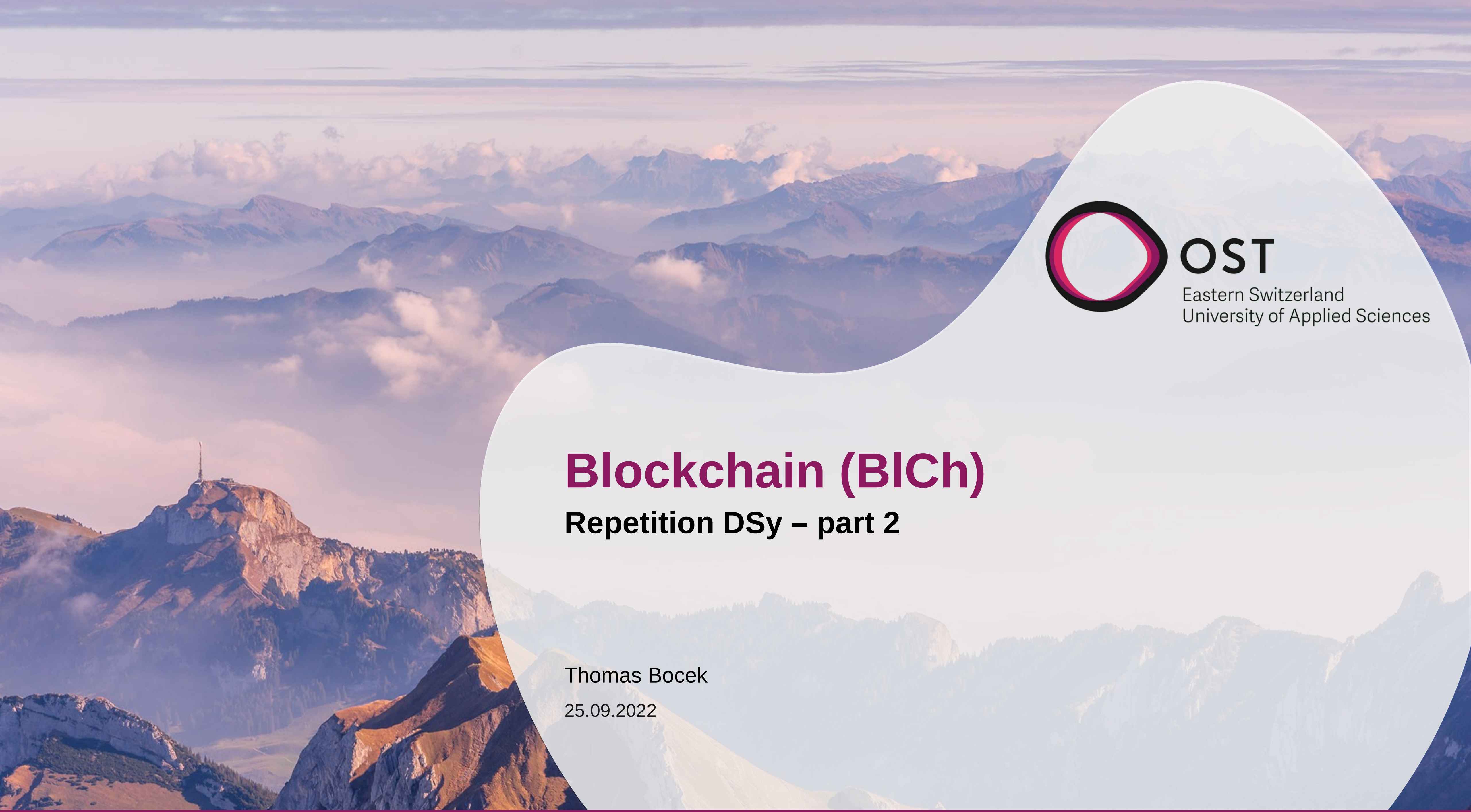


Eastern Switzerland
University of Applied Sciences

Blockchain (BlCh)

Repetition DSy – part 2

Thomas Bocek

25.09.2022

Lecture 6

Authentication

- Authentication
  - Single-factor authentication
    - E.g. password
  - Multi-factor authentication / 2FA
    - E.g. password **and** software token, SMS (15.03.2021)
- Password rules
  - Don’t use:
    - The name of a pet, child, family member, or significant other
    - Anniversary dates and birthdays
    - Birthplace
    - Name of a favorite holiday
    - Something related to a favorite sports team
    - The word "password“
  - Don’t’ reuse passwords, use password managers

- Don’t enter passwords on unencrypted sites
- Password length:
  password cracking with 5000$ in 2018 with hashcat
  - Hashtype: WPA/WPA2: 1190.5 kH/s

