On the Far Side of REST

An Architecture for a Future Internet

FOSDEM ‘22
Real Time Communications devroom

Funding

Internet Society Foundation

https://www.isocfoundation.org/

nlnet FOUNDATION

https://nlnet.nl/

NGI ZERO

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

European Commission

https://ec.europa.eu/

Funding Topics

- NGI0 (NLNet; EC):
  - Transport and lower-level protocols with focus on streaming capabilities
- ISOC Foundation:
  - Higher-level protocols; (mostly) the topic of this presentation

Organisational

- Public interest company (waiting for tax office)
- Why a “public interest company”?
  - Lightweight, 1-person, limited liability company.
  - “Public interest” defined (in Germany) as e.g. research, with the provision that results **must** be made usable by the public (e.g. GPL).
  - May write donation receipts.
  - Can employ people (yaay!)
  - Opens access to industrial R&D funds.

Organisational

- First (part-time) hire: Adrian Cochrane
  - https://www.openwork.nz/ (Work)
  - https://adrian.geek.nz/ (Personal/F(L)OSS)
  - @alcinnz@floss.social (Fediverse)
- Help from Autonomic Co-Operative
  - https://autonomic.zone/
  - Specifically, Rebecca Bulboacă
    - https://hazelnot.xyz/ (Portfolio)
    - @hazelnot@sunbeam.city (Fediverse)

Human Rights in the Digital Realm

Digital Human Rights

- Universal Declaration of Human Rights does not stop on the Internet.
- European Commissions Next Generation Internet Initiative talks about an Internet of Humans.
- Constant battle in e.g. weakening/strengthening encryption:
  - Weaken: help prevent crime, help investigate it.
  - Strengthen: help prevent crime, protect from abuse.
  - Global Encryption Coalition: https://www.globalencryption.org/

UDHR Examples (IANAL)

- Article 12: “No one shall be subjected to *arbitrary interference* with his **privacy**, family, home or **correspondence**, (...)”
  - Right to (end-to-end) encryption?
- Article 17.2: “No one shall be *arbitrarily deprived* of his **property**.”
  - Removal of paid-for content from X-as-a-Service? (Definition of property matters!)
- Article 5: “No one shall be subjected to torture or to cruel, inhuman or **degrading** treatment or **punishment**.”
  - UX dark patterns such as cookie banners?
  - Does this imply a right to accessibility?

Engineering Choices

- “No politics in X” always implicitly supports the status quo.
- The status quo includes e.g. surveillance capitalism, voter manipulation, doxxing and death threats (also against FOSS developers), racist/biased AI due to racist/biased training sets, etc., etc.
- Do my choices help protect human rights? Do they hinder them? Or do they (really) have no impact?

REST Architecture

REST

- Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000. https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
  - Paper compares different architectures, excellent read!
- **IS NOT** HTTP (but inspired by work on HTTP/1.1)
- **IS NOT** software, framework, CRUD-style use of HTTP methods, etc.
  - *RESTful* has little to do with *REST*.
- **IS** a ~~decentralized~~, scalable architecture.

