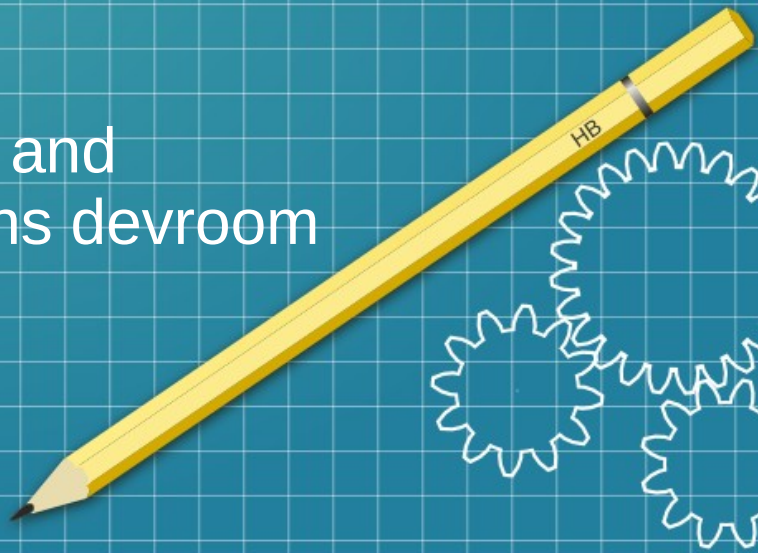
Designing a Human Centric Next Generation Internet

FOSDEM ‘21
Collaborative Information and
Content Management Applications devroom



Introduction

- Jens Finkhäuser (Finkhaeuser)
- 30+ years since “hello, world” (C64)
- 20+ years since paid ~~“hello, world”~~ real software
- Dial-up modems, 1200 Baud (still hear the sound)
- BBSes before Internet/Web
- PHP → Python → C/C++
- Fullstack → Backend/Libraries → Network protocols

Introduction

- From 2006: Video on Demand streaming over P2P network at Joost
  - Netflix started VoD 2007
- March 2008: Live streaming “March Madness” college basketball over P2P network
  - DLive “world’s first decentralized live streaming protocol” announced at BitTorrentX 2020-12-21
    https://medium.com/@BitTorrent/launch-of-new-dlive-protocol-announced-at-bittorrentx-product-release-conference-fb8b7d9f5308

Introduction

- early 2019: worked with Blockchain startup on (live) video over decentralized networks
  - too much Blockchain focus
- Looked for funding
- Grant from NLNet confirmed Dec. 2019
- 2020 pandemic, lockdown :(
- Mid-2020: joined AnyWi Technologies B.V. :)