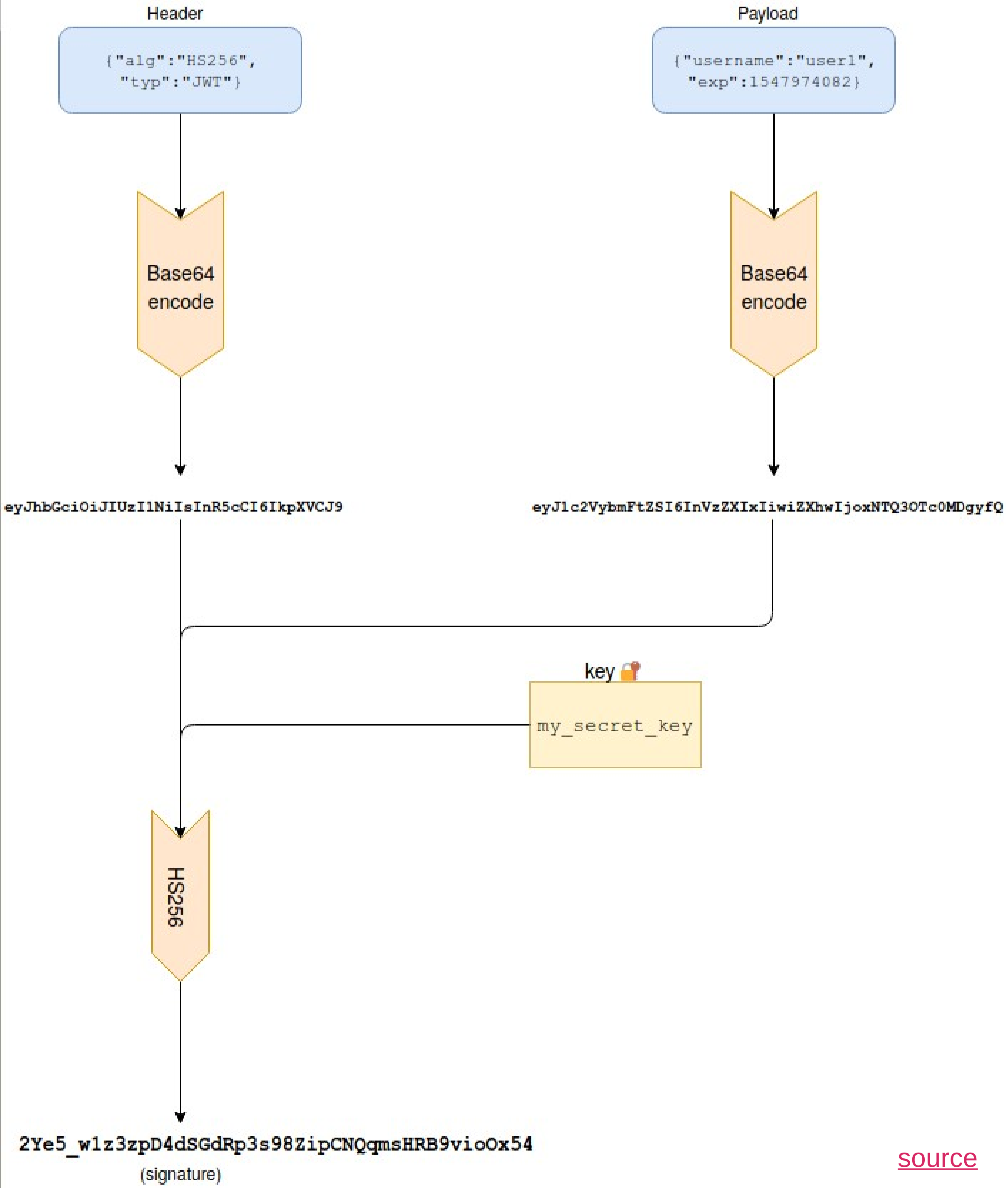
Pw length	Combinations	Time
6	11m	9s
7	656m	9m
8	38b	8h
9	7 *10^{15}	186y
10	4 *10^{17}	11ky
11	2 *10^{19}	665ky
12	1 *10^{21}	38my

- Combinations depend on PW complexity

Authentication

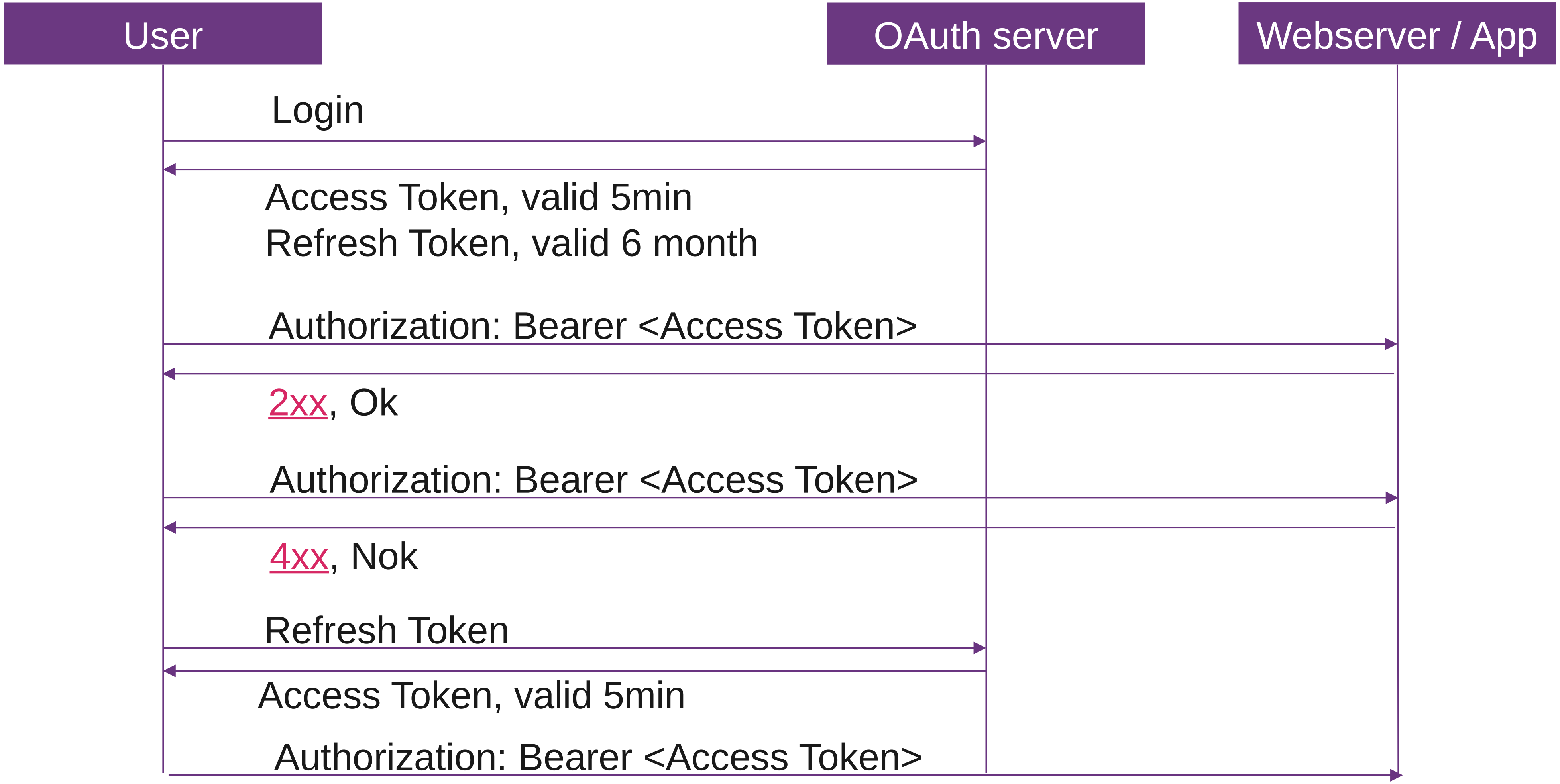
- JSON-based access tokens
  - Header: {"alg" : "HS256"}
  - Payload: {“sub" : "tom", "role" : "admin", “exp" : 1422779638}
- Signature (simple): keyed-hash message
  - ~hash(base64(header)+base64(payload) + secret token)
- Client can store user_token in
  - localStorage.setItem(“token", userToken);
- Example in golang with JWT
  - Tutorial: here and here

