HTTP 2 P2P

From HTTP to P2P: A survey on how to extend HTTP into a fully P2P protocol

Extending HTTP → P2P

- We can make the web fully P2P
    - With 4 backwards-compatible, orthogonal extensions
    - Where each provides independent utility and adoption incentive
- Our work is to create Protocol Specification, Tools, and Apps for each extension.

The 4 Extensions

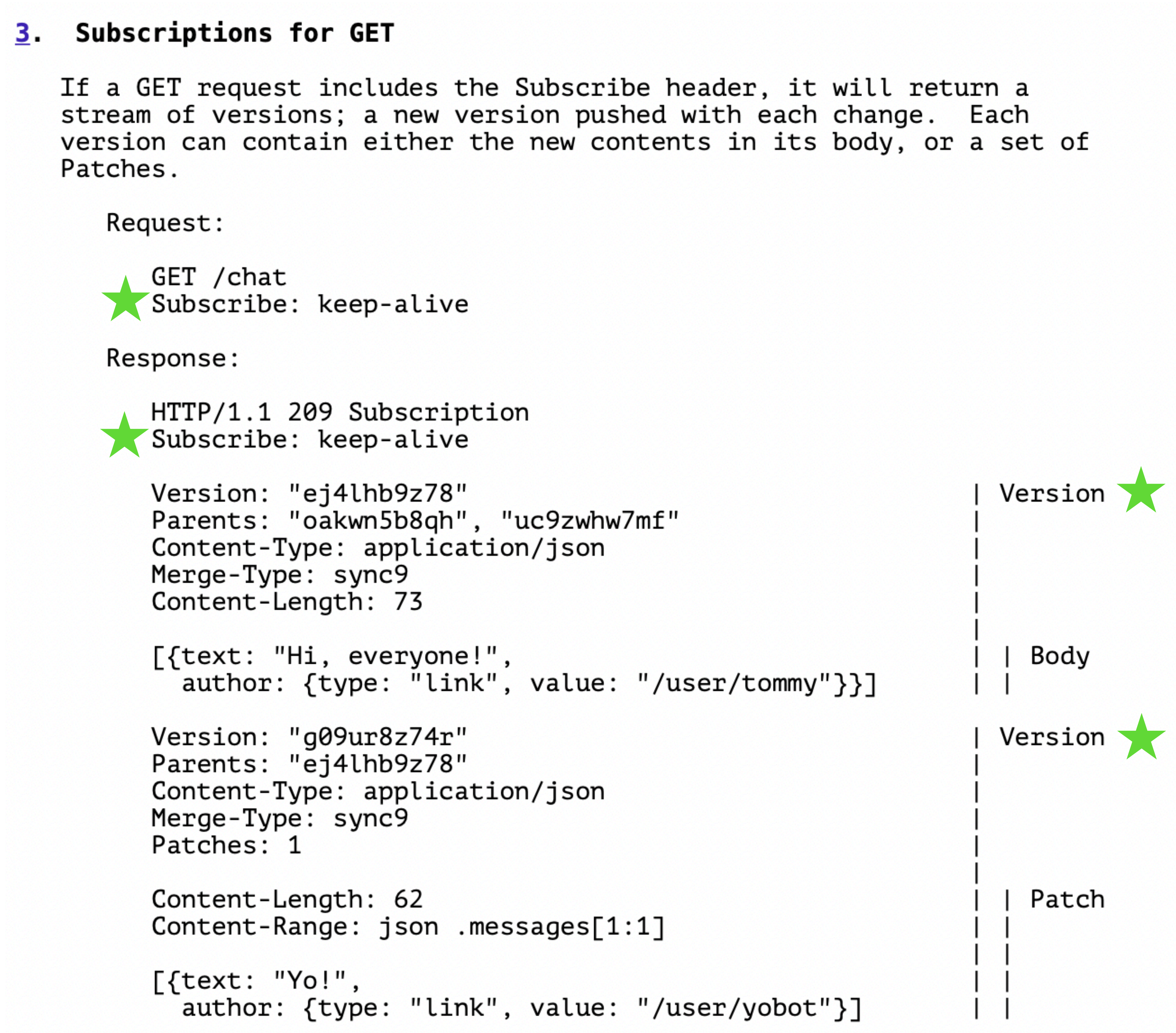
- 1. Subscriptions
- 2. Multi-writer Mutations
- 3. HTTP2P Message Semantics
- 4. HTTP2P Transport

Available in Braid-HTTP!