Funding

https://nlnet.nl/

https://www.ngi.eu/ngi-projects/ngi-zero/

Related Work

AnyWi TECHNOLOGIES

https://www.anywi.com/

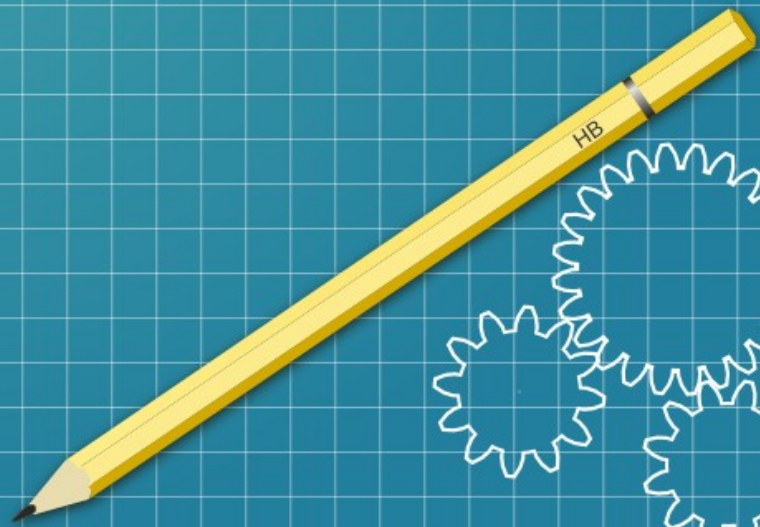
COMP4DRONES

https://comp4drones.eu/

ADACORSA

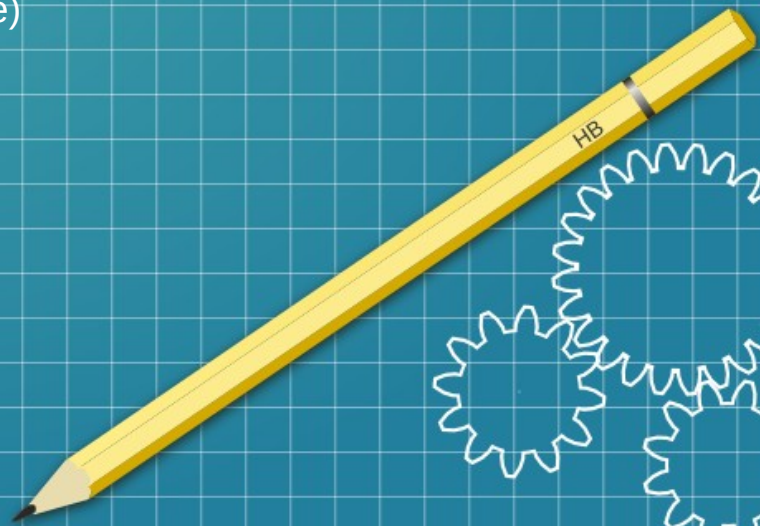
https://adacorsa.eu/

The Web is dead.