- OAuth - protocol for authorization 3rd party integration
  - Grant access on other websites without giving them the passwords



source

Access Token / Refresh Token



User

OAuth server

Webserver / App

Login

Access Token, valid 5min
Refresh Token, valid 6 month

Authorization: Bearer <Access Token>

2xx, Ok

Authorization: Bearer <Access Token>

4xx, Nok

Refresh Token

Access Token, valid 5min

Authorization: Bearer <Access Token>

Access Token / Refresh Token

- Access Token only short lifetime, e.g., 10min.
  - If public key / secret is known, the content in the token can be trusted, e.g., in the serivce
  - Can have userId, role, etc.
    - No need to query DB for those information, e.g.:

```
type TokenClaims struct {
	MailFrom string `json:"mail_from,omitempty"`
	MailTo   string `json:"mail_to,omitempty"`
	jwt.Claims
}
```

- Refresh Token longer lifetime, e.g., 6 month
  - A refresh token is used to get a new access token
  - IAM / Auth server creates access tokens

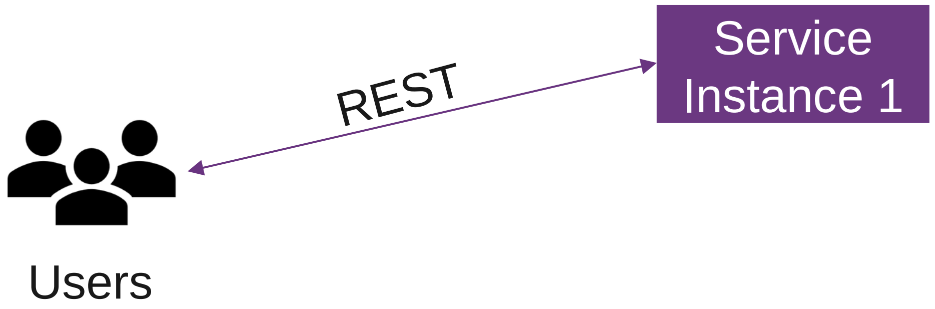
- Only access token, with long lifetime
  - If a user credential is revoked – how to inform every service?
- Only refresh token
  - Tightly coupled Service/Auth, every request to Service, Auth needs to be involved for every access
- Access + Refresh token
  - If a user credential is revoked, user has max. 10min more to access service
  - Auth only involved if access token is expired
- Authorization Code Flow with Proof Key for Code Exchange (PKCE)

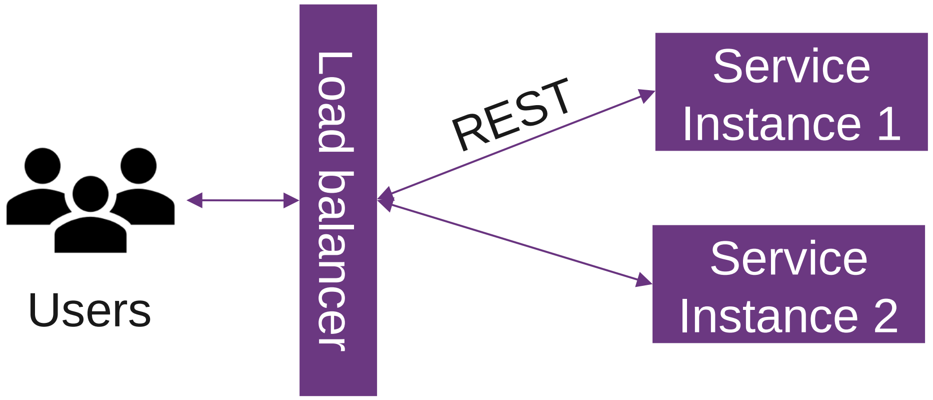
Load Balancing

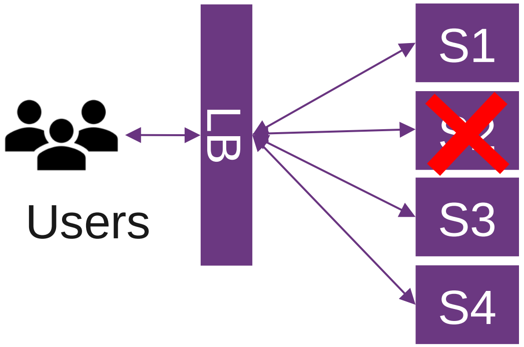
- What is load balancing
  - Distribution of workloads across multiple computing resources
    - Workloads (requests)
    - Computing resources (machines)
  - Distributes client requests or network load efficiently across multiple servers [link]
    - E.g., service get popular, high load on service

→ horizontal scaling

- Why load balancing
  - Ensures high availability and reliability by sending requests only to servers that are online
  - Provides the flexibility to add or subtract servers as demand dictates







Caddy

- Configuration: dynamic
  - Static: Caddyfile
- One-liners:
  - Quick, local file server: caddy file-server
  - Reverse proxy: caddy reverse-proxy --from example.com --to localhost:9000

```
:7070
reverse_proxy 127.0.0.1:8081 127.0.0.1:8080 {
  unhealthy_status 5xx
  fail_duration 5s
}
```

- Open Source, software-based load balancer: https://github.com/caddyserver/caddy
  - “Caddy 2 is a powerful, enterprise-ready, open source web server with automatic HTTPS written in Go”
  - L7 load balancer
  - Reverse proxy
  - Static file server
  - HTTP/1.1, HTTP/2, and experimental HTTP/3
  - Caddy on docker hub