In Prototype

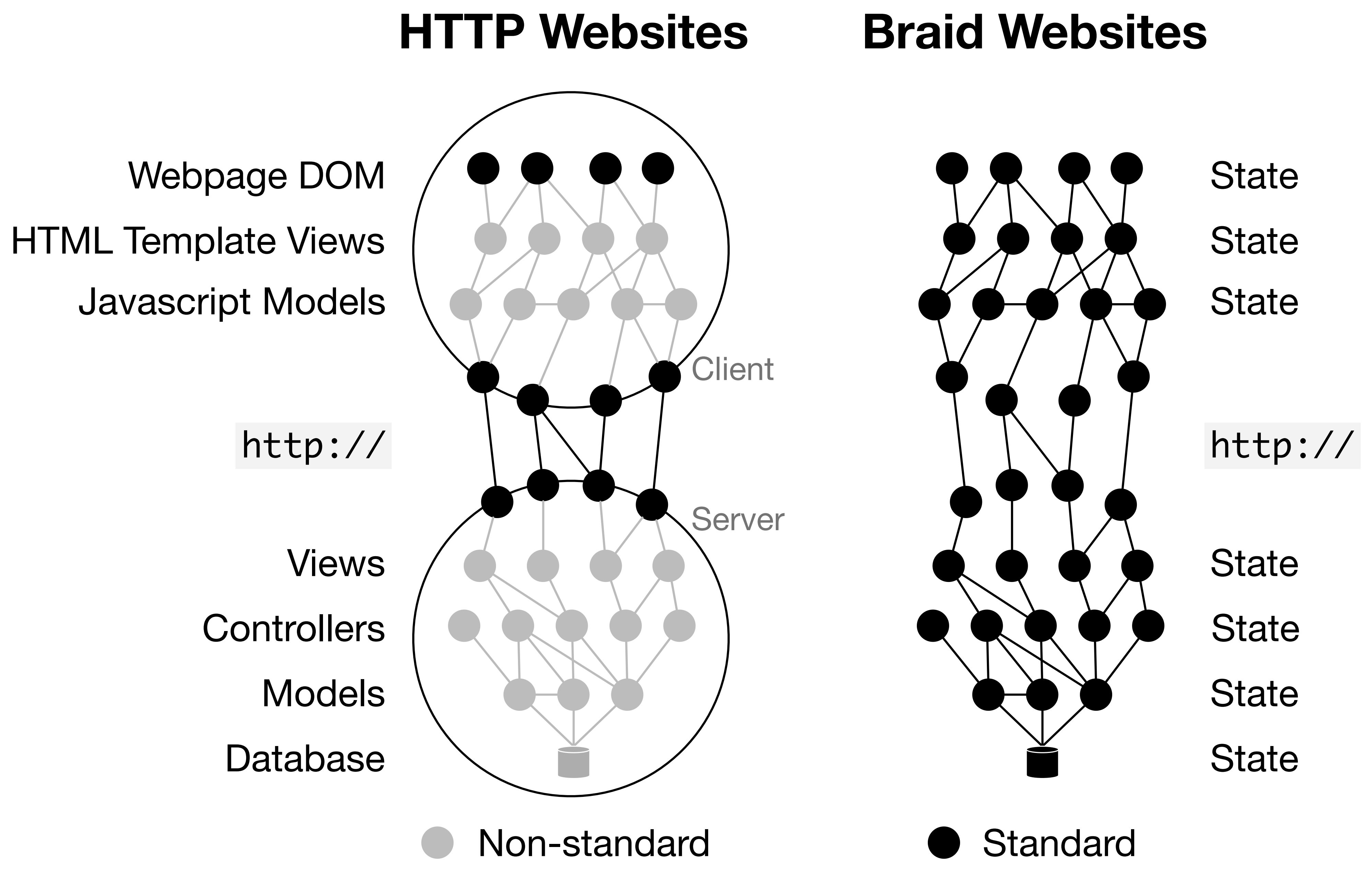
Open for discussion

1. Subscriptions



```
3.  Subscriptions for GET

   If a GET request includes the Subscribe header, it will return a
   stream of versions; a new version pushed with each change.  Each
   version can contain either the new contents in its body, or a set of
   Patches.

      Request:

         GET /chat
         Subscribe: keep-alive

      Response:

         HTTP/1.1 209 Subscription
         Subscribe: keep-alive

         Version: "ej4lhb9z78"                                 | Version
         Parents: "oakwn5b8qh", "uc9zwhw7mf"                   |
         Content-Type: application/json                        |
         Merge-Type: sync9                                     |
         Content-Length: 73                                    |
                                                               |
         [{text: "Hi, everyone!",                              | | Body
           author: {type: "link", value: "/user/tommy"}}]      | |

         Version: "g09ur8z74r"                                 | Version
         Parents: "ej4lhb9z78"                                 |
         Content-Type: application/json                        |
         Merge-Type: sync9                                     |
         Patches: 1                                            |
                                                               |
         Content-Length: 62                                    | | Patch
         Content-Range: json .messages[1:1]                    | |
                                                               | |
         [{text: "Yo!",                                        | |
           author: {type: "link", value: "/user/yobot"}]       | |
```