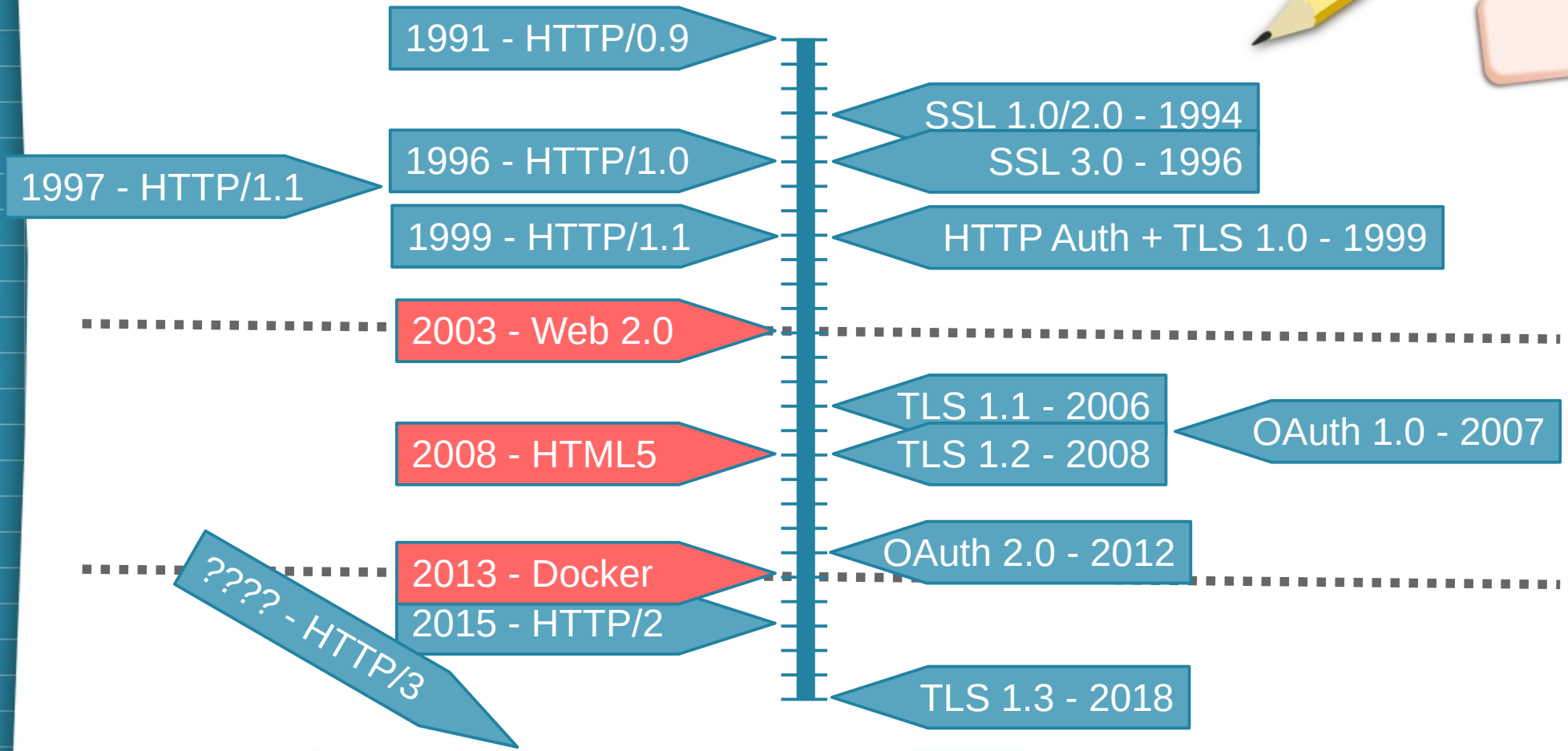


The Web is dead.

(it will stay undead for quite some time)



Web Timeline



Web Eras

- From 1991 Early Web: Simplicity
  - Establish HTTP as a decentralized protocol
  - Add some basic security features
- ca. 2003 Web 2.0: Get everyone online
  - Pull users to centralized providers instead of everyone running their own
  - Authentication/authorization more important
  - 2006: start HTML5, “Working Draft” in 2008
- ca. 2013 Docker: containerization, microservices, RESTish APIs
  - Feedback loop: consolidation → scale → complexity → consolidation

Web Protocol Issues

- Security has always been an afterthought
- Privacy never even a topic
- *Commercial* interests, not *user* interests determine innovation areas
- HTTP/2 effectively Google’s SPDY
- HTTP/3 effectively Google’s QUIC

Web Protocol Issues

- HTTP is and isn't CRUD (Create, Read, Update, Destroy)
  - Conceptually a resource store access protocol/methods map to similar functionality
  - POST and PATCH have media type/resource specific interpretation
- GET Range headers not always supported, PATCH under-specified → inefficient at finer resolution than "entire resource"
- Unlike all (?) prior CRUD systems, HTTP forgoes resource ownership and access rights
  - Left to server implementation/application aka unspecified
- Must lead to server "owning" data

HTML Issues

- XHMTL 2.0 vs. HTML5 start in 2006
- “Living Standard” since 2012 with W3C/WHATWG split
- WebKit as best funded engine standards driver for a long time
- 2020: Mozilla abandons own engine
- Google de facto owns HTML5 via WebKit
  - React? Typescript?

HTML Issues

- HTML historically mixes data and representation.
  - XHTML tried to fix, HTML5 “fixes” by leaving custom tag semantics unspecified
- With JavaScript, also mixes processing
  - What about MVC (MVP, MVVM)?
- Worse: separation possible, but linking/embedding CSS+JS is simpler
- Must lead to “data silos”, only accessible via linked View/Controller

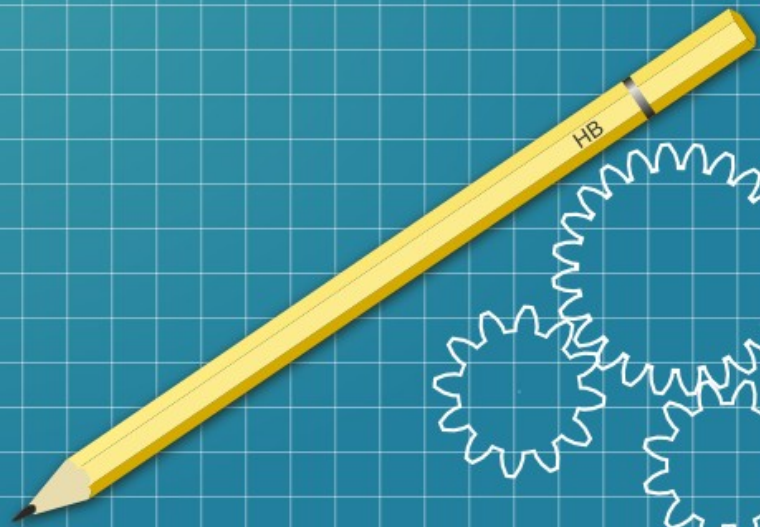
Strengths

- HTTP
  - Early simplicity (though long term weakness)
  - Server-side processing (though also weakness)
    - Merges API and resource store concepts
- HTML
  - Simplicity with getting started (though long term weakness)

Strengths

- HTTP
  - Early simplicity (though long term weakness)
  - Server-side processing (though also weakness)
    - Merges API and resource store concepts
- HTML
  - Simplicity with getting started (though long term weakness)

So: Whither Web?