Dockerfile

- Example: caddy as LB, go as Service
  - docker-compose up --scale services=5

```
#docker-compose.yml
version: '3'
services:
  services:
    build: .
    ports:
      - "8080-8085:8080"
  lb:
    image: caddy
    ports:
      - "7070:7070"
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile
```

```
#Caddyfile
:7070
reverse_proxy * {
  to http://dsy-services-1:8080
  to http://dsy-services-2:8080
  to http://dsy-services-3:8080
  to http://dsy-services-4:8080
  to http://dsy-services-5:8080

  lb_policy round_robin
  lb_try_duration 1s
  lb_try_interval 100ms
  fail_duration 10s
  unhealthy_latency 1s
}
```

CORS

- CORS = Cross-Origin Resource Sharing
  - For security reasons, browsers restrict cross-origin HTTP requests initiated from scripts (among others)
  - Mechanism to instruct browsers that runs a resource from origin A to run resources from origin B
- Solution
  - Use reverse proxy with builtin webserver, e.g., nginx, or user reverse proxy with external webserver.

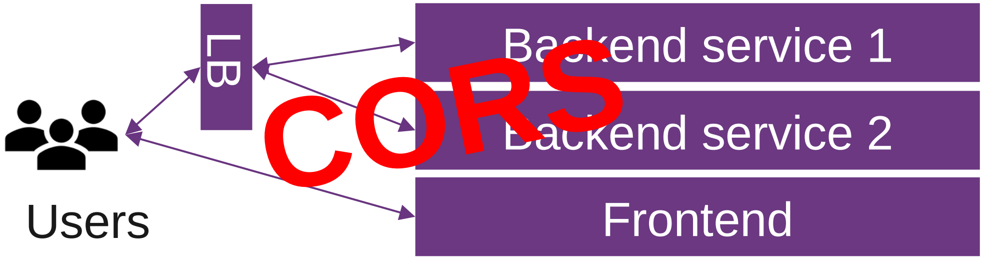
→ The client only sees the same origin for the API and the frontend assets

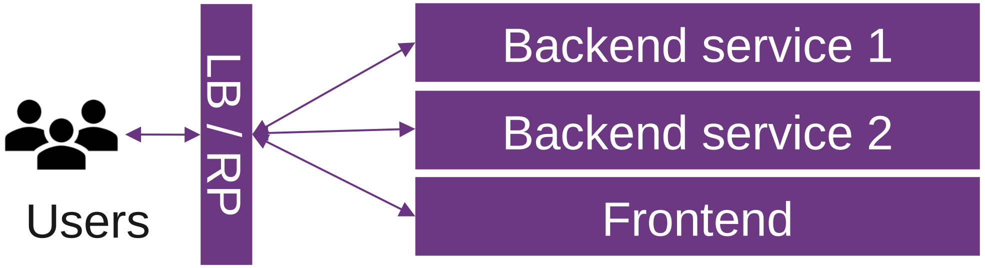
- Access-Control-Allow-Origin: https://foo.example

→ For dev: Access-Control-Allow-Origin: *

- w.Header().Set("Access-Control-Allow-Origin", "*")

- Reverse proxy



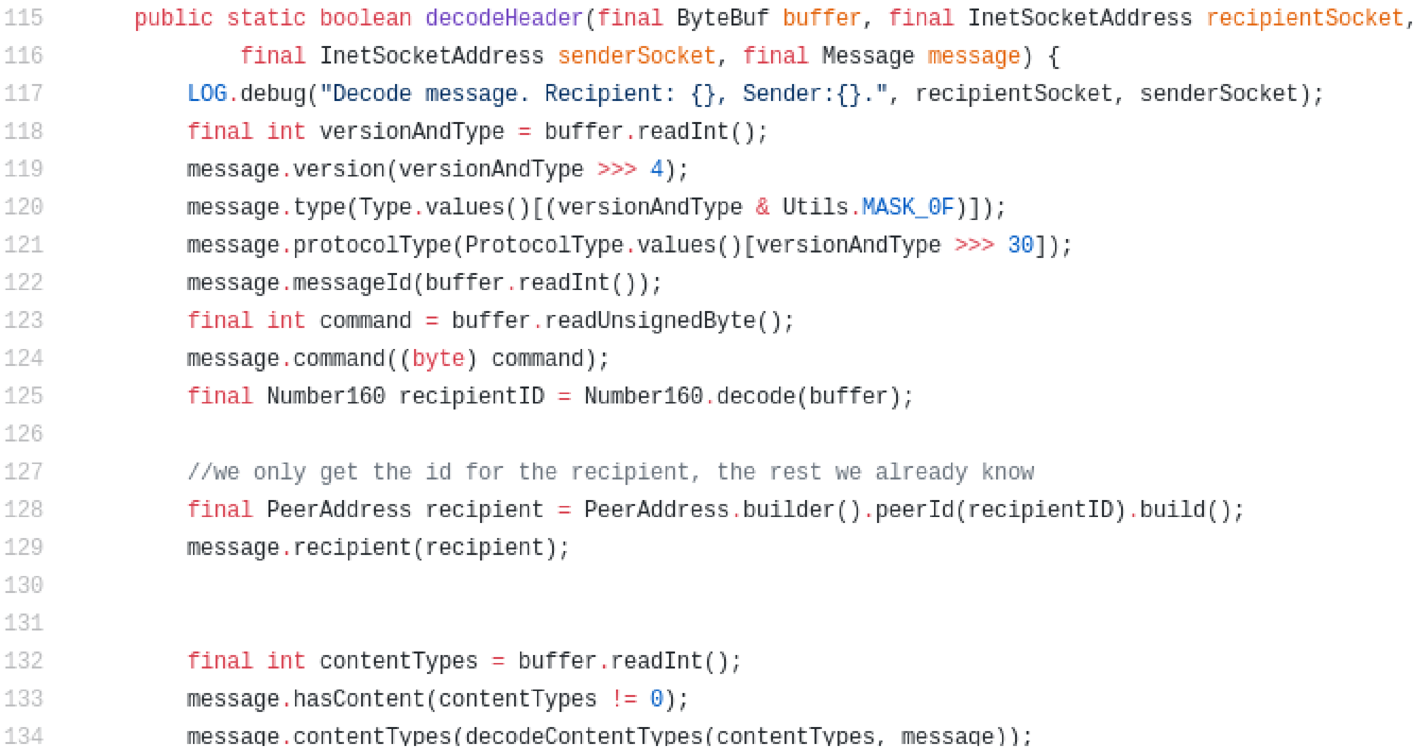


Lecture 7

Protocols

- Custom encoding/decoding
  - You control every aspect
  - You send more time on it