REST Diagram

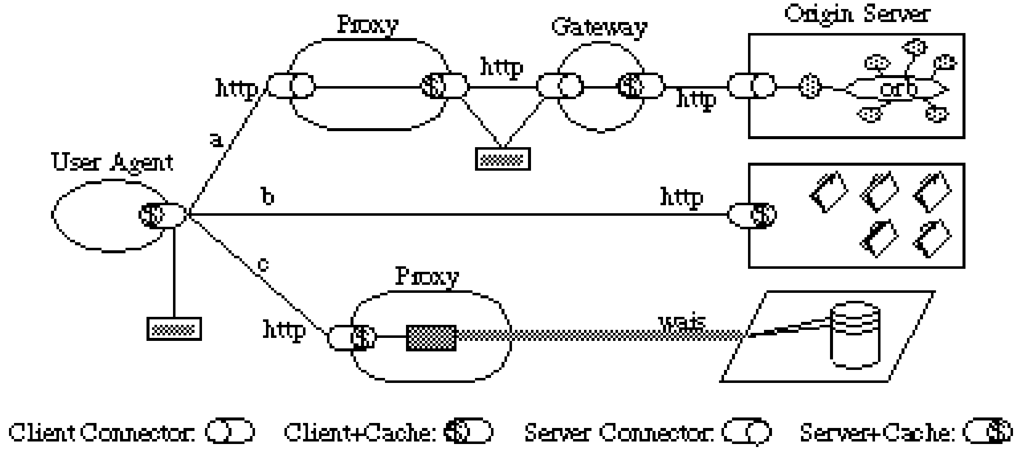


Figure 5-10. Process View of a REST-based Architecture

A user agent is portrayed in the midst of three parallel interactions: a, b, and c. The interactions were not satisfied by the user agent's client connector cache, so each request has been routed to the resource origin according to the properties of each resource identifier and the configuration of the client connector. Request (a) has been sent to a local proxy, which in turn accesses a caching gateway found by DNS lookup, which forwards the request on to be satisfied by an origin server whose internal resources are defined by an encapsulated object request broker architecture. Request (b) is sent directly to an origin server, which is able to satisfy the request from its own cache. Request (c) is sent to a proxy that is capable of directly accessing WAIS, an information service that is separate from the Web architecture, and translating the WAIS response into a format recognized by the generic connector interface. Each component is only aware of the interaction with their own client or server connectors; the overall process topology is an artifact of our view.

REST: Architectural Properties #1

- Performance:
  - Network Performance: throughput, overhead
  - User-Perceived Performance: latency, completion time
  - Network Efficiency: best application performance is obtained *by not using the network*
- Scalability: refers to the ability of the architecture to support large numbers of components or interactions
- Simplicity

REST: Architectural Properties #2

- Modifiability:
  - Evolvability: change implementation without affecting system
  - Extensibility: add functionality safely
  - Customizability: client-initiated server behaviour (e.g. content type)
  - Reusability: of components
- Visibility: ability to monitor or mediate
- Portability: move code along with data (see later slides)
- Reliability: resilience to (partial) failure of components

REST evolution (in HTTP)

- *Commercial* interests, not *user* interests determine innovation areas
- HTTP/2 effectively Google’s SPDY
- HTTP/3 effectively Google’s QUIC
- **NO** architectural changes
- **NO** security (added on via TLS)
- **NO** privacy (never on the radar)

REST (HTTP) Problems #1

- There are at least three different use-cases nowadays:
  - “Document web”: the original use-case of providing documents for public dissemination. Good fit.
  - “API web”: treat “documents” as structured data in response to equally structured data requests.
    - Problem: REST (and HTTP) do not contain provisions for server-initiated communications. Limited fit.
  - “Streaming web”: media streaming (and other real-time applications) may perform best over lossy medium, but HTTP requires loss protection (REST ignores it). Bad fit.

REST (HTTP) Problems #2

- Authentication and authorization are (server) component defined
  - Reduces reliability and modifiability
- Authorization implementations are often stateful
  - Server statelessness is the “state transfer” in REST
  - Statefulness reduces reliability and modifiability, and may reduce performance
- REpresentational nature makes (optional) code transfer mandatory:
  - How else to implement a generic client for a server-defined protocol?
  - Reduces portability by requiring fatter, more complex clients
  - Eliminates decentralization: data + code under server control

REST (HTTP) Problems #3

- Separation of “client” and “server” roles does not make much sense when synchronizing data between “client” devices.
  - Does make sense from a “who provides and who uses service?” perspective.
  - Multiple devices:
    - Do I “push” changes to other devices?
    - Do other devices “pull” changes to themselves?
    - Who coordinates what to push and/or pull?
    - REST answer is to always require a centralized synchronization service.

Future Key Properties & Constraints

in support of Human Rights

Future Key Properties #1

- Other properties **shall** encourage distribution, **must not** require centralization.
- Baran, Paul (1964) Memorandum RM-3420-PR
- Resilience in comm. networks.