Censorship

Eric Weinstein
@EricRWeinstein

Replying to @EricRWeinstein

Level 0: This is an attack on free speech in the public square.

Level 1 rejection: This is not the public square but a matter of for profit companies and the freedom to do what they please.

Level 2: Level 1 only works if there is a public sector of the internet for all to use.

3:53 am · 10 Jan 2021 · Twitter for iPhone

Centralization as Default

moxie0 commented on May 6, 2016

> I don't think that the several LibreSignal users are a big impact on the server. Anyway, since you rely on donations, what is the difference for you if the user is using Signal-Android or another Signal fork?

The difference is huge; one we have control over, the other we don't. I understand that federation and defined protocols that third parties can develop clients for are great and important ideas, but unfortunately they no longer have a place in the modern world. Even less of a place for an organization the size of ours. Everyone outside the FOSS community seems to know it, but it took actually building the service for me to come to the same understanding, so I don't expect you to believe me.

Not my highlight

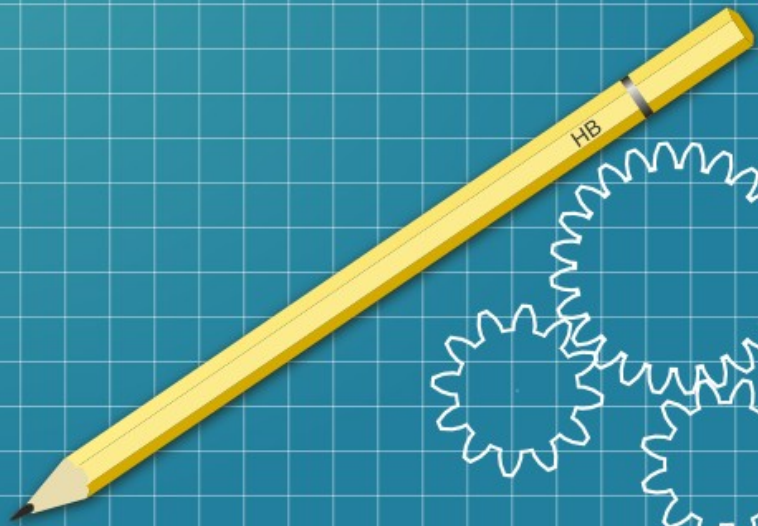
Yielding Control as Default

The internet interprets censorship as damage and routes around it.

– John Gilmour

My ~~TED~~FOSDEM talk

Collaboration



Mechanics of Collaboration

- Communication
- Sharing stuff
  - Give, receive digital assets
  - More “trade” than social media
- Sharing skills (aka working together on stuff)
  - Update parts of a resource
  - Selling services
- This is “human centric”, humans collaborate

Mechanics of Collaboration

- Communication
- Sharing stuff
  - Give, receive digital assets
  - More “trade” than social media
- Sharing skills (aka working together on stuff)
  - Update parts of a resource
  - Selling services

Real-time

Access control, ownership

Finer resolution than entire resource

Server-side processing?

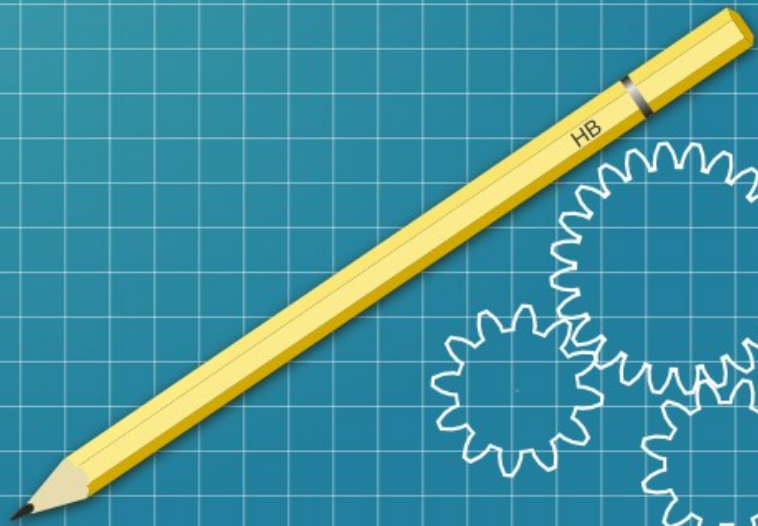
What is Real-Time?

