YAHOO! FINANCE TechCrunch YAHOO! SPORTS HUFFPOSTI MAKERS YAHOO! NEWS engadget YAHOO! MAIL YAHOO! ENTERTAINMENT Aol. BUILD RYOT

Outbound Refactoring

"I'll clean it up when it works" and other jokes we tell ourselves.

Aaron Canary • ATS Summit Fall 2019

verizon media

Why you (might) care?

Refactoring of core TS code

Making HttpSM dev friendly

Required for Http2-to-Origin

Required for Layer 7 Routing

I want you to understand how these are related and build off one another.

Agenda

1. Cleanup
2. UML
3. Refactor Http1 (WIP)
4. Propose Http2 Work
5. Propose Layer 7 Work
6. Follow up on Thursday

Cleanup

- **Renaming files & classes**
  - HttpServerSession -> Http1ServerSession
  - ProxyClientTransaction -> ProxyTransaction
- **Replaced ProxyTxn::outbound vars with accept::options**
- **Moved ProxyTxn::host_res_style to http_sm::t_state.dns_info**
- 🔥 **Removed ProxyTxn::restart_immediate**

verizon media

But I never could see, or wrap my head around what was happening.

When you have clean declarations

Thank you to the Http3 Team.

This is the entire definition if H3Session interface. Once I read this, its crystal clear I needed to clean up the H1 and H2 headers before continuing any further work.

verizon media

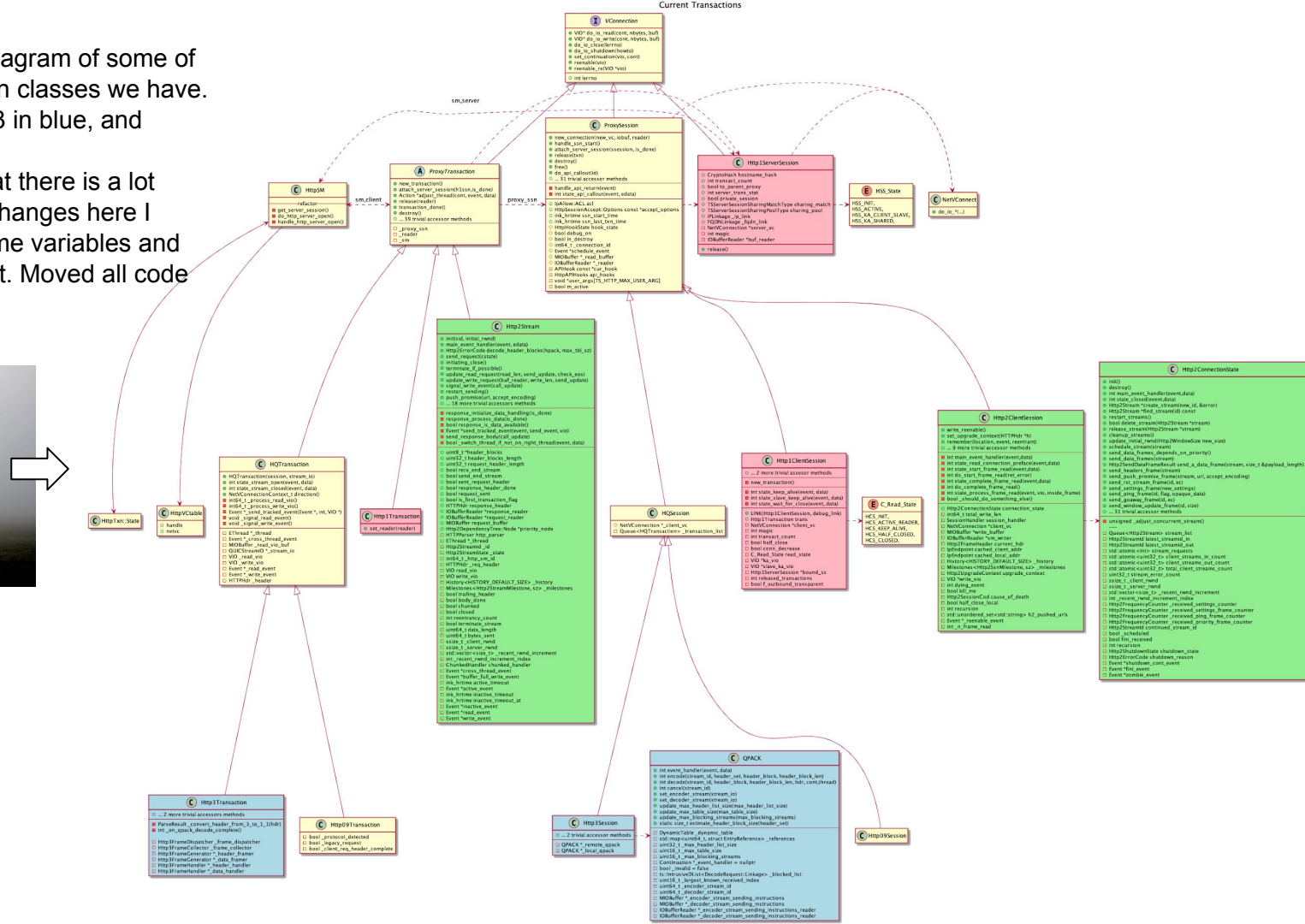
```
class HQSession : public ProxySession
{
public:
  using super = ProxySession; ///< Parent type

  HQSession(NetVConnection *vc) : _client_vc(vc){};
  virtual ~HQSession();

  // Implement VConnection interface