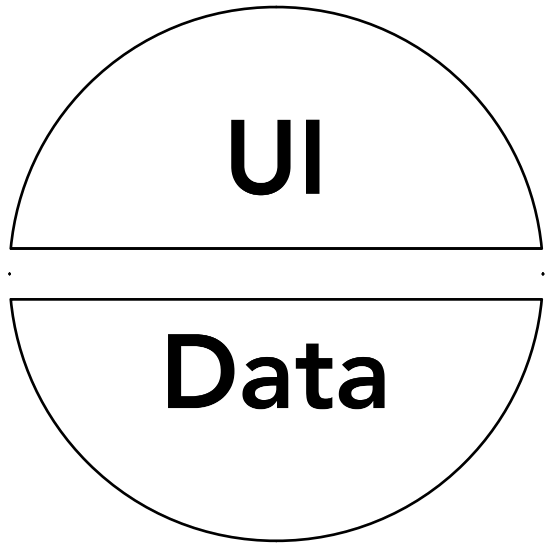
1. Subscriptions

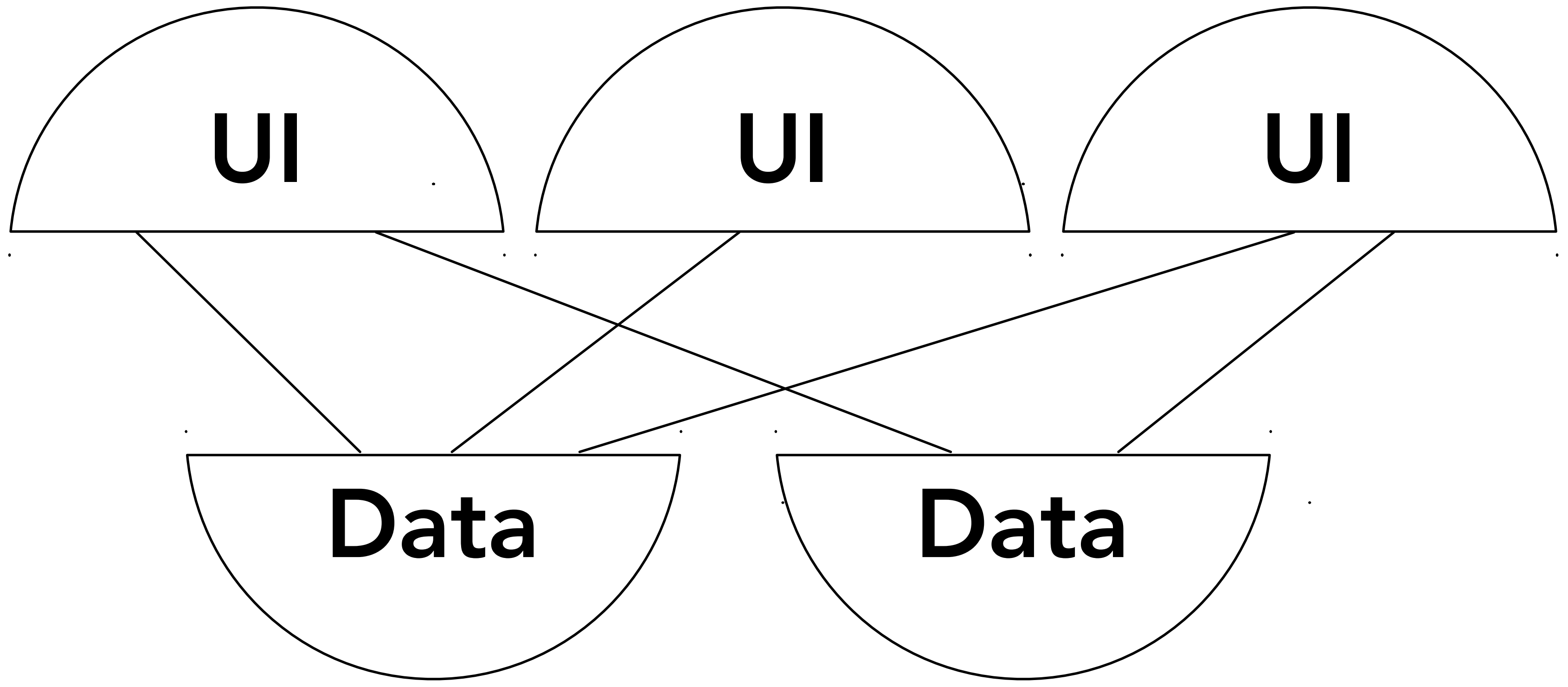


1. Subscriptions

- Standardizes the MVC cruft
- Benefits developers
    - Unifies HTTP w/ Sync tools (e.g. websocket)
    - Less to learn, and more powerful
    - Better architecture: state vs. events
        - 70% less code. React across the stack.
- Decentralizes web
    - Separates UI from Data
    - Decentralizes control over UI and attention

1. Subscriptions





Today, the owner of data
controls the interface.

Now we can separate interface from data.
Users control their interface.

2. Mutations

```
  PUT /chat
★ Version: "g09ur8z74r"                                 | Version
★ Parents: "ej4lhb9z78"                                 |
  Content-Type: application/json                         |
★ Merge-Type: sync9                                      |
  Patches: 2                                             |
                                                         |
  Content-Length: 62                                     | | Patch ★
  Content-Range: json .messages[1:1]                     | |
                                                         | |
  [{text: "Yo!",                                         | |
    author: {type: "link", value: "/user/yobot"}]        | |
                                                         |
  Content-Length: 40                                     | | Patch ★
  Content-Range: json .latest_change                     | |
                                                         | |
  {"type": "date", "value": 1573952202370}               | |
```

2. Mutations

- Guarantee Multi-Writer Consistency
  - Collaborative Editing
  - Offline mode (local-first)
- But each sub-feature also useful independently
  - Version
  - Parents
  - Patches
  - Merge-Type