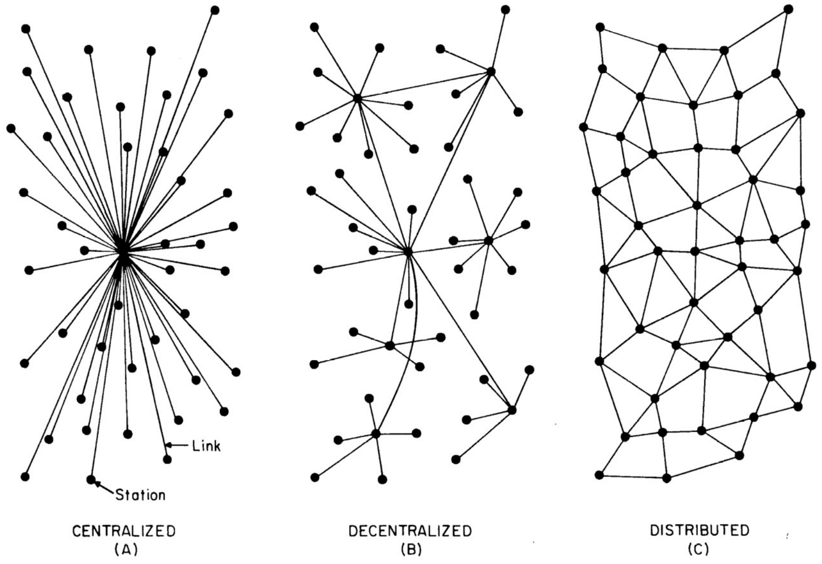


FIG. I — Centralized, Decentralized and Distributed Networks

Baran, P. (1964). On Distributed Communications, Memorandum RM-3420-PR. https://www.rand.org/content/dam/rand/pubs/research_memoranda/2006/RM3420.pdf

Future Key Properties #2

- Authenticated and anonymous uses supported.
- Authorized and public uses supported.
- Private by default.
- Real-time (streaming) and on-demand uses supported.
- Bi-directional uses supported (“client” or “server” initiated)
- Reachability: shall not require data link/transport properties that exclude device classes.
- Genericity: application logic, data formats and data transmission must be separate concerns.

Additional/Modified Constraints #1

- Encrypted transport induces privacy and authenticated uses.
  - Anonymous uses supported via authentication levels and client control: “I know I’ve spoken to you before” vs. “I know where you live”
  - Additional constraint on REST’s connectors.
- Distributed authorization induces authorized uses, optional authorization induces public uses.
  - Not covered by REST.
- Real-time/lossy capable transport induces real-time use cases.
  - Additional constraint on REST’s connectors.

Additional/Modified Constraints #2

- Publish-subscribe support induces server-initiated communications.
  - Weakens REST’s client-server constraint.
  - Weakens REST’s statelessness constraint.
  - Introduces need for different “verbs” (methods) compared to HTTP.
- Cache constraint remains.
- Uniform interface remains (but see comment on verbs above).
- Layered system remains.

