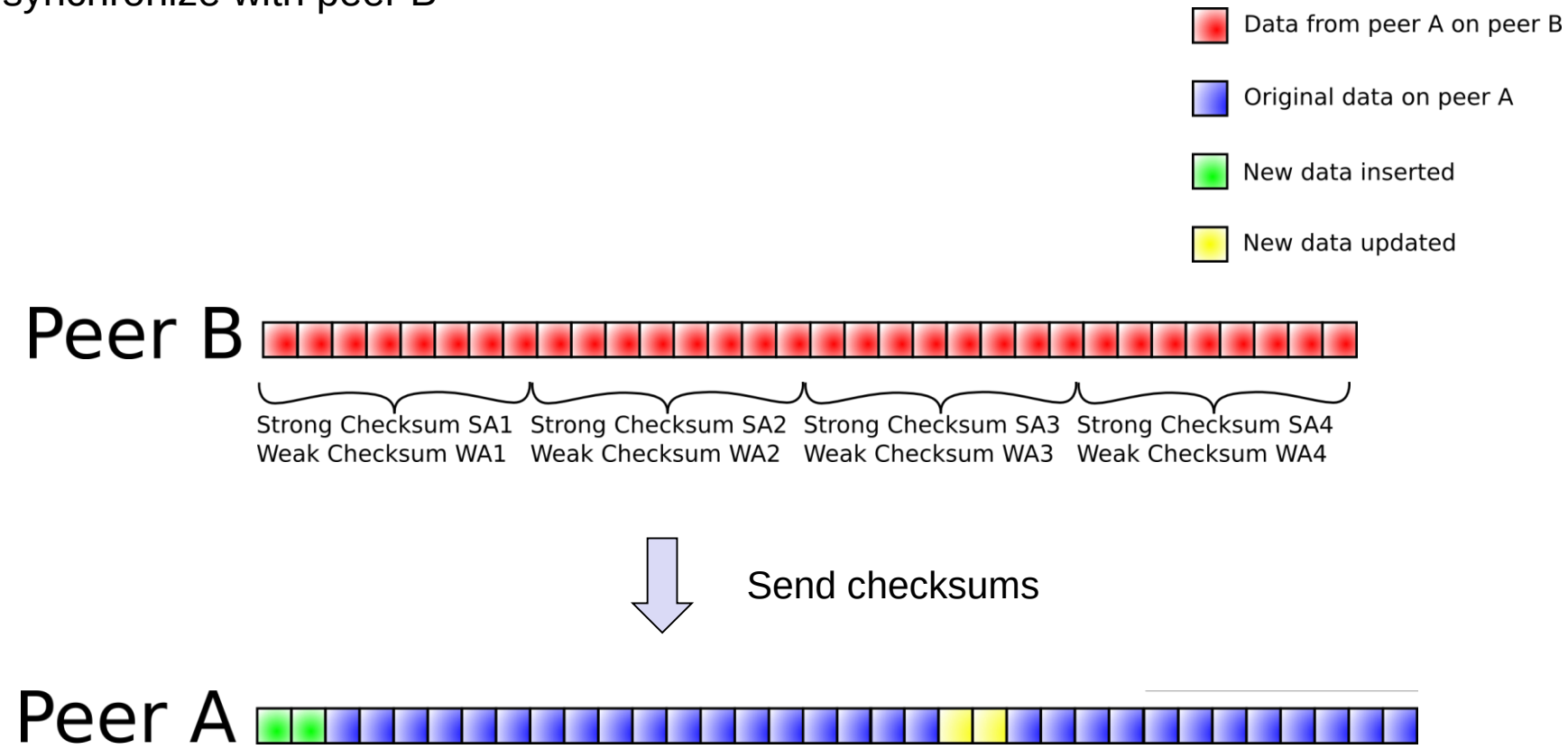
- **Peer A modifies data (insert, update)**
  - Wants to synchronize with peer B