  VIO *do_io_read(Continuation *c, int64_t nbytes = INT64_MAX, MIOBuffer *buf = nullptr) override;
  VIO *do_io_write(Continuation *c = nullptr, int64_t nbytes = INT64_MAX, IOBufferReader *buf = 0, bool owner = false) override;
  void do_io_close(int lerrno = -1) override;
  void do_io_shutdown(ShutdownHowTo_t howto) override;
  void reenable(VIO *vio) override;

  // Implement ProxySession interface
  void new_connection(NetVConnection *new_vc, MIOBuffer *iobuf, IOBufferReader *reader) override;
  void start() override;
  void destroy() override;
  void release(ProxyTransaction *trans) override;
  NetVConnection *get_netvc() const override;
  int get_transact_count() const override;

  // HQSession
  void add_transaction(HQTransaction *);
  HQTransaction *get_transaction(QUICStreamId);

protected:
  NetVConnection *_client_vc = nullptr;

private:
  // this should be unordered map?
  Queue<HQTransaction> _transaction_list;
};

class Http3Session : public HQSession
{
public:
  using super = HQSession; ///< Parent type

  Http3Session(NetVConnection *vc);
  ~Http3Session();

  // ProxySession interface
  const char *get_protocol_string() const override;
  int populate_protocol(std::string_view *result, int size) const override;
  void increment_current_active_client_connections_stat() override;
  void decrement_current_active_client_connections_stat() override;

  QPACK *local_qpack();
  QPACK *remote_qpack();

private:
  QPACK *_remote_qpack = nullptr; // QPACK for decoding
  QPACK *_local_qpack  = nullptr; // QPACK for encoding
};
```