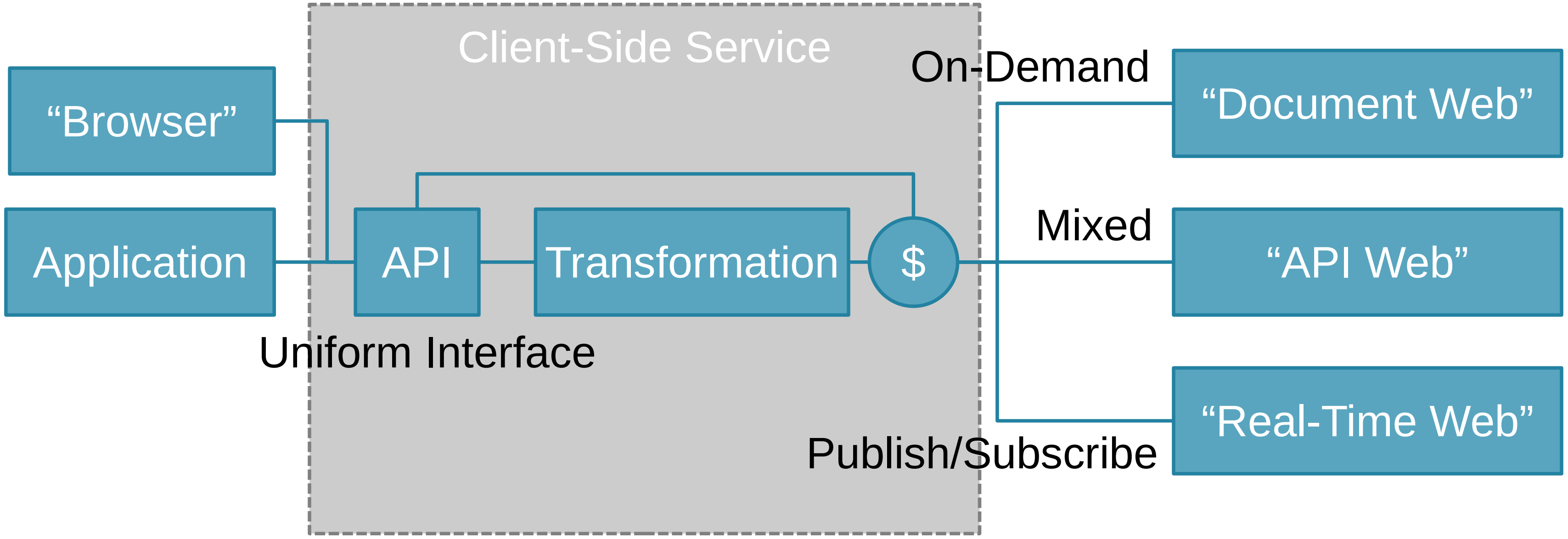
Additional/Modified Constraints #3

- Local first. APIs for applications must not expose resource location (i.e. no URLs), just identification.
  - Strengthens cache constraints and layering.
  - Induces distribution; the most distributed system is one where every node is completely independent of the rest of the system.
- Purposeful representation:
  - “JSON” is serialization format, it says nothing about purpose.
  - Requiring some purpose induces genericity via (semi-)standardization.

Additional/Modified Constraints #4

- Decoupled code-on-demand:
  - Code coupled to resource location induces centralization and data lock-in.
  - Code coupled to purposeful data format induces portability, distribution, privacy and genericity.
- Data format transformations:
  - Decouple applications from specific data formats:
    - Design for e.g. audio/opus
    - System can be extended with transformation from audio/opus to audio/speex.
  - Same as above, but stronger.

New Architecture (Simplified)

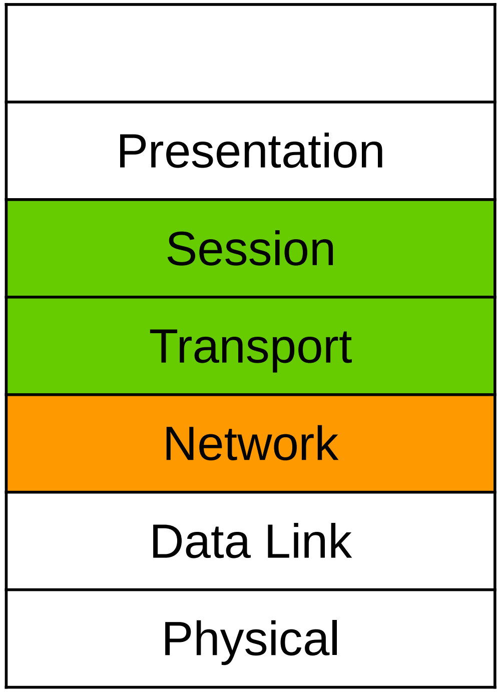


Layers "behind"/to the right of document/API/real-time as in REST.

Interpeer Project(s)

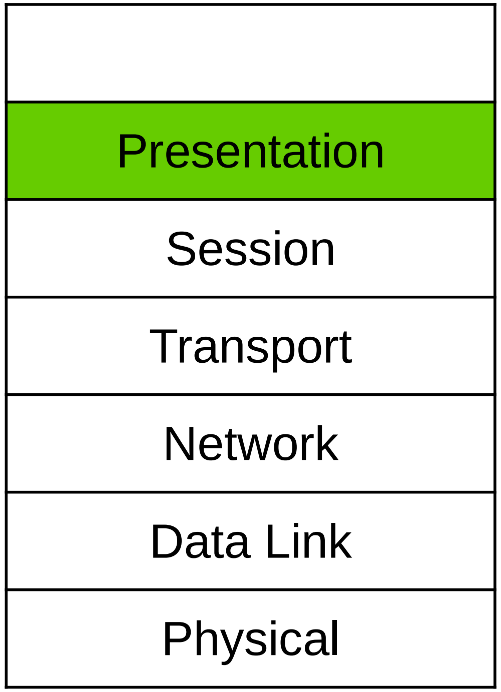
Channeler

- NGI0 funded
- Session/transport protocol with multiplexed channels with **individual** reliability capabilities.
  - Similar in some concepts to HTTP/3, but without inheriting/considering HTTP semantics.
- Encryption (next; technically presentation layer?).
- Network layer use requires routing (future).



