YAHOO!

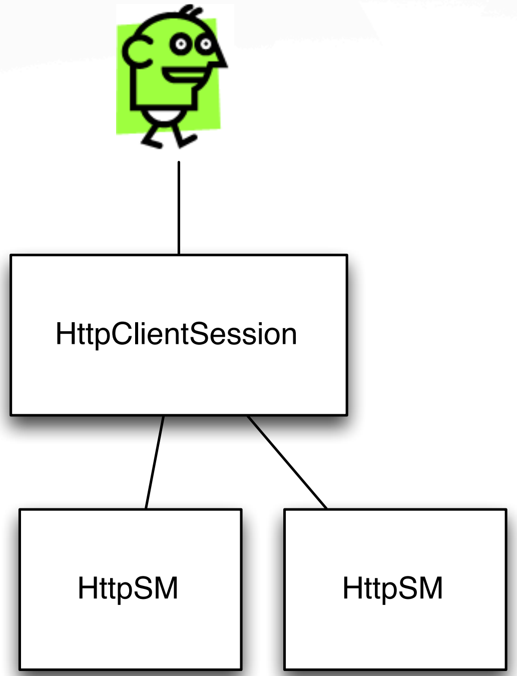
TS-3612: Detangling Sessions and Transactions

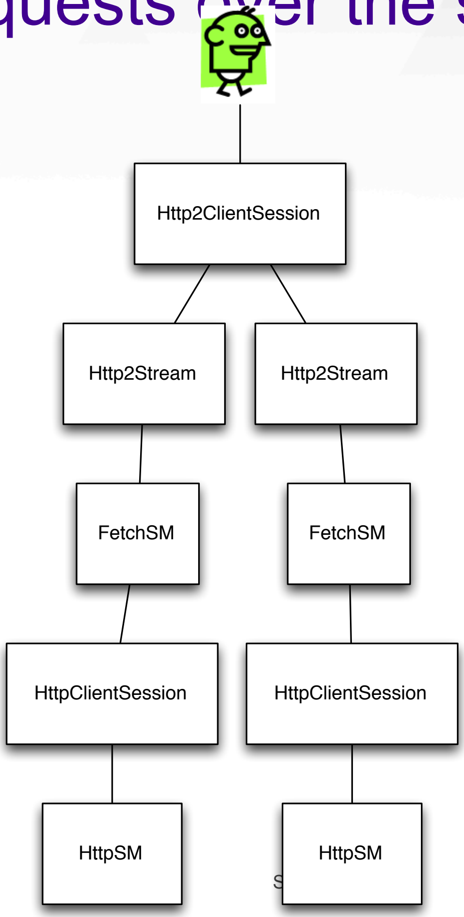
Susan Hinrichs – ATS Summit Spring 2016

The Problem

- ATS was built with HTTP 1.0 and HTTP 1.1 in mind.
  - HttpClientSession represents the user agent in the HTTP State Machine
- HTTP 1.x allows one outstanding transaction per connection/session
  - HttpClientSession mixes the session and transaction logic
  - HttpSM mostly represents the transaction, but there is some protocol-specific transaction data you’d like to model.
    - E.g., stream ID, protocol translation data, protocol state machine logic
- SPDY and HTTP/2 allow multiple outstanding transactions
  - Model independent streams (Transactions) running over the same session
- ATS uses FetchSM to map the streams onto HTTP 1.x session/ transactions