2. Mutations

- Examples:
  - Versioning your <script src=“foo.js”> files (only need *version*)
  - Subscribe to server logs (only need *patches*)
  - Append-only chat (only need *patches*)
    - Reconnect (need *version*)
  - CDNs hosting dynamic state
  - WebRTC fallback when server dies (need *version* & *parents*)
    - User community could fork a website

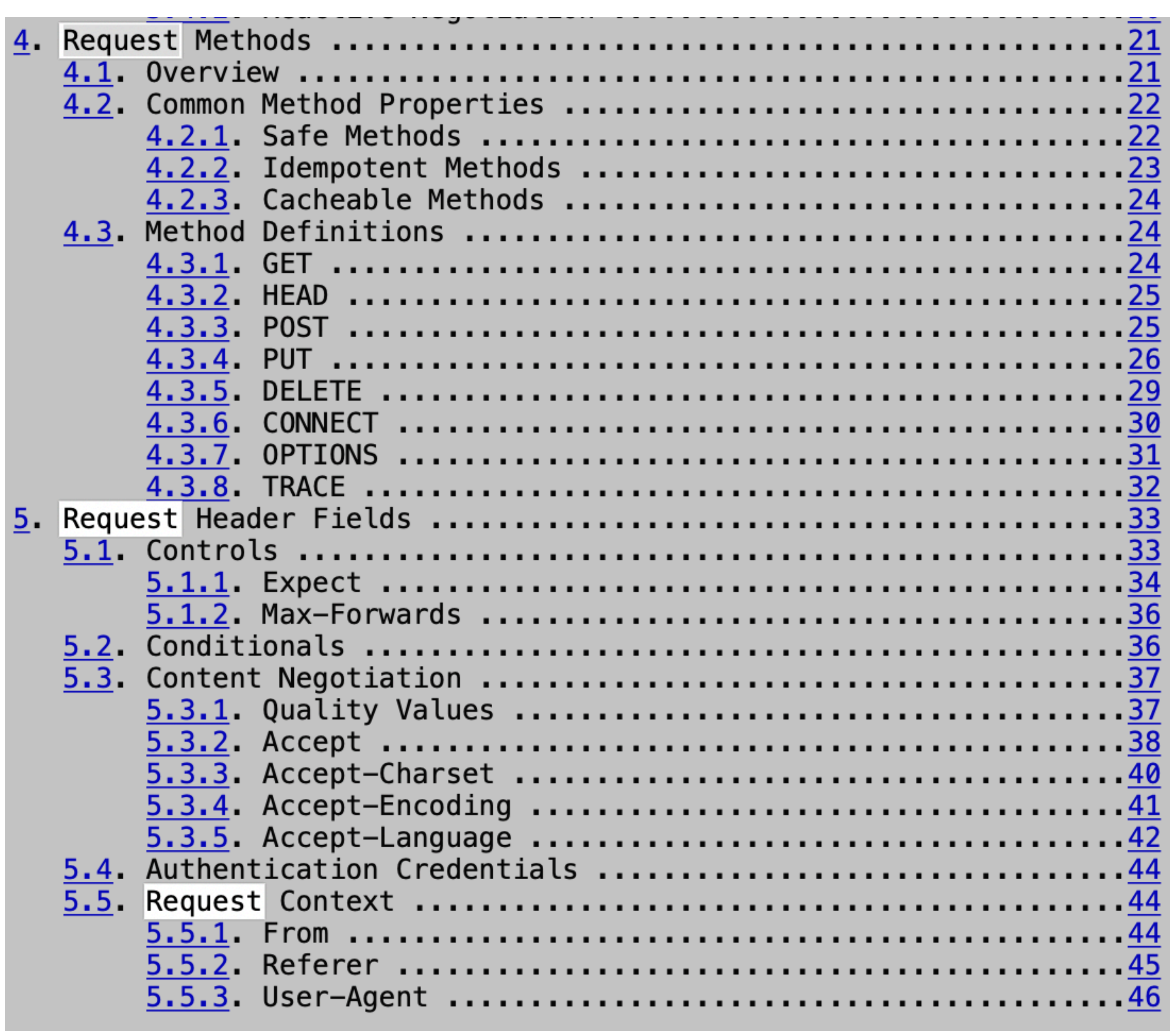
2. Mutations

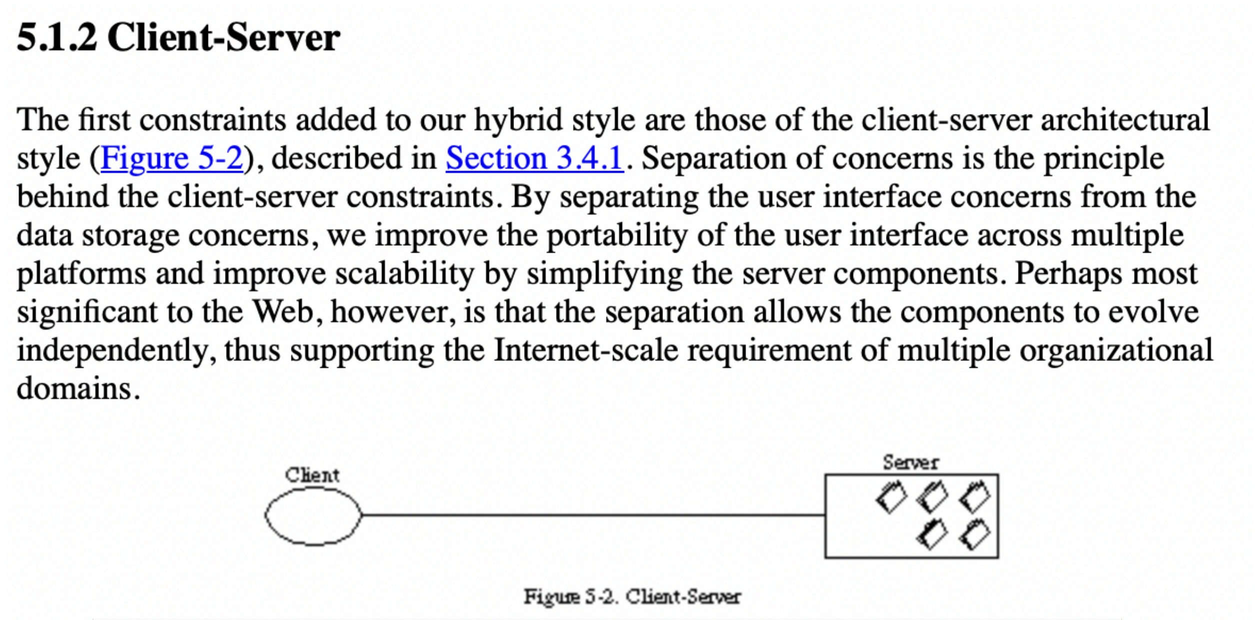
- Enables:
  - Collaborative Editing
  - Local-first Offline Mode
  - Dynamic CDNs
- Enables app-specific decentralization:
  - WebRTC P2P networking fallback
  - Community can fork away from malicious server

3. P2P Message Semantics

- HTTP is
  - Request/Response
  - Client/Server
  - GET, PUT, POST, DELETE
- This limits the behavior of
  - Mutations
  - Validation
  - Acknowledgements

3. P2P Message Semantics





5.1.2 Client-Server

The first constraints added to our hybrid style are those of the client-server architectural style (Figure 5-2), described in Section 3.4.1. Separation of concerns is the principle behind the client-server constraints. By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components. Perhaps most significant to the Web, however, is that the separation allows the components to evolve independently, thus supporting the Internet-scale requirement of multiple organizational domains.