- Consume data while it is in the process of being produced
- Produce, consume in chunks
  - Finer resolution than “entire resource”
- Indeterminate size of data (at outset)
- Data streaming
  - Video streaming makes a great use-case: it’s high bandwidth, and has low latency requirements.

Requirements of Collaboration

Requirement	Solution
Data streaming	*Something other than HTTP*
Access control	E2E Encryption (shared symmetric key)
Ownership	Encryption (private key)
Server-side processing	Remote APIs
No man in the middle required/wanted	*Something other than HTTP*

Data Locality & Devices



Data Locality

- Web is client/server
  - Data resides at central location
  - Allows for multiple remote clients
- Multiple personal devices, each with own storage
- IoT/Smart Sensors
  - Data is collected and resides (temporarily?) in hundreds of locations
  - IoT connectivity (BLE, LoRa) may not make IoT device ideally suitable for “client” role

Data Locality & Devices

- Web is client/server
  - Data resides at central location
  - Allows for multiple remote clients
- Multiple personal devices
  - Multiple link technologies (WiFi, LTE)
  - each with own storage
- IoT/Smart Sensors
  - Data is collected and resides (temporarily?) in hundreds of locations
  - IoT connectivity (BLE, LoRa) may not make IoT device ideally suitable for “client” role

Smooth handover

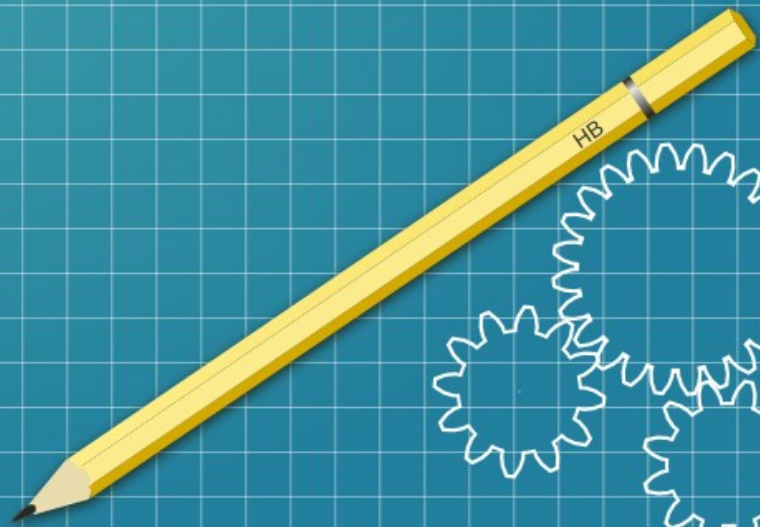
Selective synchronization

Heterogeneous link technologies, no strict client/server

Requirements of Collaboration

Requirement	Solution
Data streaming	*Something other than HTTP*
Access control	E2E Encryption (shared symmetric key)
Ownership	Encryption (private key)
Server-side processing	Remote APIs
No man in the middle required/wanted	*Something other than HTTP*
Smooth handover	Multi-home/-path/-link
Selective synchronization	PubSub of resources
Heterogeneous link technologies	Overlay network
No strict client/server	P2P