Cleanup (cont.)

- **Renaming ProxySession members**
  - proxy_ssn -> _proxy_ssn
  - current_reader -> _sm
  - sm_reader -> _reader
- **Moved all function definitions to .cc**

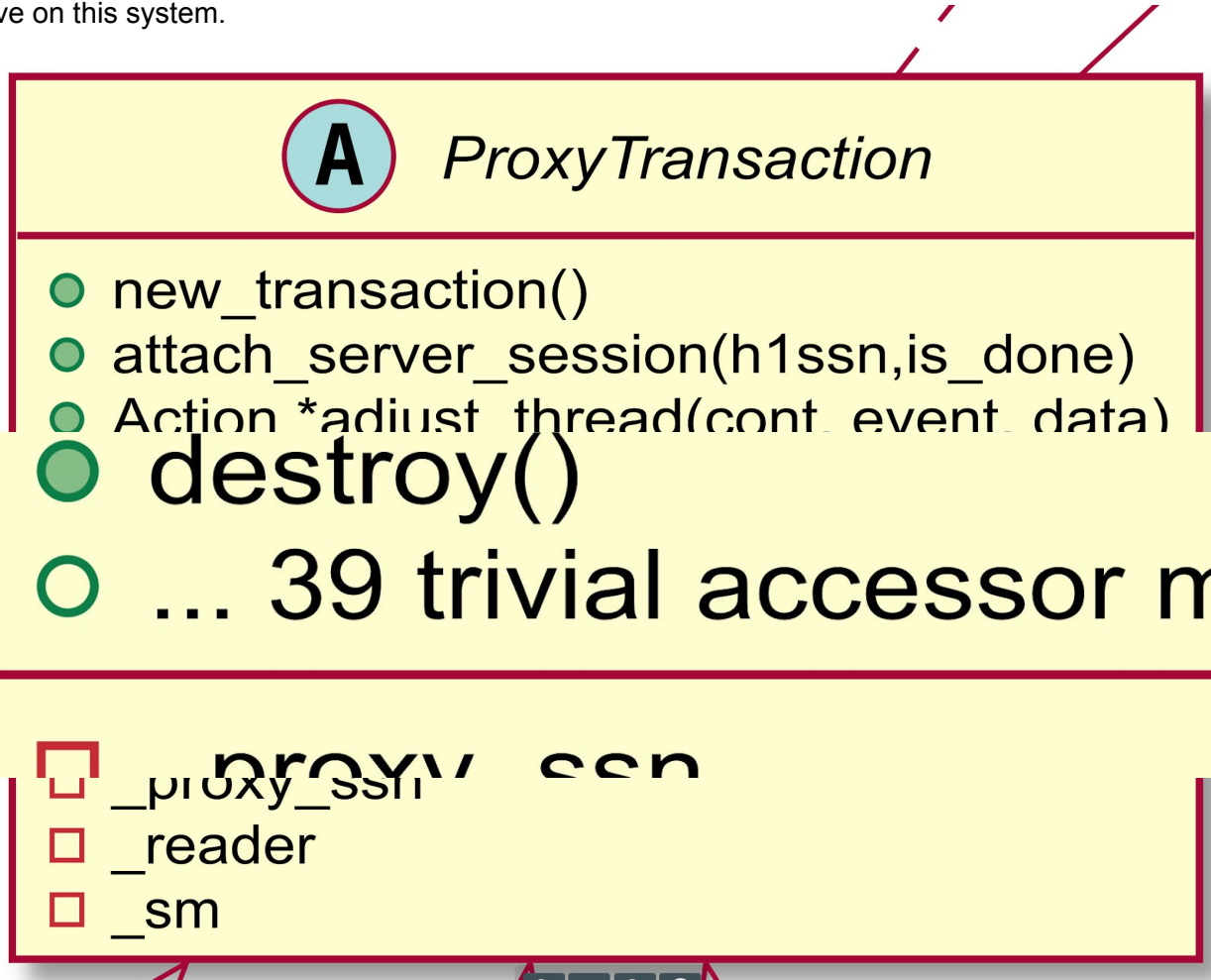
verizon media

This is a simplified class diagram of some of the transaction and session classes we have. H1 in pink, H2 in green, H3 in blue, and abstract classes in yellow.
The point I'm making is that there is a lot here. Before making any changes here I wanted to clean up. Rename variables and methods. Remove the cruft. Moved all code the .cc file.



verizon media

When I say simplified, I mean that I collapsed all trivial accessor methods, and abbreviated method signatures. So this puts a steep learning curve on this system.