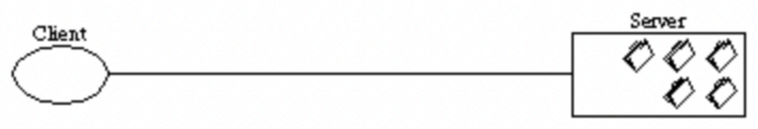
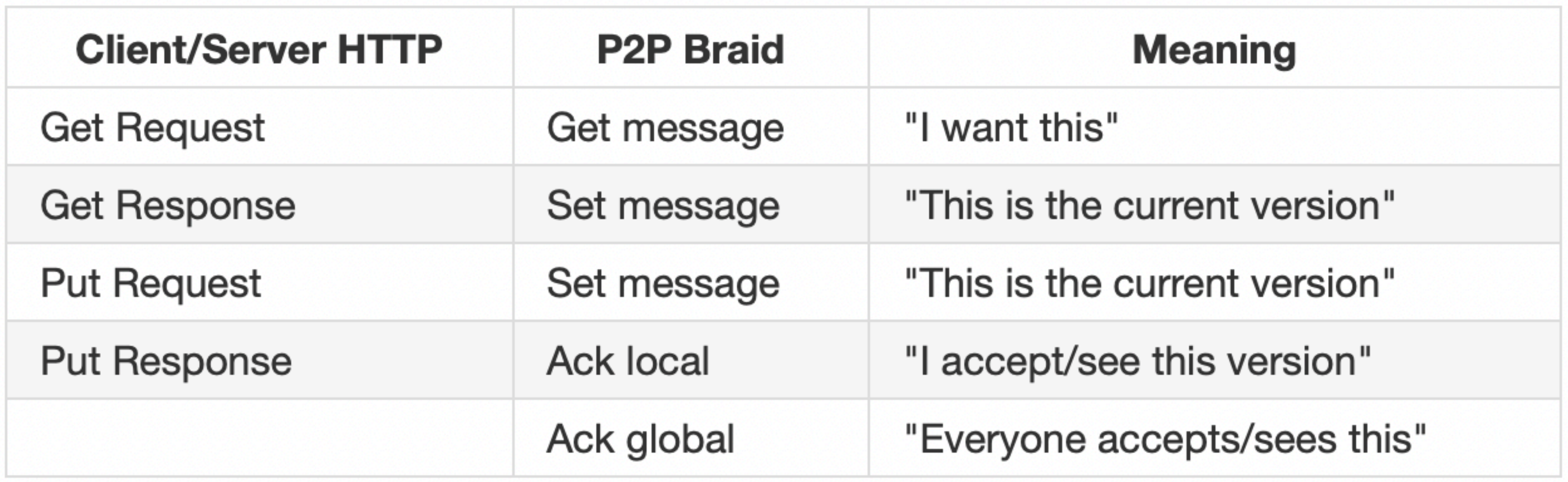


Figure 5-2. Client-Server

HTTP needs P2P Semantics for:

- Mutation
- Validation
- Acknowledgement

3. P2P Message Semantics



Client/Server HTTP	P2P Braid	Meaning
Get Request	Get message	"I want this"
Get Response	Set message	"This is the current version"
Put Request	Set message	"This is the current version"
Put Response	Ack local	"I accept/see this version"
	Ack global	"Everyone accepts/sees this"

Re-imagines HTTP Methods as P2P instead of Client/Server!

Mutation, Validation, Acknowledgement

3. P2P Message Semantics

Client/Server HTTP	P2P Braid	Meaning
	Welcome	"A peer joined"
	Fissure	"A peer left"

Re-imagines HTTP Methods as P2P instead of Client/Server!