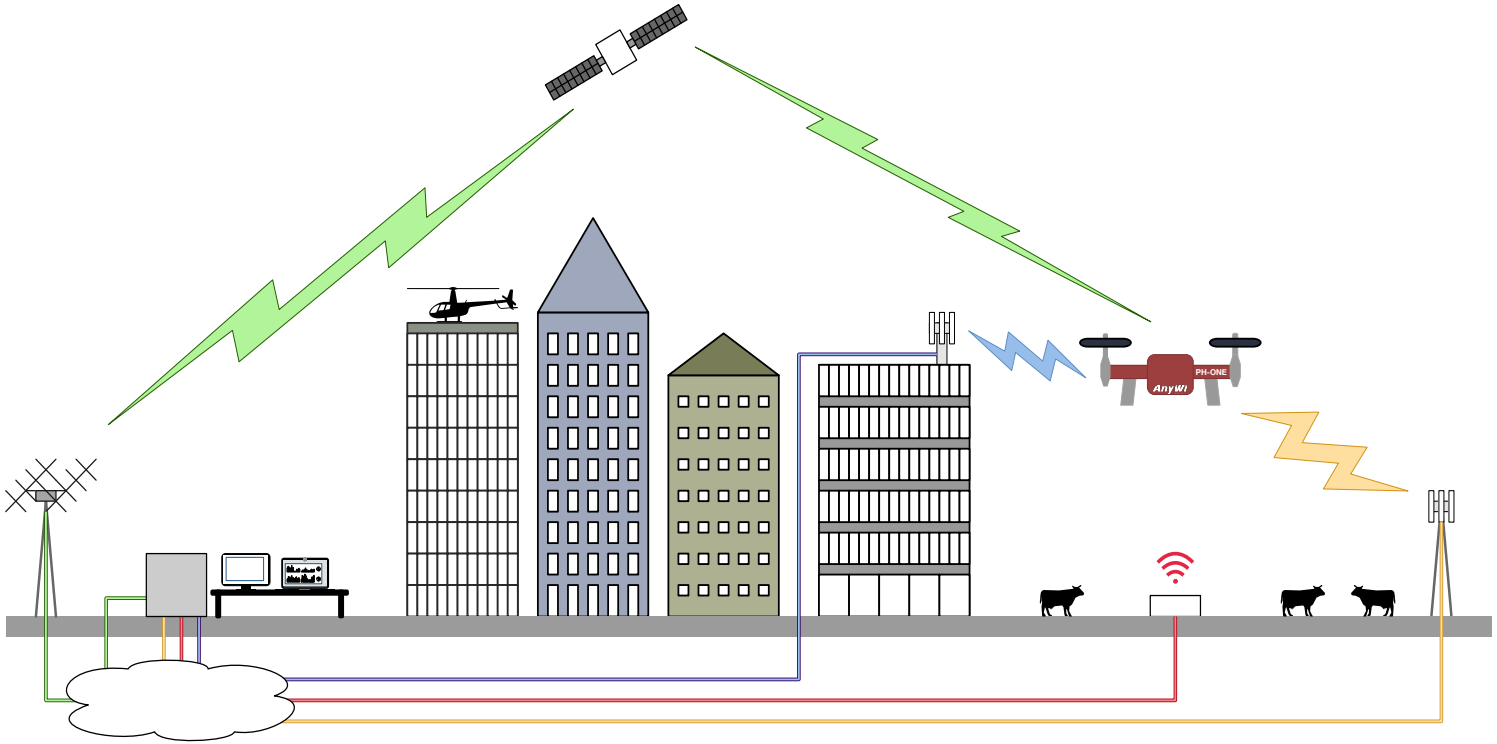
Drones



BVLOS

- Beyond Visual Line of Sight
  - aka no toys
  - also non-military (with “unlimited” budget)
- EASA requirements: reliable Command, Control & Communications (C^3) links
- Reliable C3 Links for UAS (paper preprint)