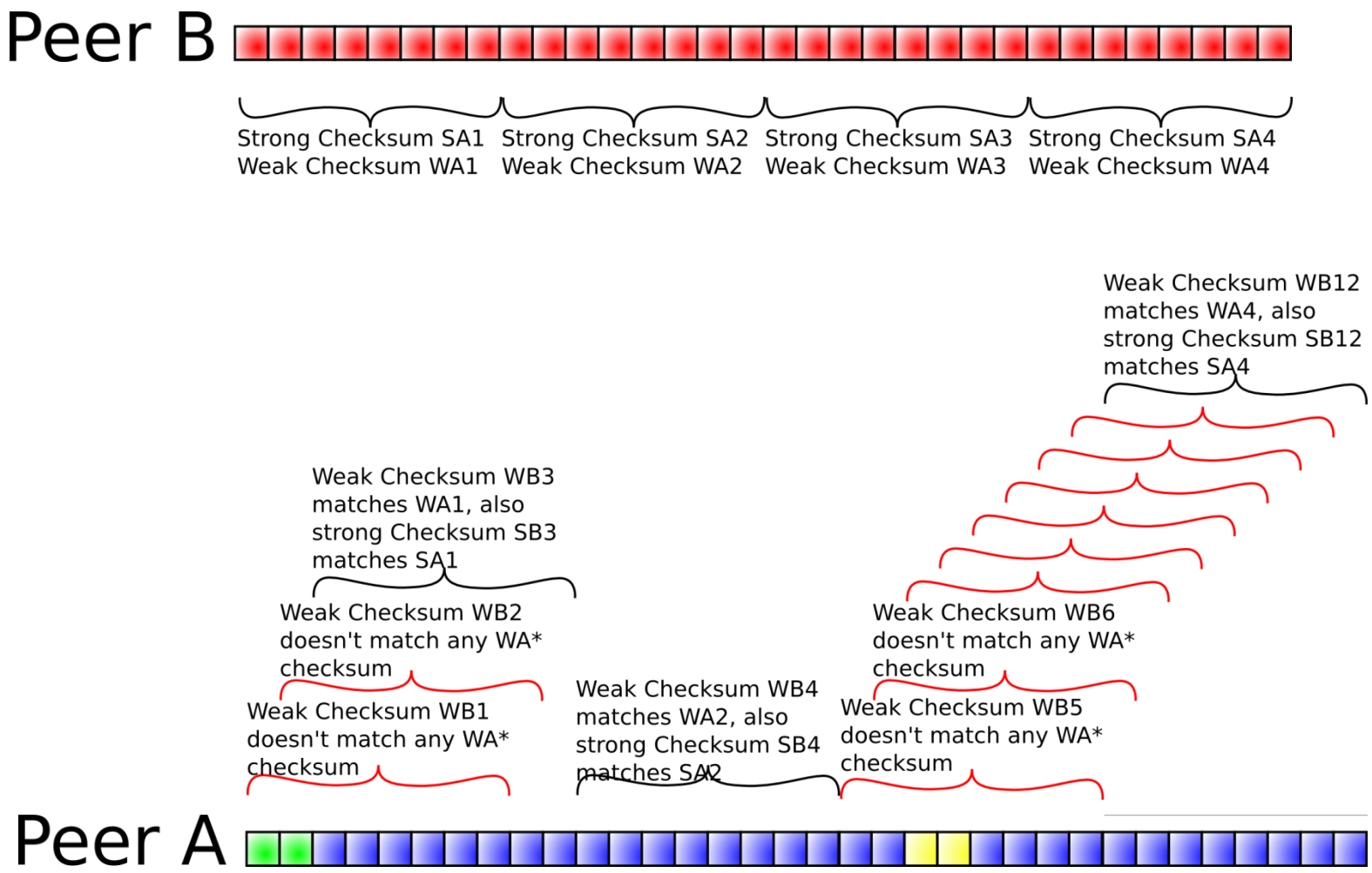


Rsync - Example

- **Peer A modifies data (insert, update)**
  - Wants to synchronize with peer B

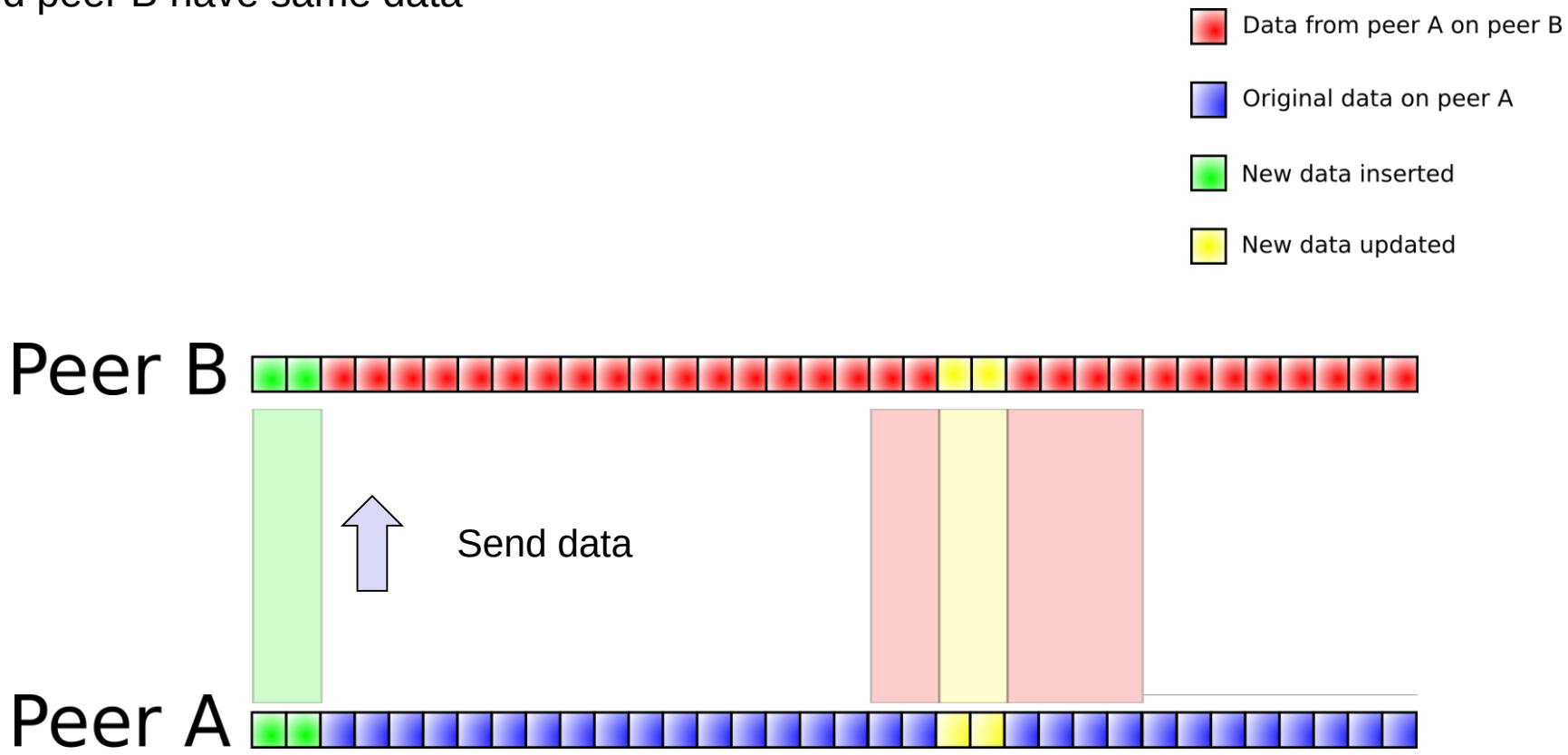


Rsync - Example



Rsync - Example

- **Peer A sends 2 + 8 blocks to peer B**
  - Peer A and peer B have same data



URL: b.socrative.com/student/

Room: HSR

Tor

- **The Onion Router**
- **Started ~1995 by U.S. Naval Research Laboratory**
  - Protect US intelligence communication
  - Naval Research Laboratory released the code under free license in 2004
  - EFF began funding development
- **Anonymous internet communication system**
  - SSL / TLS is not enough
  - protect privacy of users
- **Encrypts all messages (also header, IP)**
- **Sends data through virtual circuit (3 random relays)**
  - Use SOCKS proxy to connect via Tor
  - Example Tor Browser accessing http://tomp2p.net