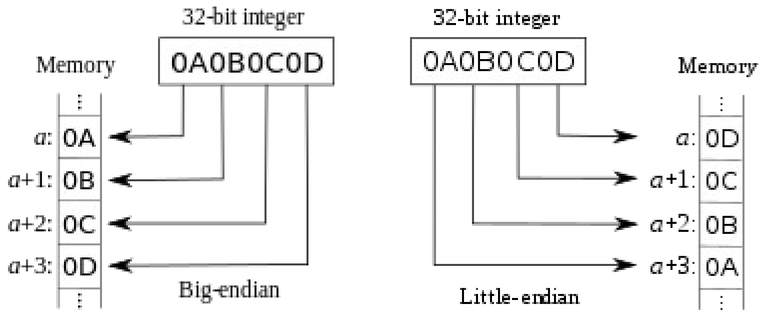
- Little-endian / Big-endian
  - sequential order where bytes are converted into numbers
- Networking, e.g. TCP headers: Big-endian
- Most CPUs e.g., x86: Little-endian, RISC-V: Bi-endianness



```
public static boolean decodeHeader(final ByteBuf buffer, final InetSocketAddress recipientSocket,
    final InetSocketAddress senderSocket, final Message message) {
  LOG.debug("Decode message. Recipient: {}, Sender:{}.", recipientSocket, senderSocket);
  final int versionAndType = buffer.readInt();
  message.version(versionAndType >>> 4);
  message.type(Type.values()[(versionAndType & Utils.MASK_0F)]);
  message.protocolType(ProtocolType.values()[versionAndType >>> 30]);
  message.messageId(buffer.readInt());
  final int command = buffer.readUnsignedByte();
  message.command((byte) command);
  final Number160 recipientID = Number160.decode(buffer);

  //we only get the id for the recipient, the rest we already know
  final PeerAddress recipient = PeerAddress.builder().peerId(recipientID).build();
  message.recipient(recipient);


  final int contentTypes = buffer.readInt();
  message.hasContent(contentTypes != 0);
  message.contentTypes(decodeContentTypes(contentTypes, message));
```



[source]

JSON example

- JSON + REST/HTTP
    - Human-readable text to transmit data
    - Often used for web apps
- 187 bytes

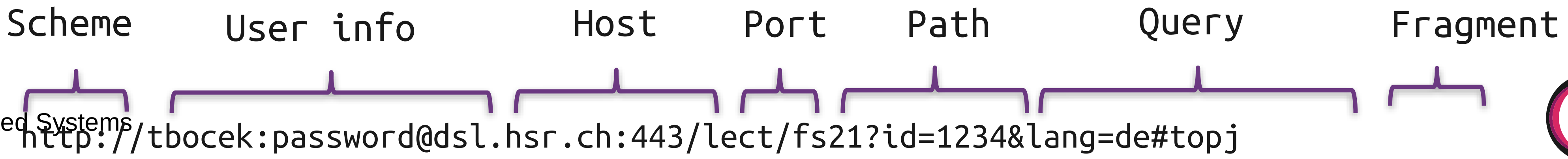
```
func main() {
  fmt.Println("Connecting...")
  req, _ := http.NewRequest("POST", "http://localhost:7000",
    strings.NewReader(`{"code": 5,"message": "Anybody there?"}`))
  req.Header.Set("Content-Type", "application/json")
  client := &http.Client{}
  resp, err := client.Do(req)
  if err != nil {
    panic(err)
  }
  defer resp.Body.Close()
  fmt.Printf("wrote request")
}
```

- Parsing overhead, JSON slower than binary protocol - benchmarks

```
[
    {
        "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“
    }, ...
]
```

Application Protocol: HTTP

- HTTP (HyperText Transfer Protocol): foundation of data communication for www
- Started in 1989 by Tim Berners-Lee
  - HTTP/1.1 published in 1997
  - HTTP/2 published in 2015
    - More efficient, header compression, multiplexing
  - HTTP/3 wip (April 2022: HTTP/3 protocol is an Internet Draft – not yet final)
- Request / response (resource)
- HTTP resources identified by URL
  - https://dsl.hsr.ch/design/hsr_logo.svg



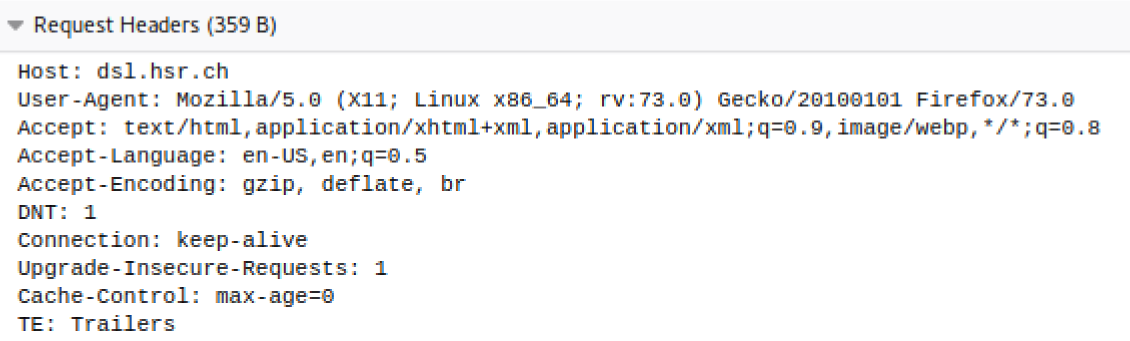
Scheme User info Host Port Path Query Fragment

http://tbocek:password@dsl.hsr.ch:443/lect/fs21?id=1234&lang=de#topj

- Text-based protocol

```
openssl s_client -connect dsl.hsr.ch:443
… TLS handshake …
GET /
```