Object Creation to execute two requests over the same session





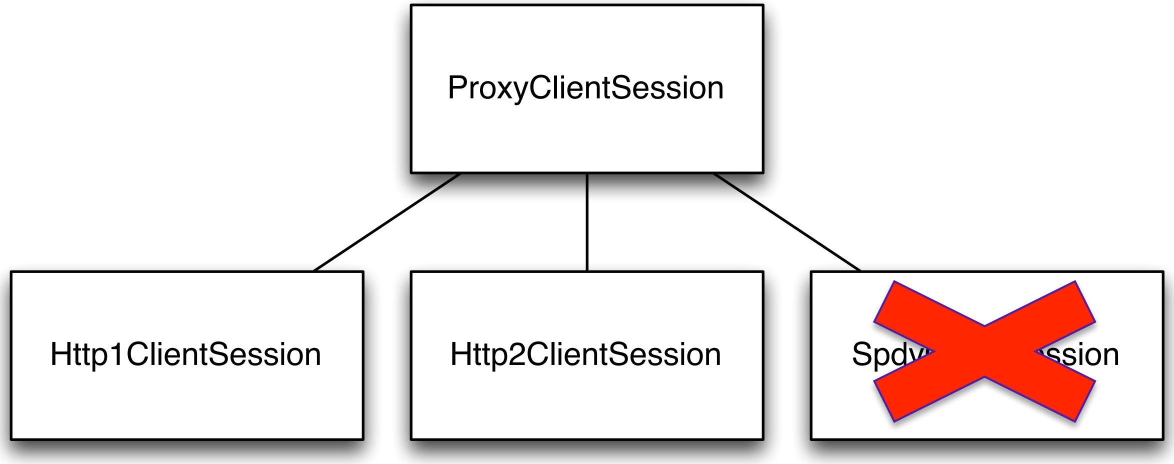
The Problem in Specifics

- Cannot get to the network level information (e.g. addresses and session-oriented logging info) (TS-4012)
  - In general it is not possible (yet very useful) to be able to track back from an HttpSM to the overall client session (network level).
- Session and Transaction hooks are confused (TS-3612, TS-3578, TS-2748, TS-1007).
  - Http1.1 gets the Session hooks called on each transaction even if the transactions are on the same network session.
- Debugging through FetchSM is awkward.
- HTTP/2 must treat its transactions as black boxes due to FetchSM's architecture.