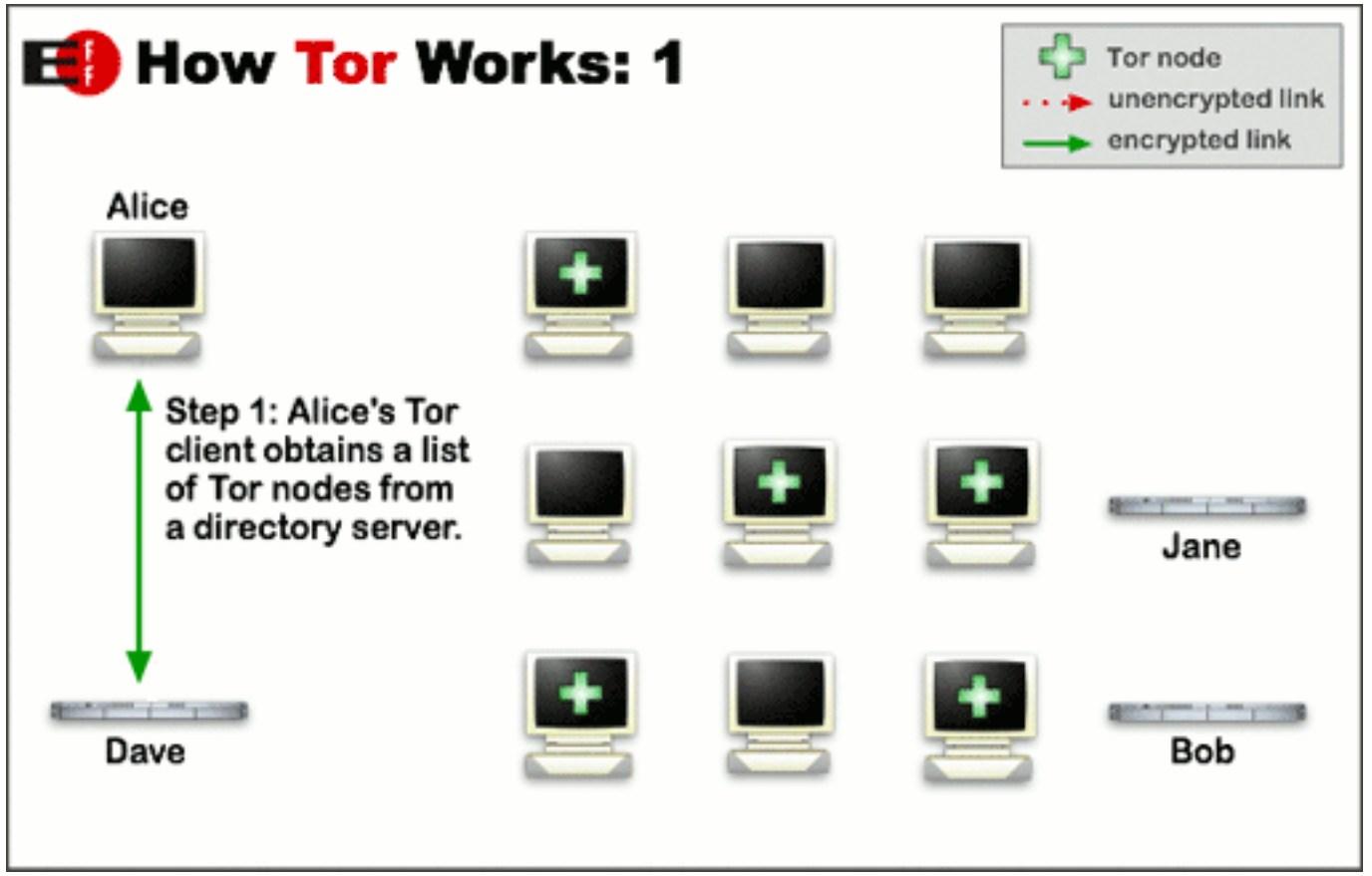
Tor

- **Hidden services:**
  - Servers configured to receive connections only through Tor
  - Silk road was using hidden services + Bitcoins → bad press
  - Using DHT
- **Main use-case: journalists, whistleblowers, and dissidents**
  - Can be used against price discrimination
- **Attention**
  - Tor exit node can see traffic! – Use SSL/TLS
  - Services block Tor exit nodes, e.g., example: Wikipedia