Mutation, Validation, Acknowledgement

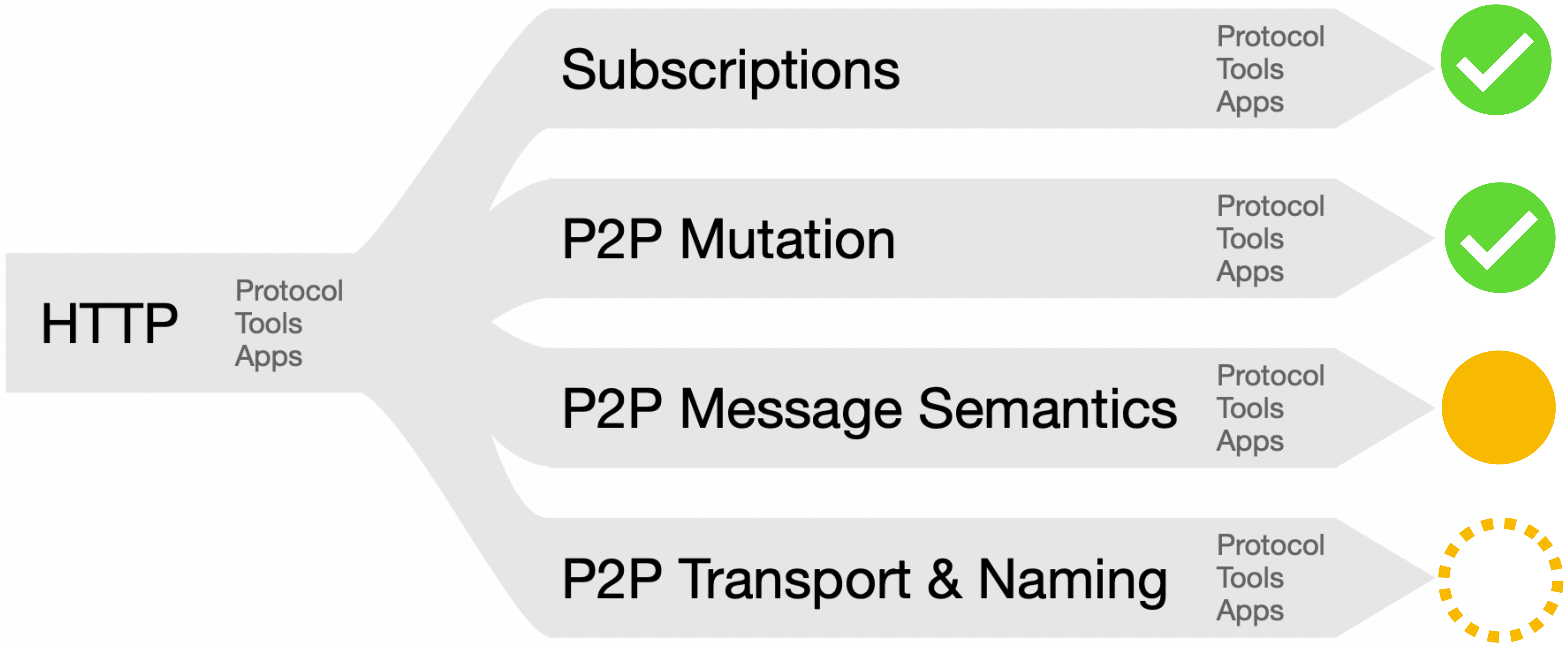
3. P2P Message Semantics

- Enables:
  - Server can request state from client
  - Client can request state from peer
  - Server1 can request state from server2
    - Server1 can validate based on Server2 + db
  - Multi-server transactions can validate via client
  - Validation rules can be standardized and distributed
  - P2P network that prunes history automatically

4. P2P Transport

- Transport
  - HTTP2 already has P2P message frames
- Naming (DNS)
- Encryption (TLS)
  - Identity (certificates)
- Route-finding
- URLs

Overview of Work



4. P2P Transport

Discussion:

Many Dweb projects focus on **Transport**—but the web's immediate pain points are in **Subscriptions** and **Mutations**.