

A State Synchronization Working Group

Michael Toomim
Dispatch IETF 118 Prague

Most application networking protocols implement **State Synchronization**

FTP is about Synchronizing (the state of two peers)

- Van Jacobson

Examples: FTP, HTTP, POP/IMAP/JMAP, WebDAV, SIP, SCIM, COAP, LDAP, IPP, NFS, YANG Push, WebSub, ActivityPub, Matrix, Mercure, DNS, ALTO.

Also at other layers: OSPF, BGP, ROHC, IS-IS...

State Synchronization protocols start simple

- "Send state X to Alice"

State Synchronization protocols start simple

- "Send state X to Alice"

But over time, we want:

- **Performance:** Realtime updates, pushed, delta-compressed, pub/sub
- **Collaboration:** Multiple editors over the network
- **Reliability:** Consistent vs. race conditions & machine failures
- **Decentralization:** New network topologies, offline peers
- **Versioning:** History, archiving, rewinding & playback

State Synchronization protocols start simple

- "Send state X to Alice"

But over time, we want:

- **Performance:** Realtime updates, pushed, delta-compressed, pub/sub
- **Collaboration:** Multiple editors over the network
- **Reliability:** Consistent vs. race conditions & machine failures
- **Decentralization:** New network topologies, offline peers

- **Versioning:**

But we also want:

- To not have to spec & implement it all ourselves
- To interoperate with other systems

Examples: a History of Sacrifices

FTP

POP → IMAP → JMAP

HTTP

→ WebDAV → CalDAV

→ JMAP

→ SCIM

→ SSE → ALTO

→ WebSub

→ Mercure

→ ActivityPub

→ Matrix

→ IPP

→ RESTCONF w/ YANG Push

→ PREP & Braid-HTTP

SIP

COAP

LDAP

NFS

NETCONF w/ YANG Push

ALTO → w/ SSE

DNS

ROHC

BGP

OSPF

IS-IS

Example: ALTO Incremental Updates

RFC 8895

Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol (RFC 7285) provides network-related information, called network information resources, to client applications so that clients can make informed decisions in utilizing network resources. This document presents a mechanism to allow an ALTO server to push updates to ALTO clients to achieve two benefits: (1) updates can be incremental, in that if only a small section of an information resource changes, the ALTO server can send just the changes and (2) updates can be immediate, in that the ALTO server can send updates as soon as they are available.

Example: SCIM Change Detection (IETF 117 in SF)



Goal

Provide a scalable and highly accurate method of change detection that allows a SCIM client to retrieve the current state of all resources that have changed since a prior point in order to perform a recurring incremental retrieval of data.

Some of the *possible use cases* where this can be used:

1. **Reconciliation System:**
 - a. There is a need to build a reconciliation system to identify undetected diverging data between a SCIM Client and SCIM Server in order to prevent undesirable authorization decisions based on incorrect data.
 - b. The data divergence detection may be used for reporting purposes or may be extended to either trigger provisioning of those resources in the target system or pulling changes from the target system into the source.
 - c. Without a set of end-to-end reconciliation processes, service providers cannot confidently say that there are no divergence-caused security incidents.
2. **Incremental Synchronization** where client pulls data from the server.
 - a. Example: Identity Provider pulling data from an HR system.

A State Synchronization Working Group

- **A place to solve the General State Synchronization Problem**
- Produces:
 - Abstract specifications
 - Common libraries
 - Concrete recommendations for other WGs
- Process:
 - Select features to adopt into group's scope of work
 - Start simple, then increase scope to **demand** and **doability**
 - Examine protocols → Create common models → Specify
 - Build and verify utility
 - Test interoperability with existing implementations
 - Propose concrete extensions for other WGs protocols

Example: **Braid.org** group and **Braid-HTTP**

- Braid.org: *informal* Working Group for Interoperable State Synchronization
 - Born at IETF 105, four years ago
- Four features solved in Braid-HTTP
 - Subscriptions
 - Patches
 - Version history
 - Merge consistency spec
- Multiple implementations:
 - Tested: Compatible with all existing CRDT & OT algorithms
 - Next: test compatibility with SIP, IMAP/JMAP, CalDAV, etc.
- Vision: to **read & write distributed state as easily as a local variable**

Result: Raise the level of Abstraction!

```
▶  |  | top ▼ |  | Filter
> state['https://dt.braid.org/json/chat']
< [{"user":"mike","text":"Hello everyone!"}]
> state['https://dt.braid.org/json/chat'].push({user: 'bob', text: 'Hi!'})
< 2
> state['https://dt.braid.org/json/chat'][0].user = 'alice'
< 'alice'
```

Version

Range

Content



root-0

json

= [{"user":"mike","text":"Hello everyone!"}]

root-1

json ["1"]

= {"user":"bob","text":"Hi!"}

root-2

json ["0"]["user"]

= "alice"

Conclusion: a State Synchronization Working Group

- Raise level of abstraction for both specifications and implementations
- Create a common reusable distributed state substrate
- Increase scope of features over time
 - *e.g.* live queries, consensus, validation, state-centric routing, low-level networking...

Question: Research Group vs. Working Group?

- Best standards come from marriage of **researchers** and **implementers**
- IRTF can do research, but not standards
- IETF can do standards, and include research