Currently unnamed

- NGI0 funded
- Proposed (not started) presentation layer protocol with random-access and streaming modes.
- Similarity to PPSPP (RFC 7574)
- **However**, capable of exploiting reliability capabilities of channeler as best befits the application use case.
- “An exercise in abstraction”



Caprock

- ISOC funded
- Distributed authorization.
  - Based on prior work by others: OCAP/ICAP/JWT/etc.
  - Produce/consume tokens that:
    - Are location independent (can be transmitted)
    - Prove that a given action on a given object is permitted by the object’s owner.
  - Related AAA topics.
  - Mostly learning from the past and reshuffling existing ideas.
- Recently started. Want to help?

Caprock (cont.)

- OCAP (object capabilities)
  - Designed for local only inter-process communications authorization.
  - J.B. Dennis, E.C. Van Horn. “Programming Semantics for Multiprogrammed Computations.” Communications of the ACM, 9(3):143–155, March 1966
- ICAP (identity based capabilities)
  - Designed for networked IPC authorization.
  - Criticized for relying on centralized components & predates efficient cryptography.
  - Li Gong, “A Secure Identity-Based Capability System” Proceedings of the 1989 IEEE Symposium on Security and Privacy, 1989

Caprock (cont.)

- JWT (JSON web tokens; RFC 7519):
  - https://jwt.io/
  - Compact token format for multiple purposes.
  - OAuth+JWT (RFC 9068) for authentication, but centralized.
- Certificate PKIs face related issues:
  - Protocol extensions in e.g. TLS (RFC 5280) to address these (in centralized fashion).

Wyrd

- ISOC funded
- Adrian, Jens
- “Treat documents as a series of changes”
  - Simplest case: each change appends a BLOB
  - Best case: conflict-free replicated data types (CRDTs)
  - Feasible: git, etc?
- Intended to sit between presentation layer protocol and application API.

Wyrd (cont.)

- HTTP has no concept of “this is the resource I want”.
  - Has a concept of “the resource I want (probably) lives here”
  - Has a variety of headers that together narrow down which specific content is requested.
    - **BUT** server can still send something else.
  - See e.g. https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching
- Purpose of Wyrd is to provide a *uniform interface with significantly reduced complexity and efficient implementation*.

Wyrd (cont.)

- Reduce complexity:
  - Embrace that multiple versions of a resource may exist.
  - Give explicit control over which version to use to application.
    - Default e.g. to “latest”
  - Stay out of the way of document formats.
- Efficient implementation:
  - Must be aware of “change” boundaries in data stream for mapping well onto I/O subsystem (network, cache).

Wyrd (cont.)

- Uniform interface:
  - Either let the application handle file-like objects.
  - Or let the application handle sequences of well-defined content operations.
- Framework for different implementations of such content operations
  - Essentially a “content type for CRDT kind”.

Future

- Routing and discovery (partially NGI0 funded), e.g. distributed hash table.
  - Overlap with librecast project; will try to cooperate as much as feasible: https://librecast.net/
- Data transformation framework: same general API as Wyrd, but a layer on top.
  - One of the most powerful use-cases has data transformation happen server-side; good research topics here.
- A lot of glue between the pieces. Pareto principle applies.

Contributing

How can I help?

- Ask in #interpeer on https://libera.chat/
- Look at open issues on https://gitlab.com/interpeer
- Donate:
  - NLNet is a good place here in the EU
  - ISOC foundation is a good place in US/worldwide
  - Interpeer non-profit will be best, once up and running.
    - Grants are fantastic for specific R&D topics.
    - Grants do not cover bugfixing, administrative, IT/ops work well.

Questions?

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