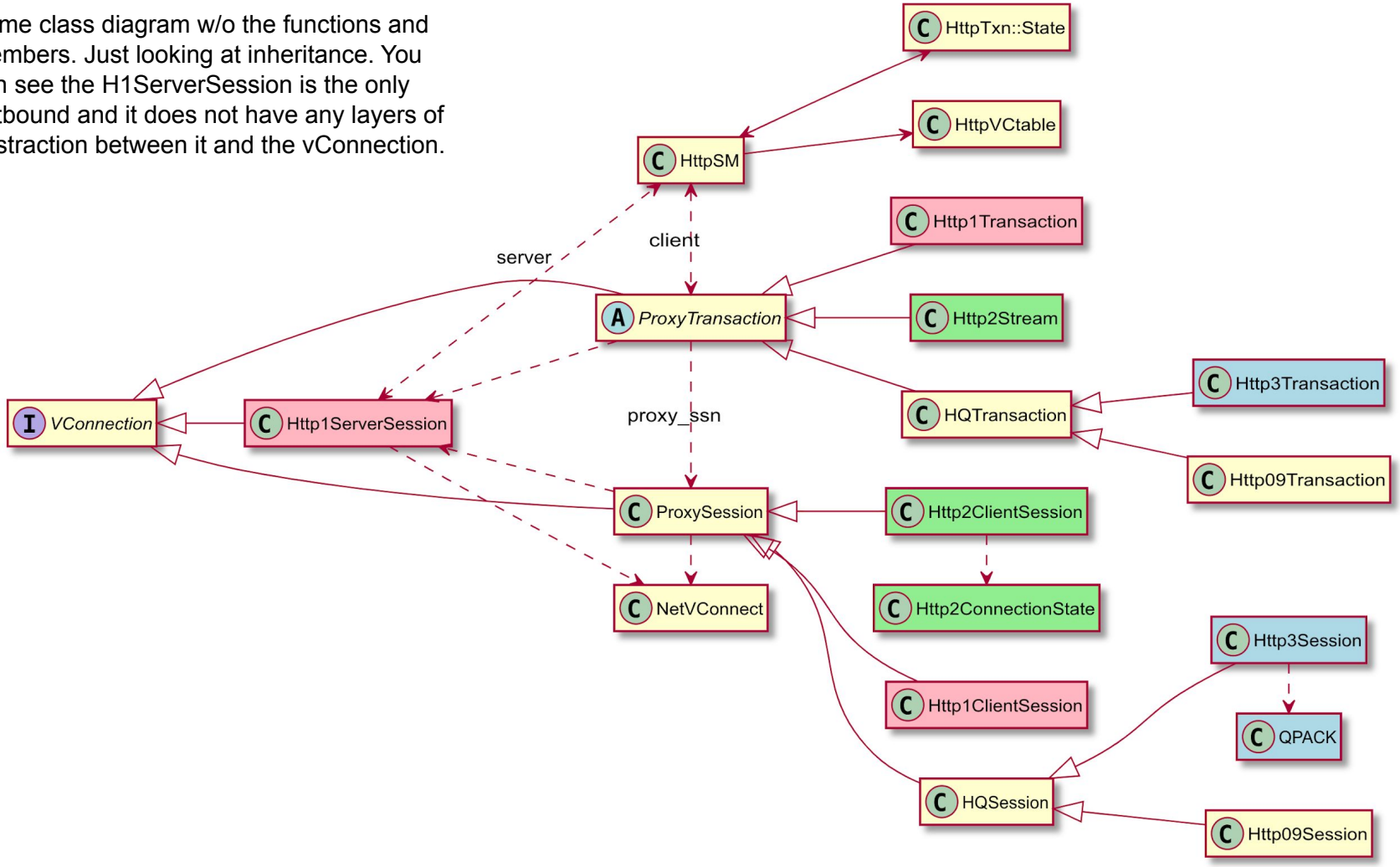


Http1 Refactor In Progress

Goals:

- **Simplify HttpSM**
- **Foundation for outbound logic (i.e. H2-to-Origin)**
- **Maintain Parity**
- **Http1ServerSesson derive from ProxySesson**
- **Abstract Http1 code out of SM, into Http1Sesson**
- **Feature branch: h1outbound**
  - Merge into 9.1+