  - Tor exit node operator facing copyright claims - template
- **Large project: 670’000 LoC**
- **TCP only**

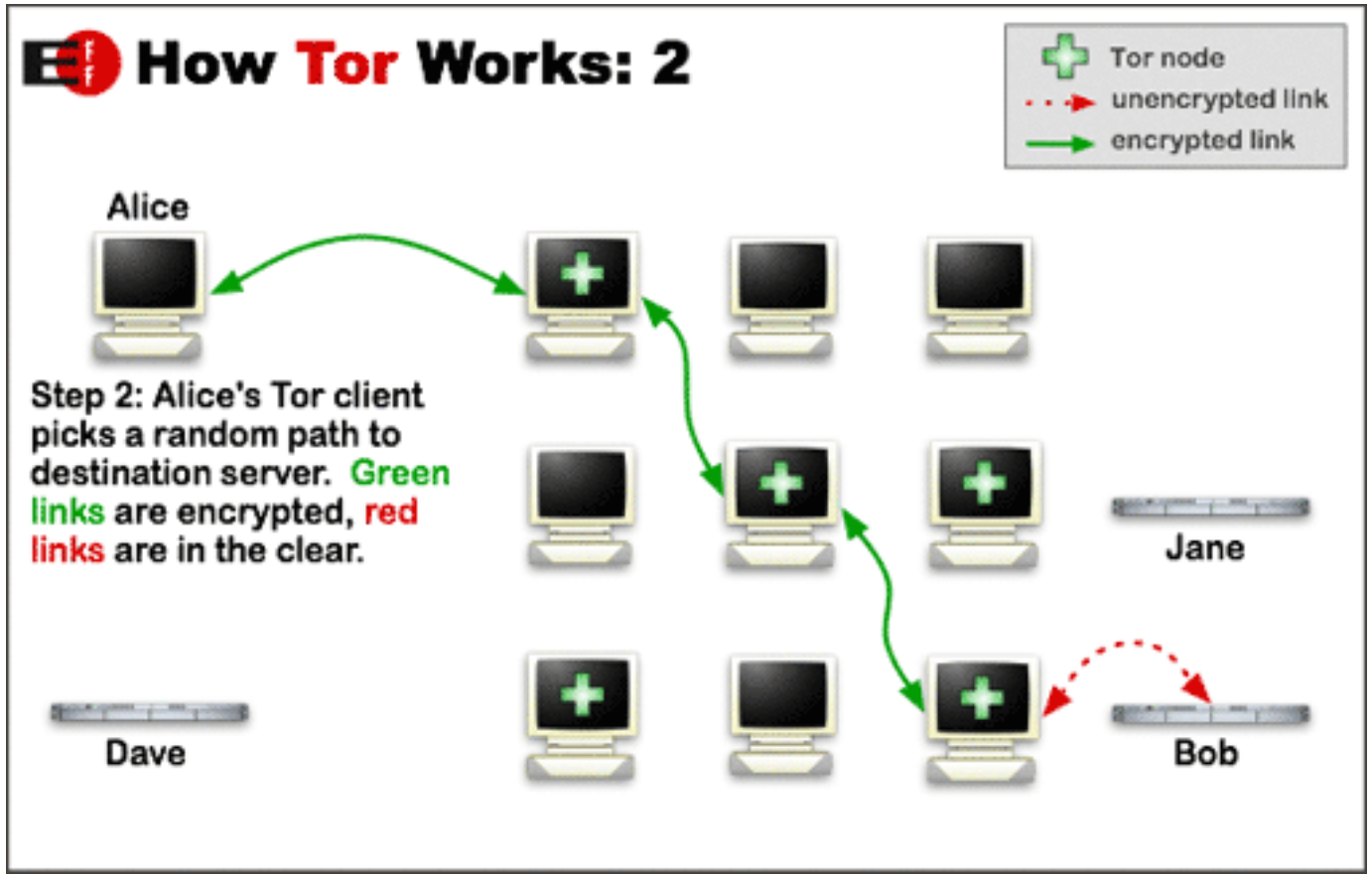
1. Install a Web Server
   ```
   apt-get install apache2
   ```
2. Install TOR
   ```
   apt-get install tor
   ```
3. Edit config file
   ```
   HiddenServiceDir /var/lib/tor/…
   HiddenServicePort 80 127.0.0.1:8080
   ```
4. Start TOR, start web server