C³ Link Handover

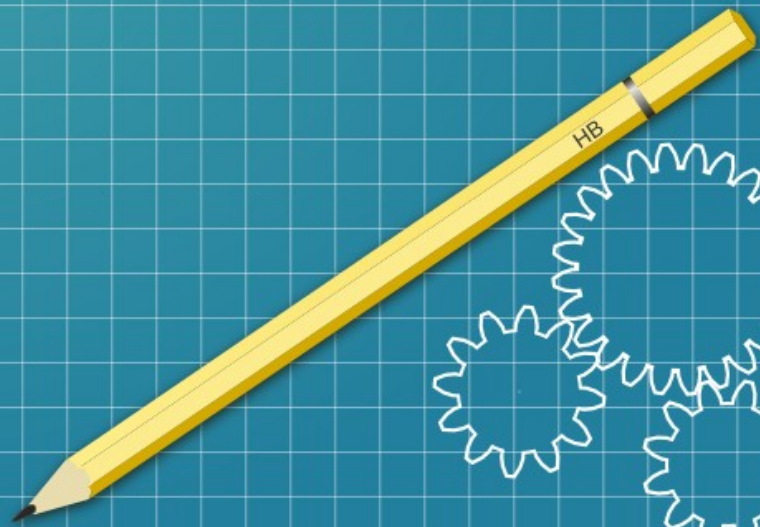


Requirements of Reliable C³ Links

Requirement	Solution
Data streaming	*Something other than HTTP*
~~Access control~~ Tamper-proofing	E2E Encryption (shared symmetric key)
~~Ownership~~ Identification	Encryption (private key)
~~Server-side processing~~	~~Remote APIs~~
No man in the middle required/wanted	*Something other than HTTP*
Smooth handover	Multi-home/-path/-link
~~Selective synchronization~~	~~PubSub of resources~~
Heterogeneous link technologies	Overlay network
No strict client/server	P2P

Vision

(quick summary)



Requirements of Collaboration

Requirement	Solution
Data streaming	*Something other than HTTP*
Access control	E2E Encryption (shared symmetric key)
Ownership	Encryption (private key)
Server-side processing	Remote APIs
No man in the middle required/wanted	*Something other than HTTP*
Smooth handover	Multi-home/-path/-link
Selective synchronization	PubSub of resources
Heterogeneous link technologies	Overlay network
No strict client/server	P2P
Ease of adoption	Simplicity in usage

Human Centric

- “Our” data will live on many devices
  - But be safe from malicious access
- **We can** access our data anytime, in part (fast) or in full (potentially slow, requires full sync)
- **We can** access our data from any device, including not our own (requires private key)
- **We can** share and collaborate on our data
- **We can** selectively allow access to our data from processing nodes
  - e.g. devices such as Printers

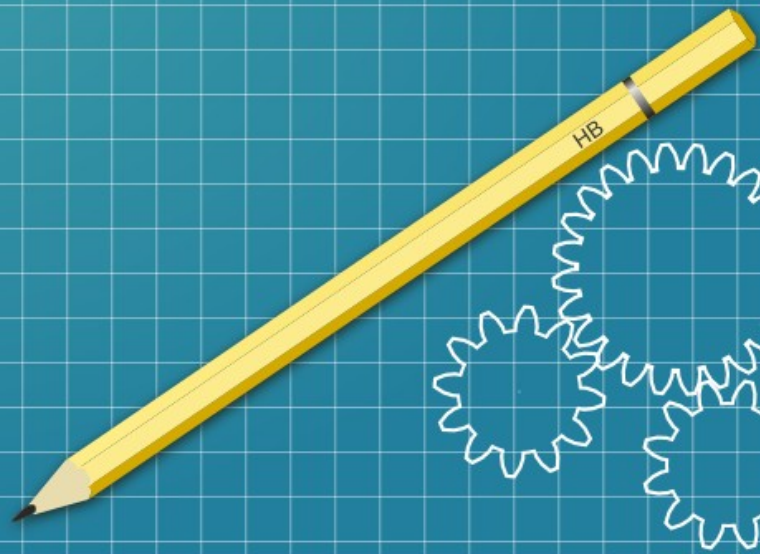
Infrastructure

- Web’s focus on “Server” functionality aka application pushes web solutions to the “application” level of the OSI stack
- Internet != web
  - Internet is infrastructure, ubiquitous, doesn’t care about specific applications
- Need **infrastructure protocols**, not more **application frameworks**

Peer-to-peer

- **Peer as in host:** no distinct client or server role → shifts responsibility to host owner
- **Peer as in person:** host-oriented is foundation for person-oriented (human centric) networking

Progress



Liberate

- https://gitlab.com/interpeer/liberate/
- “small platform liberation library” → minimal platform abstractions
- C++
- Linux, FreeBSD (other BSD?), OS X, Windows, Android, (iOS?)