- Browser sends a bit more…

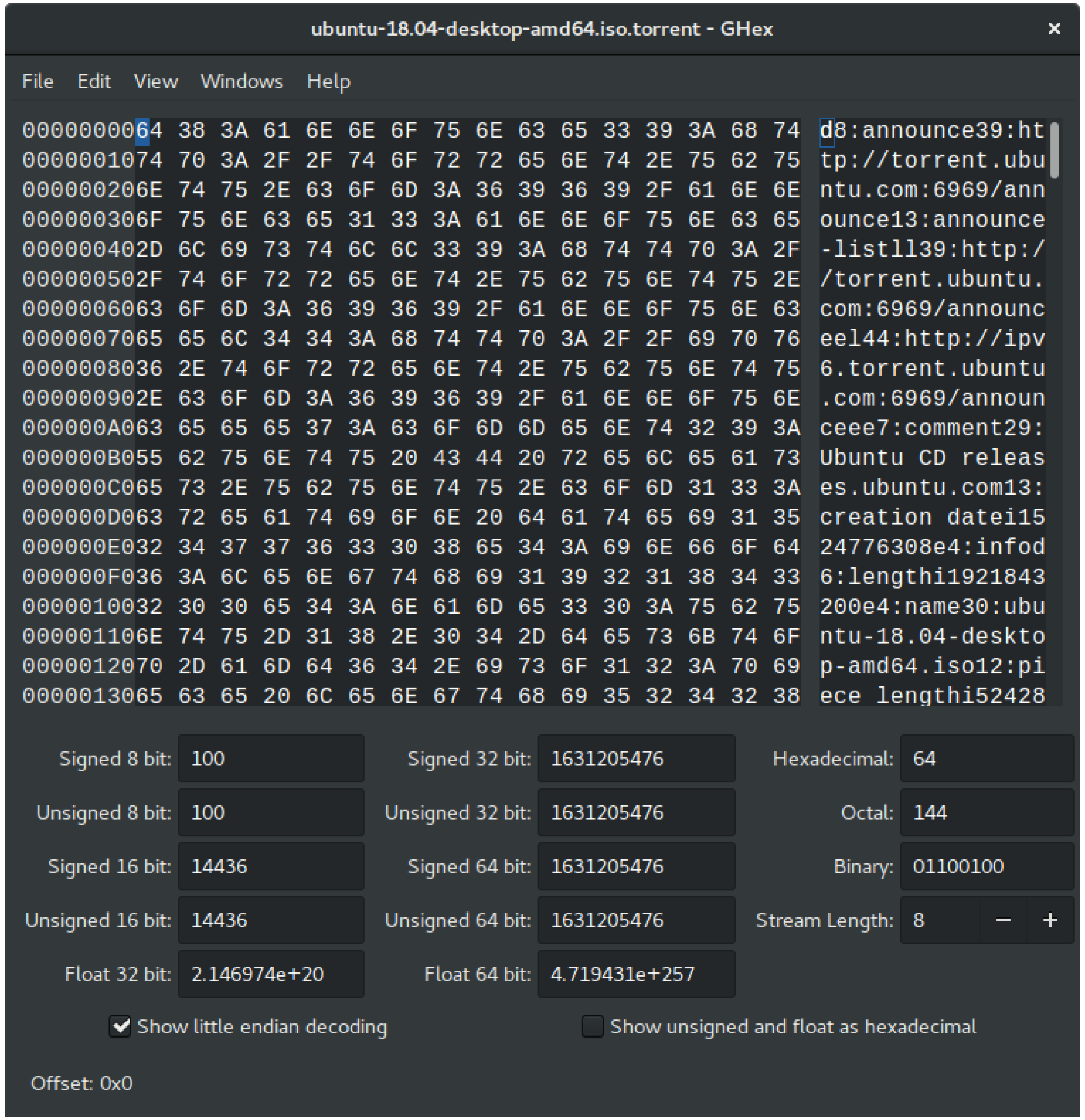


Request Headers (359 B)

```
Host: dsl.hsr.ch
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
TE: Trailers
```

Protocols Bencoding and Others

- Benconding
  - Integers: i42e, Byte string: 4:test, list: l4:testi42ee
  - Map/dictionary: d4:test3:hsr3:tomi42ee
- Use: BitTorrent
- UBJSON
- Cap’n Proto , FlatBuffers
  - Do not serialize, just copy, little-endian
- Apache Arrow
  - Do not serialize, copy, and optimally layout for memory access
- … and many others
- Benchmarks, benchmarks, …



```
0000000064 38 3A 61 6E 6E 6F 75 6E 63 65 33 39 3A 68 74  d8:announce39:ht
0000001074 70 3A 2F 2F 74 6F 72 72 65 6E 74 2E 75 62 75  tp://torrent.ubu
000000206E 74 75 2E 63 6F 6D 3A 36 39 36 39 2F 61 6E 6E  ntu.com:6969/ann
000000306F 75 6E 63 65 31 33 3A 61 6E 6E 6F 75 6E 63 65  ounce13:announce
000000402D 6C 69 73 74 6C 6C 33 39 3A 68 74 74 70 3A 2F  -listll39:http:/
000000502F 74 6F 72 72 65 6E 74 2E 75 62 75 6E 74 75 2E  /torrent.ubuntu.
0000006063 6F 6D 3A 36 39 36 39 2F 61 6E 6E 6F 75 6E 63  com:6969/announc
0000007065 65 6C 34 34 3A 68 74 74 70 3A 2F 2F 69 70 76  eel44:http://ipv
0000008036 2E 74 6F 72 72 65 6E 74 2E 75 62 75 6E 74 75  6.torrent.ubuntu
000000902E 63 6F 6D 3A 36 39 36 39 2F 61 6E 6E 6F 75 6E  .com:6969/announ
000000A063 65 65 65 37 3A 63 6F 6D 6D 65 6E 74 32 39 3A  ceee7:comment29:
000000B055 62 75 6E 74 75 20 43 44 20 72 65 6C 65 61 73  Ubuntu CD releas
000000C065 73 2E 75 62 75 6E 74 75 2E 63 6F 6D 31 33 3A  es.ubuntu.com13:
000000D063 72 65 61 74 69 6F 6E 20 64 61 74 65 69 31 35  creation datei15
000000E032 34 37 37 36 33 30 38 65 34 3A 69 6E 66 6F 64  24776308e4:infod
000000F036 3A 6C 65 6E 67 74 68 69 31 39 32 31 38 34 33  6:lengthi1921843
0000010032 30 30 65 34 3A 6E 61 6D 65 33 30 3A 75 62 75  200e4:name30:ubu
000001106E 74 75 2D 31 38 2E 30 34 2D 64 65 73 6B 74 6F  ntu-18.04-deskto
0000012070 2D 61 6D 64 36 34 2E 69 73 6F 31 32 3A 70 69  p-amd64.iso12:pi
0000013065 63 65 20 6C 65 6E 67 74 68 69 35 32 34 32 38  ece lengthi52428
```