Tor

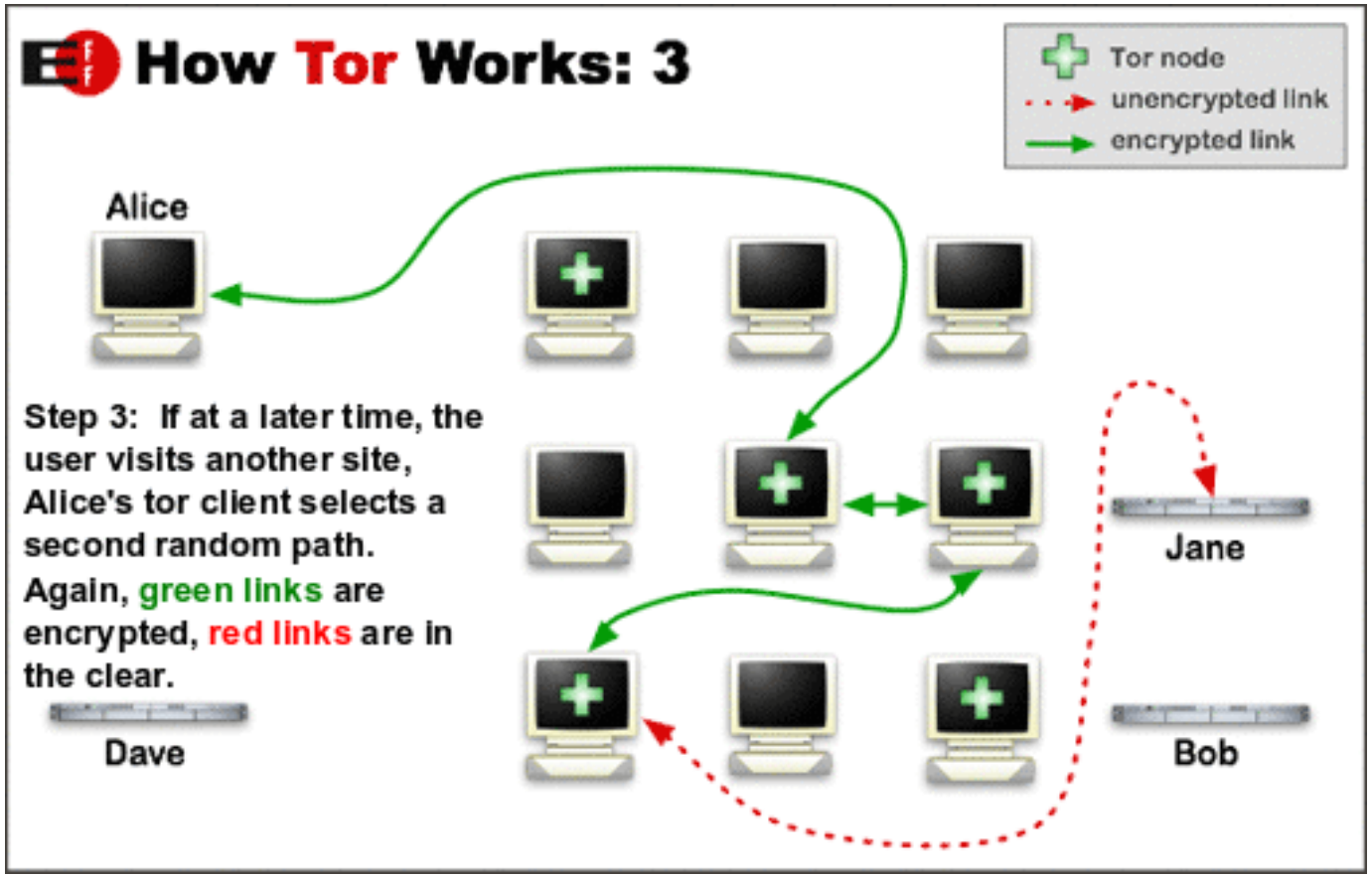
- How it works



- **Alice to Bob**

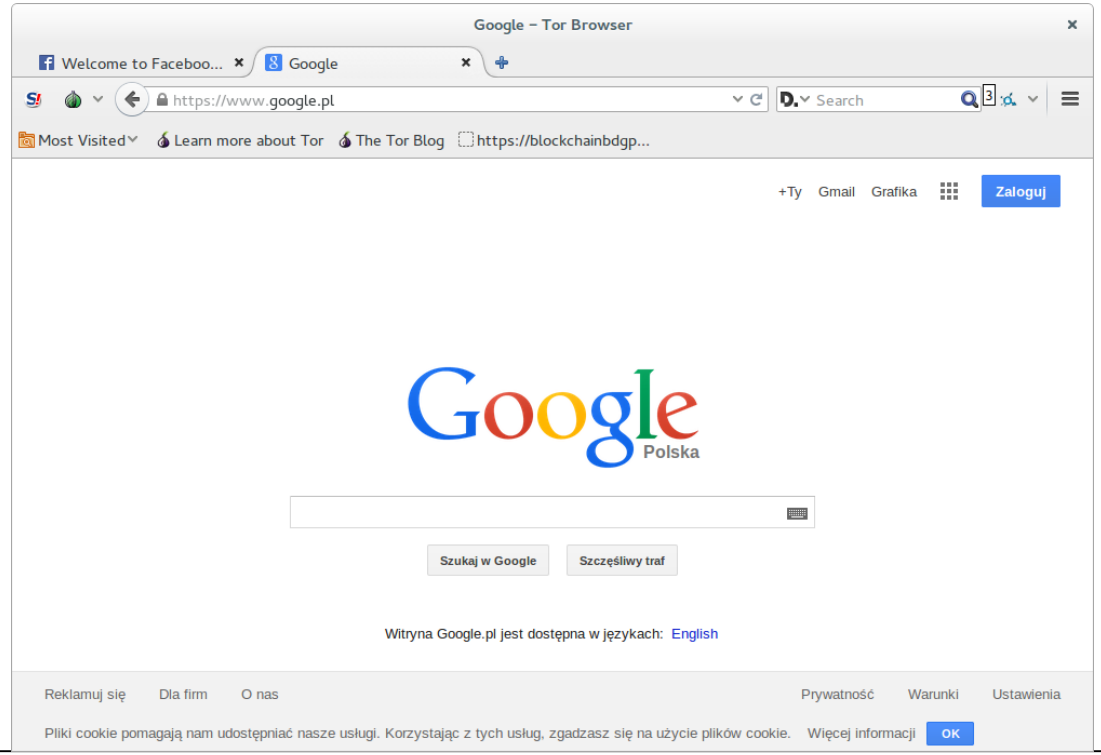

How Tor Works: 2
Tor node
unencrypted link
encrypted link
Alice
Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.
Jane
Dave
Bob

- **Alice to Jane**



Tor

- **Don't provide your name or other revealing information in web forms**
- **Data is only encrypted within TOR – if no HTTPS is used, it still can be read**
- **Bomb threats at Harvard**
  - FBI figured out, Tor was used – check who was using Tor in the Harvard network → 2 days later student was caught.
- **Language may be different**