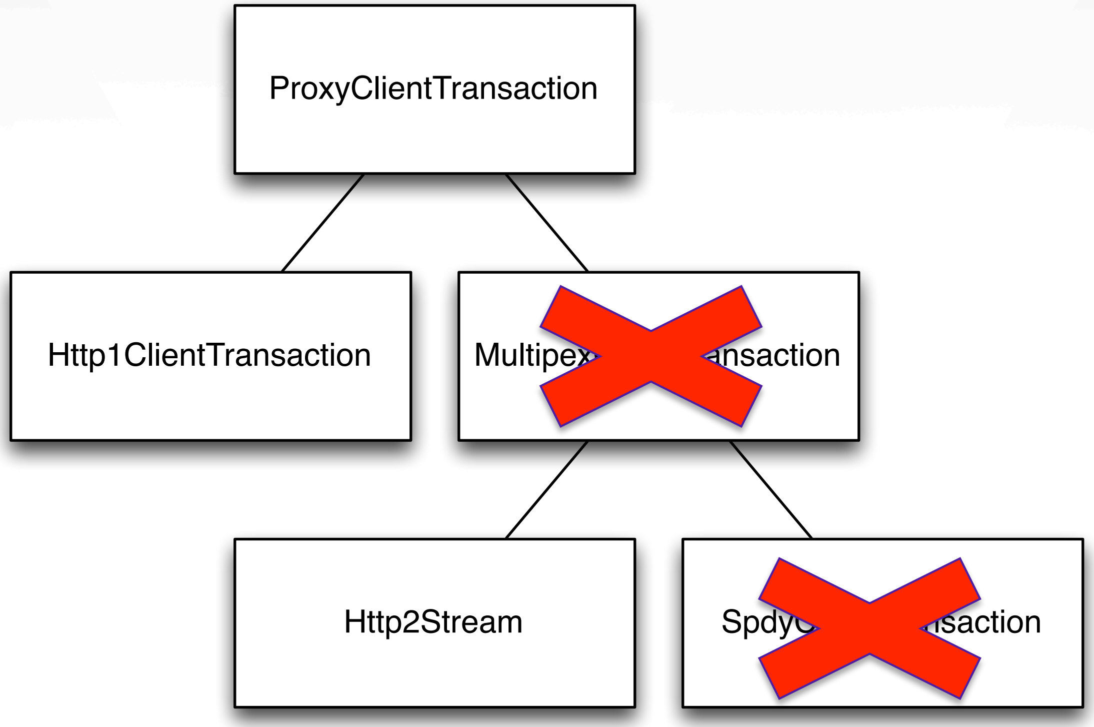
The Solution

- Explicitly separate Session and Transaction
  - Create a ProxyClientTransaction hierarchy in parallel to the ProxyClientSession hierarchy.
- Remove FetchSM and have protocols directly spawn and interact with HttpSM
- HttpSM interacts with ProxyClientTransaction object rather than HttpClientSession
- Design Proposal
  - https://www.dropbox.com/s/j87lph2z66vx05t/ProxyClientSessionRe-architectProposal.pdf?dl=0