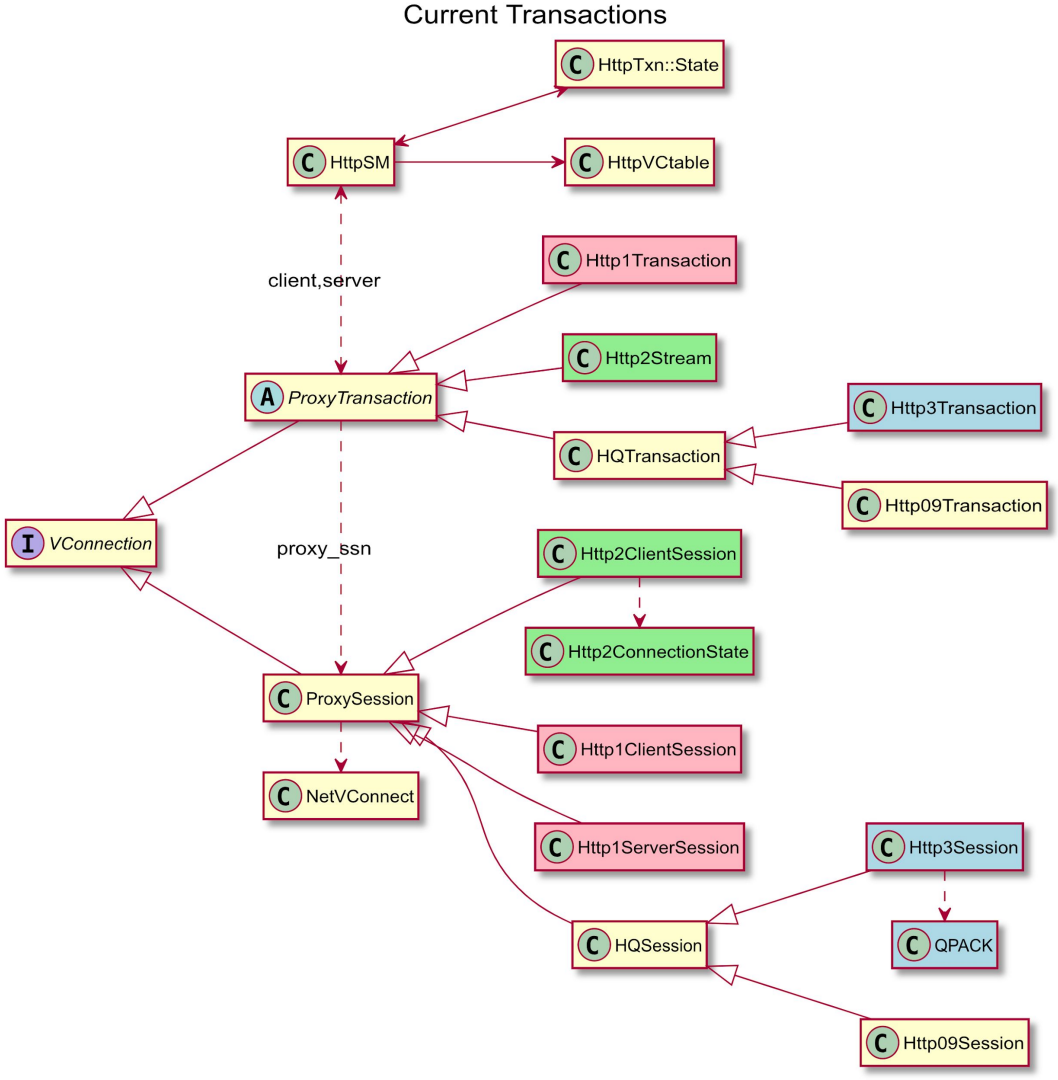
Same class diagram w/o the functions and members. Just looking at inheritance. You can see the H1ServerSession is the only outbound and it does not have any layers of abstraction between it and the vConnection.



Post Http1 Refactor

HttpSM only interfaces with ProxyTransaction

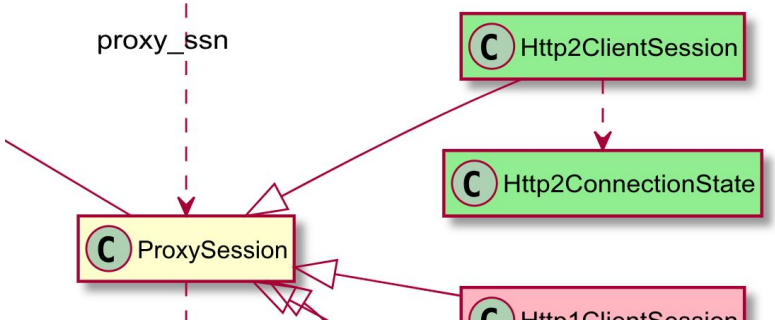
Now that HttpSM only interfaces with ProxyTransactions, it will be much easier to add outbound transaction logic, which we will do for H2-to-origin and Layer 7 routing.