Packeteer

- https://gitlab.com/interpeer/packeteer/
- simplified, asynchronous, event-based socket API.
- C++
- Linux, FreeBSD (other BSD?), OS X, Windows, Android, (iOS?)
- Focus different from other projects: Cross platform (highest priority), **low usage complexity**, stable API/ABI, **packet oriented I/O friendly**, scalable, efficient, extensible (lowest priority)

Packeteer

- Windows port works, but has bugs
  - Probably wants a partial re-write
- Some refactoring of POSIX code required
- No scatter/gather I/O yet
- Some tweaks, extensions
- Semi-modern C++ (focus on library != templates)

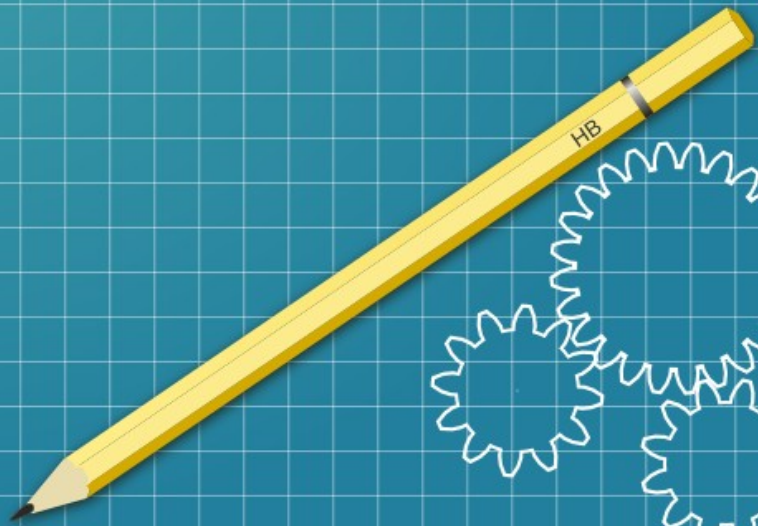
Channeler

- Highly WIP
- Multi-channel, packet-oriented protocol
  - Channels as in HTTP/3, should allow for efficiency multiplexing various streams
- Ready for protocol extensions such as encryption
  - Encryption based on WireGuard/NOISE, with extra handshake
- Provisions for multi-link
- Future: channel-specific tuning of reliability
  - FEC, resends, SCTP-like reliability without strict ordering
- UDP-based, but could be on IP or Ethernet

WIP: Multi-Path Tunnelling Protocol

- Part of ADACORSA
- Abstract protocol, focus on messages and state machines
- Paper being written alongside, hopefully published in mid-2021
- Multi-link in Channeler will follow this pattern

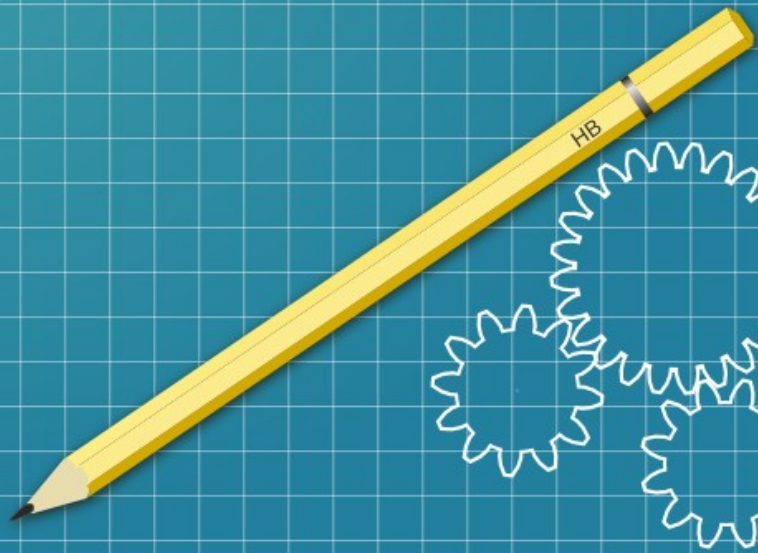
Future



Future

- Previous should provide for stable P2P links
- Need a distributed hash table
  - Kademlia is good, but needs tweaks to exploit network infrastructure better
- Need a streaming protocol over stable links, something akin to PPSPP https://tools.ietf.org/html/rfc7574
- Distributed filesystem(-like) as best API for simple, safe, distributed applications?

Contributing



Contributing

- Design identity & guides
- Website & documentation templates
- Any protocol feedback! Any feedback!
- *Relatively* financed for the next few years
  - Not set up for donations yet, but intend to
  - Would be good to hire a few developers & staff
  - Fundraising experience? Yes please!

Questions?

https://interpeer.io
jens@interpeer.io
https://reset.substack.com/