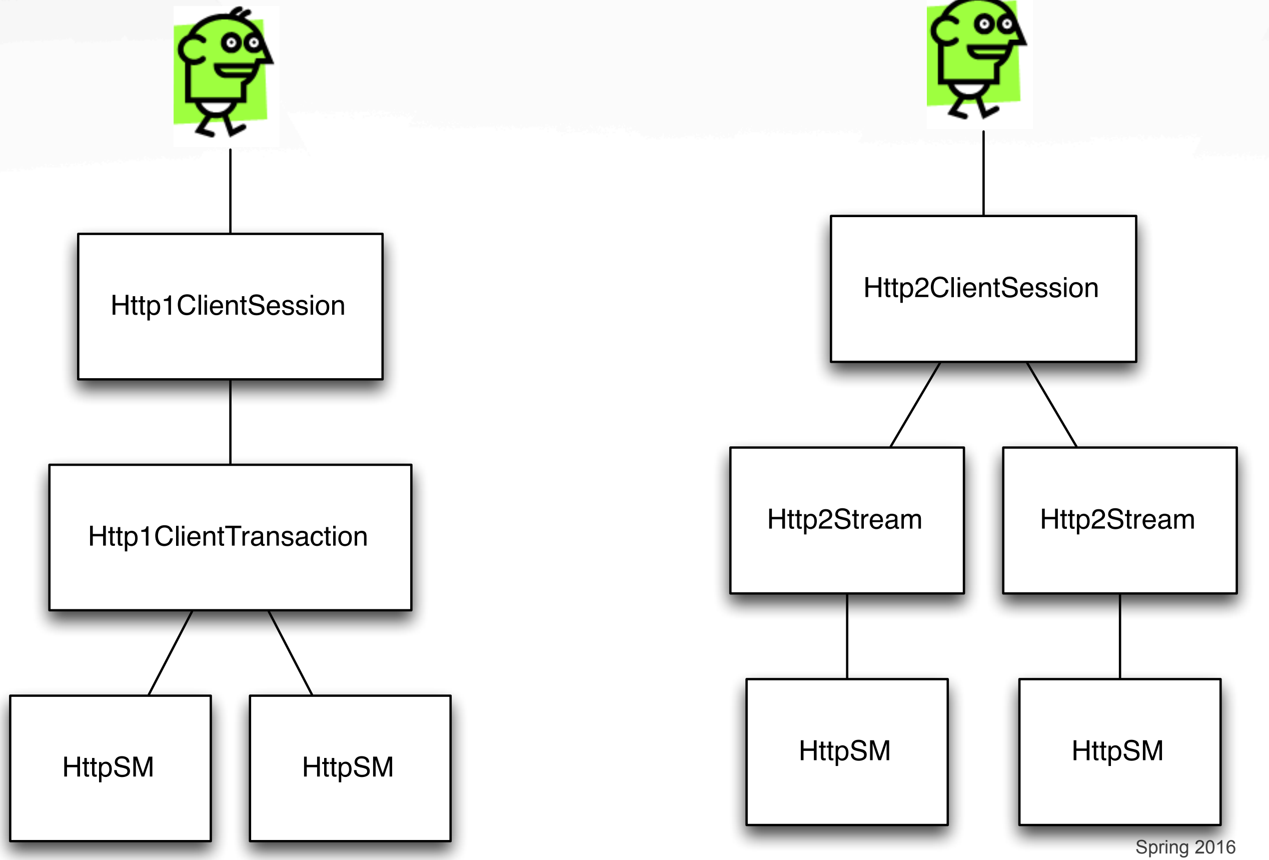
Session Class Diagram



Transaction Class Diagram



Object creation for two GETs over the same session



```
▪class ProxyClientTransaction : public Vconnection {
public:
  virtual void new_transaction();
  virtual NetVConnection get_netvc() const;
  virtual void set_active_timeout(ink_hrtime timeout_in) = 0;
  virtual void set_inactivity_timeout(ink_hrtime timeout_in) = 0;
  virtual void cancel_inactivity_timeout() = 0;
  virtual void attach_server_session(HttpServerSession *ssession, bool
transaction_done = true);
  int  get_transact_count() const;
  bool is_transparent_passthrough_allowed();
  void set_half_close_flag(bool flag);
  bool get_half_close_flag();
  const AclRecord *get_acl_record() const;
  virtual void release(IOBufferReader *r);
  virtual bool ignore_keep_alive() { return true; }
  virtual void destroy();
  ProxyClientSession *get_parent();
  virtual void set_parent(ProxyClientSession *new_parent);
  virtual bool allow_half_open() const = 0;
  virtual const char * get_protocol_string() const = 0;
```

Current Status

- Developed from November 2015 (last summit) through now.
- Bucket tested in Production starting in February
  - Additional early testing from Yahoo Japan
  - Performance similar on cache-heavy installation
  - Performance better on proxy-heavy installation
- Rolled into Yahoo mainline build a couple weeks back
- Committed to open source for 6.2

Development Challenges

- Much of the initial pain was figuring out how to offer the do_io_write/ do_io_read interfaces in Http2Stream to the HttpSM object
  - Previously got through FetchSM
  - Thinking backwards through the interface.
    - Very used to using do_io_read/do_write. Not used to thinking about providing that service.

Development Challenges

- Shutting down safely
  - How to deal with events in flight when you get the signal to shutdown
  - Assuming all network events will be handled on the same thread
  - On shutdown send around a shutdown event
    - Place no new events on the event queue after the shutdown event has been sent
    - Flushes the event queue
  - Problem of server communication showing up on different thread.
    - A read ready placed on another threads event queue breaks the shutdown flush assumption
    - Replaced many schedule to thread pool types to schedule on specific thread.

Development Challenges

- InkContInternal leaks
  - After the restructuring, the m_event_count of some InkContInternal objects were not being incremented/decremented appropriately
  - The introduction of Warning messages to track down.
- State Machine and other leaks
  - Reactivated the debug_sm_list
  - Ran for a bit and then poked around in the objects in the debugger.

Development Challenges

- Protocol specific timeouts
  - Debugging issue towards the end. Increase in number of timeouts. Reduced connection reuse.
  - State machine was setting timeouts directly on the UnixNetVC. Bypassing the Http2 level timeouts.
    - 30 second Http transaction inactivity timeout overriding 115 second Http2 inactivity timeout
    - HttpSM should not directly manipulate UnixNetVC except in cases like logging

Things Fixed So Far

- IPAllow enforced for HTTP/2
  - TS-3485
- Session Hooks and Transaction hooks trigger the correct number of times.
  - TS-1007
- Http connection stats correct regardless of HTTP version
  - TS-4136
- HTTP/2 should not trigger keep alive processing in HttpClientSession
  - TS-3584

Items Now More Fixable

- Logging network level information correctly for HTTP/2
  - TS-4012
- Plugin based protocol introspection
  - TS-4300

Expanding Server Session Modeling?

- Ultimately will want to support HTTP/2 to origin
  - A recurring request for us. Assume it will be attempted eventually.
- Will need to do a similar Transaction abstraction for Origin Server Communication
  - ProxyServerSession and ProxyServerTransaction
  - The session manager returns ProxyServerTransaction objects
  - How does ATS determine which protocol to initiate to the OS for new sessions?
- Can much of the current HTTP2 implementation be reused for a ATS acting as a HTTP2 client?