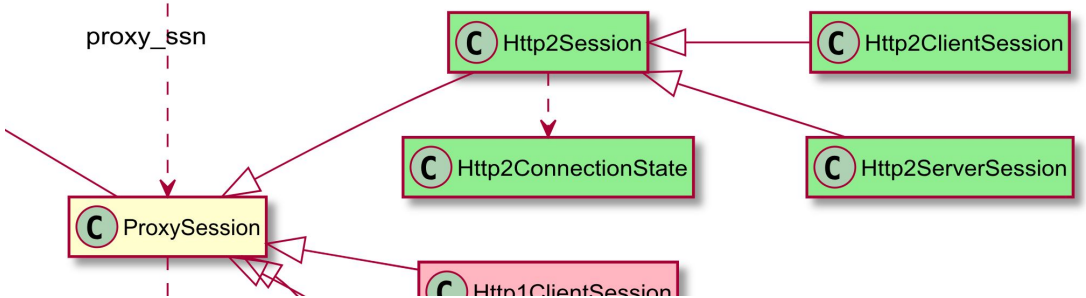


verizon media

Http2 Refactor

- Abstract Http2Session from Http2ClientSession
- Write Http2ServerSession





verizon media

Http2 Outbound

1. **Write H2StreamVacancy**

   Management layer to allocate streams on existing H2 connections

2. **Outbound Session Start/End Hooks**

   Expands API

3. **Test H2-to-Origin**

   Merge into ATS 9.x

verizon media

Pre-L7R Cleanup Interest

1. Class Allocators -> new/delete w/ jemalloc
2. create/destroy() -> class constructor/destructor
3. HttpVCTable -> ~ProxyTranaction()
4. TxnArgs -> Extendible
5. Create Object Oriented Storage w/ Plugin API
6. HostDB -> HostObj

verizon media

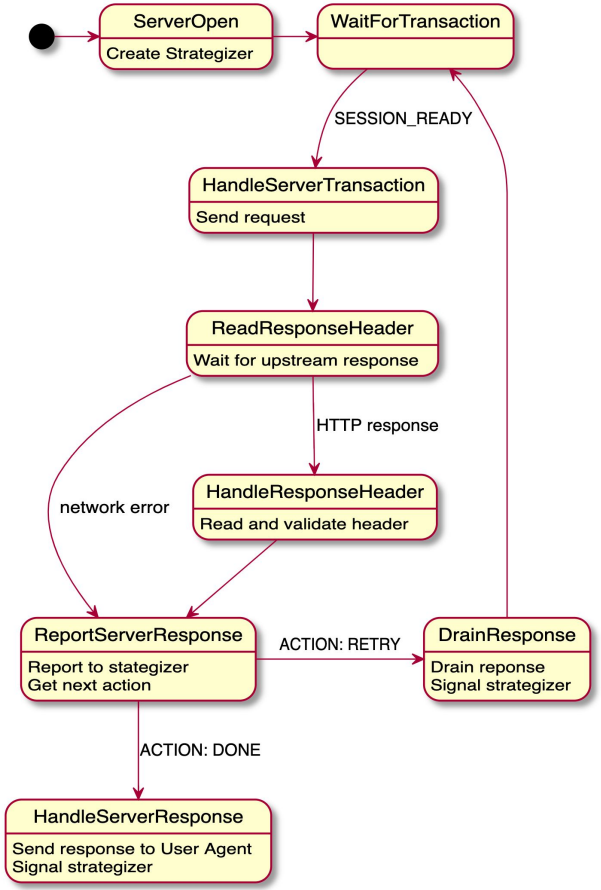
Layer 7 Refactoring

1. Isolate Upstream Selection Logic from HttpSM
2. Health Check Plugin using NetChasm daemon
3. Parent Selection Plugin
4. CDN Routing Config