Signed 8 bit: 100
Signed 32 bit: 1631205476
Hexadecimal: 64

Unsigned 8 bit: 100
Unsigned 32 bit: 1631205476
Octal: 144

Signed 16 bit: 14436
Signed 64 bit: 1631205476
Binary: 01100100

Unsigned 16 bit: 14436
Unsigned 64 bit: 1631205476
Stream Length: 8 − +

Float 32 bit: 2.146974e+20
Float 64 bit: 4.719431e+257

☑ Show little endian decoding
□ Show unsigned and float as hexadecimal

Offset: 0x0

Lecture 8

Consensus

- Definition: Consensus decision-making is a group decision-making process in which group members develop, and agree to **support a decision in the best interest of the whole**.
- A Byzantine fault is an arbitrary fault that occurs during the execution of an algorithm by a distributed system
  - Not only crash, but lie or even collude to reach an advantage
- **“Controlled” Distributed Systems:** your own nodes, your control, no collusion
- Find consensus
  - Paxos, Raft, vDHT, Zookeeper

- Often: consensus defines leader
  - Leader creates block
  - Leader adds data
  - Leader creates version
- How to find a leader?

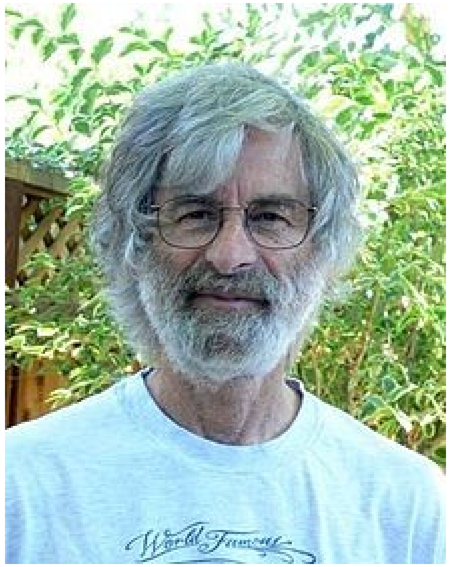
Arbitrary faults, but no collusion

... I HOPE MY MESSAGE WENT THROUGH ...
... WHEN TO ATTACK?
SYNCHRONISED ATTACK IS THE ONLY CHANGE TO WIN

source

Paxos History

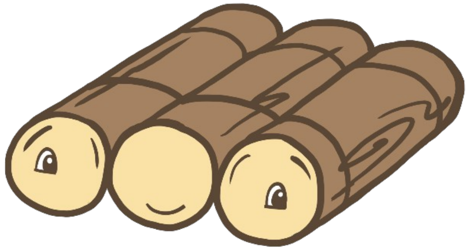
- Leslie Lamport discovered the Paxos algorithm in late 1980s
  - Attempt to prove that there was no such algorithm which can tolerate the failure of any number of its processes
  - Until he realized that he created working protocol


source

- Wrote paper and submitted it to Transactions on Computer Systems (TOCS) in 1990
  - Reviewer: was mildly interesting, but needs significant improvement
  - Leslie Lamport: “so I did nothing with the paper”
- People started to using Paxos to solve problems in distributed systems
- Resubmitted in 1998 to TOCS
  - Accepted without any major changes
- Paxos paper won an ACM SIGOPS Hall of Fame Award in 2012
- Received Turing award in 2013 , also due to Paxos
  - “Turing Award is generally recognized as the highest distinction in computer science and the "Nobel Prize of computing”“ [link]

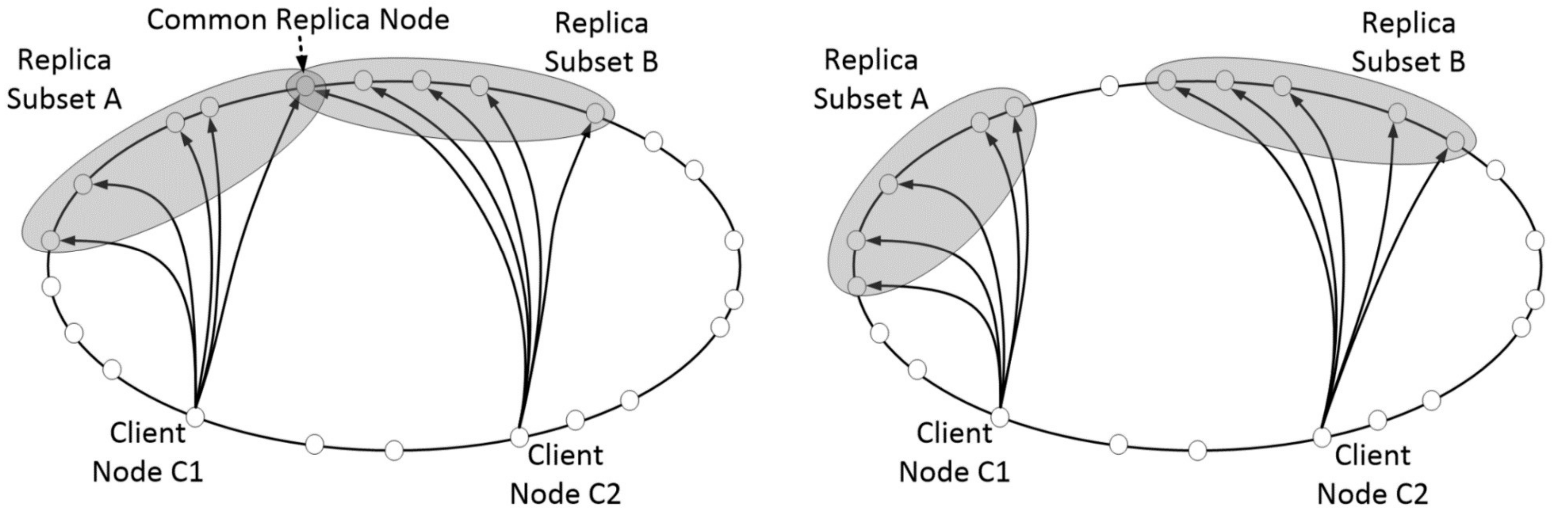
Raft (multi paxos)