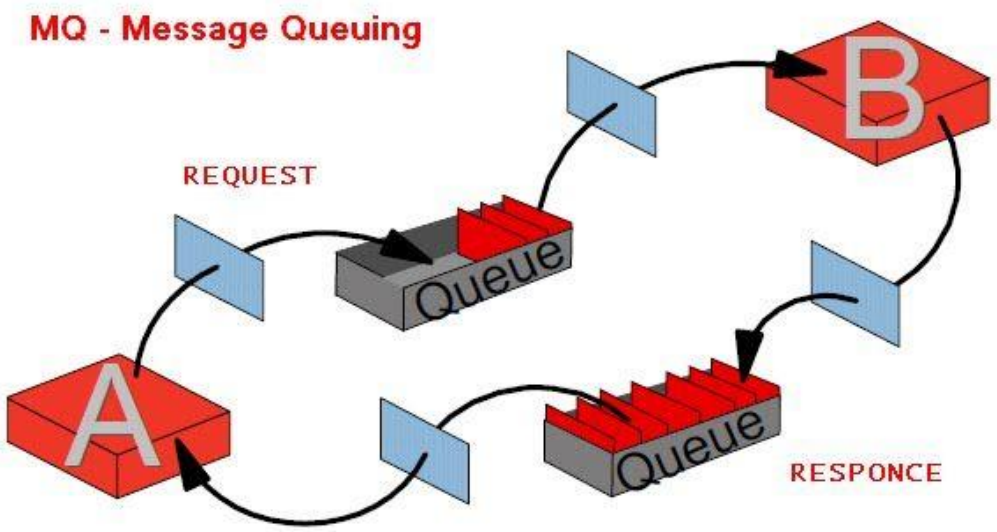


URL: b.socrative.com/student/

Room: HSR

Message Queues

- **Message queues and mailboxes**
  - Software-engineering components used IPC
  - Asynchronous communications protocol
    - no need to interact with the message queue at the same time
- **Standardization**
  - Sun Microsystems' JMS specification – early attempt
    - Advanced Message Queuing Protocol (AMQP) – feature-rich message queue protocol (HTTP)
    - Streaming Text Oriented Messaging Protocol (STOMP) – simple, text-oriented message protocol (HTTP)
    - MQTT (formerly MQ Telemetry Transport) - lightweight message queue protocol especially for embedded devices (TCP/IP)
- **RabbitMQ, ActiveMQ, QPID, ZeroMQ**
  - Accepts, stores and forwards binary blobs (messages)

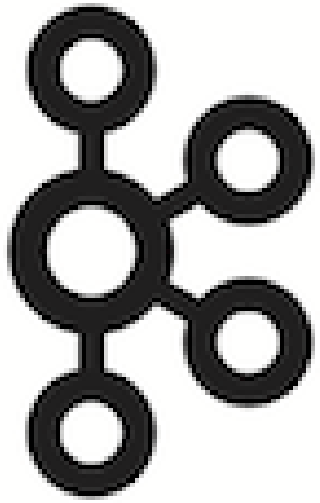


Message Queue / Apache Kafka

- **Message queue use-case:**
  - Applications (producer) push messages to broker
  - Broker stores message until consumer takes messages (ZeroMQ can run without dedicated broker)
- **Benefits:**
  - Decoupled systems, scalability, resilience, redundancy, monitoring
- **Apache Kafka: Distributed streaming platform / Message Queue**
  - Read and write streams of data (message queue)
  - Store streams of data in a distributed, replicated fault-tolerant cluster
  - Stream processing (Stream API) – aggregation or joining streams
- **Guarantees**
  - Messages sent by a producer to a topic partition will be appended in the order they are sent
  - A consumer instance sees records in the order they are stored in the log
  - For a topic with replication factor N, tolerate up to N-1 failures without losing any records in the log

Apache Kafka

- **Kafka uses ZooKeeper**
  - Distributed hierarchical key-value storage
  - Fast Processing in "read-dominant" workloads
  - Scalable: performance can be improved by adding nodes.
  - Used at Rackspace, Yahoo, Reddit, eBay, …



- **Example – github**
  - Consumer / producer / streaming
  - Streams / Table

Questions?

Prof. Dr. Thomas Bocek
thomas.bocek@hsr.ch