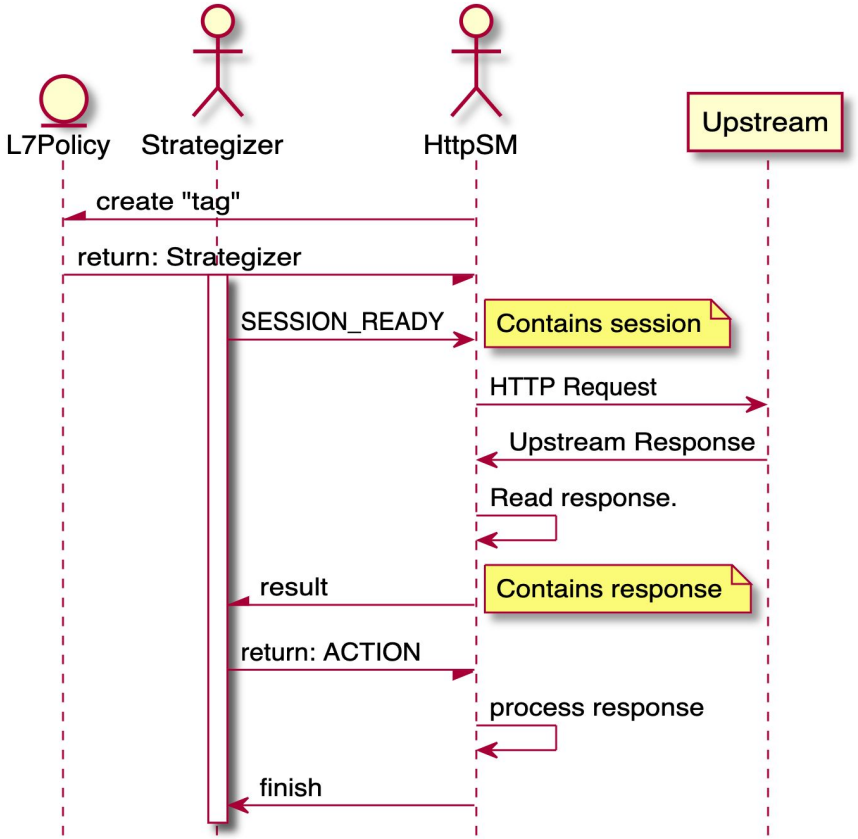
verizon media

Questions

Lets talk on Thursday

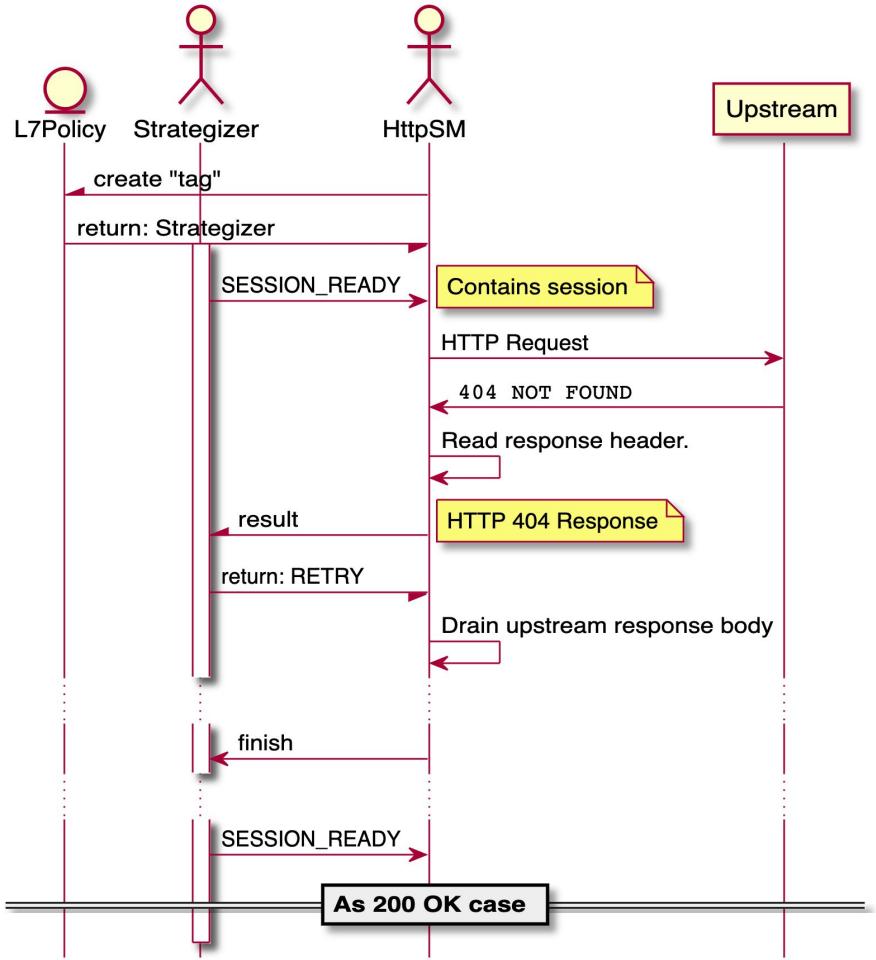


State Machine Upstream States



verizon media

Generic Transaction



verizon media