- “this makes Raft more understandable than Paxos and also provides a better foundation for building practical systems.” [link]
- RAFT: Reliable, Replicated, Redundant, And Fault-Tolerant
- Follower, Candidate, Leader [link]
  - Raft implements leadership election,
  - Once a leader has been elected, all decision-making within the protocol will then be driven only by the leader
  - Only one leader can exist at a single time
- Each follower has a timeout (typically between 150 and 300 ms) in which it expects the heartbeat from the leader.
  - The system is only available when a leader has been elected and is alive
  - Otherwise, a new leader will be elected and the system will remain unavailable for the duration of the vote
  - Starts election by increasing term counter, voting for itself, and sending a message to all other servers requesting their vote
  - If a higher term is received, become follower, if not, leader



Consistency

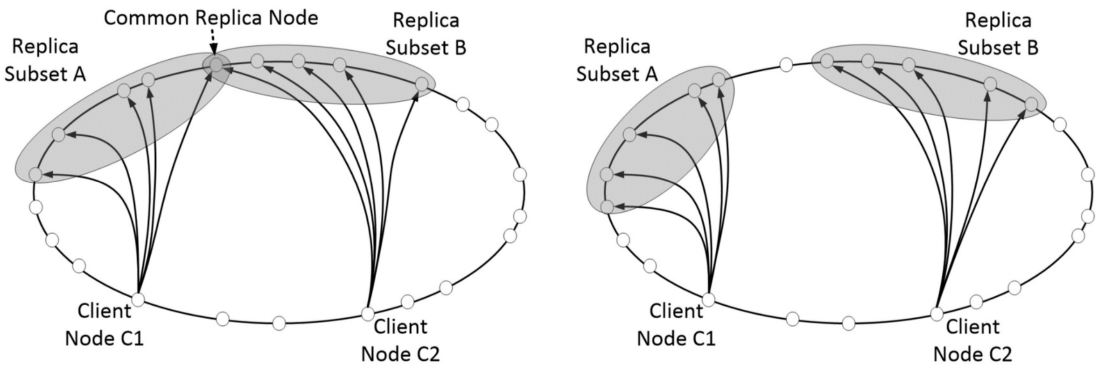
- Consistency in DHTs – vDHT, similarities to Paxos
  - Number = versions, for doing updates
  - Simplified roles (peer)
  - No leader election, works well with churn (not heavy churn)
- CoW, software transactional memory (STM) → for consistent updates. Works for light churn



Lecture 9

CRDT

- (Paxos, why take over larger number?)
  - “acceptors made a promise that no other proposal with a smaller number can make it to consensus” → If acceptor accepted, but its not majority → could stall forever, thus take over large number (link, link)
- L08S10: vDHT
  - A way how to bring consistency to DHTs
  - ~CRDT (operation-based CRDTs)
  - Conflict-free replicated data type (CRDT)
  - ~git but with no merge conflicts
- CRDT must be
  - Commutative x ● y = y ● x
  - Associative (x ● y) ● z = x ● (y ● z)
  - Idempotent x ● x = x



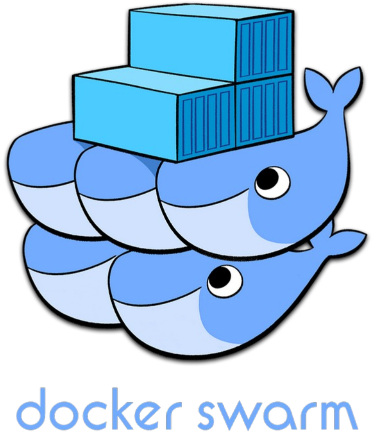
- CRDT Counter (G-Counter)
  - For each machine 1 array position for counter

A	B	C
A:6 B:0 C:0	A:0 B:3 C:0	A:0 B:0 C:9

  - Merge: max of each counter (A:6 B:3 C:9)
    - Old data A:6 B:2 C:9 merge-max / A:5 B:3 C:2 A:6 B:3 C:9
    - Commutative, associative, idempotent

Docker Swarm

- Use docker --context to run/maintain containers on other machines
  - Does not work for docker-compose, could be used with Ansible… “Ansible is also great for bootstrapping Docker itself” [source]
- Docker Swarm
  - Deploy with docker-compose.yml (deploy:)
  - Built into docker
    - docker swarm – manage swarm
    - docker node – manage nodes
  - Scheduler is responsible for placement of containers to nodes
  - Can use the same files, easy to setup?
    - Azure, Google cloud, Amazon



- Kubernetes vs. Docker Swarm
- “Docker Swarm has already lost the battle against Kubernetes for supremacy in the container orchestration space” [link]
- “Kubernetes supports higher demands with more complexity while Docker Swarm offers a simple solution that is quick to get started with.” [link]

Kubernetes

- Kubernetes, K8s
  - Container orchestration (docker)
    - Automated deployment, scaling
  - Started by Google, now with CNCF
- Kubernetes-based PaaS
  - Google, Amazon, Azure (book), Digital Ocean, …
    - Difficult pricing schemes

  - 1.0 released in 2015
  - Package manager Helm released in 2016 (convert docker-compose)
- Why Kubernetes?
  - Containers can crash, machine that runs container can crash (e.g., out of memory)
  - Development: run on one machine, deployment how and where to distribute?
  - Kubernetes manages the lifecycle of containers