AMC's Tech Corner
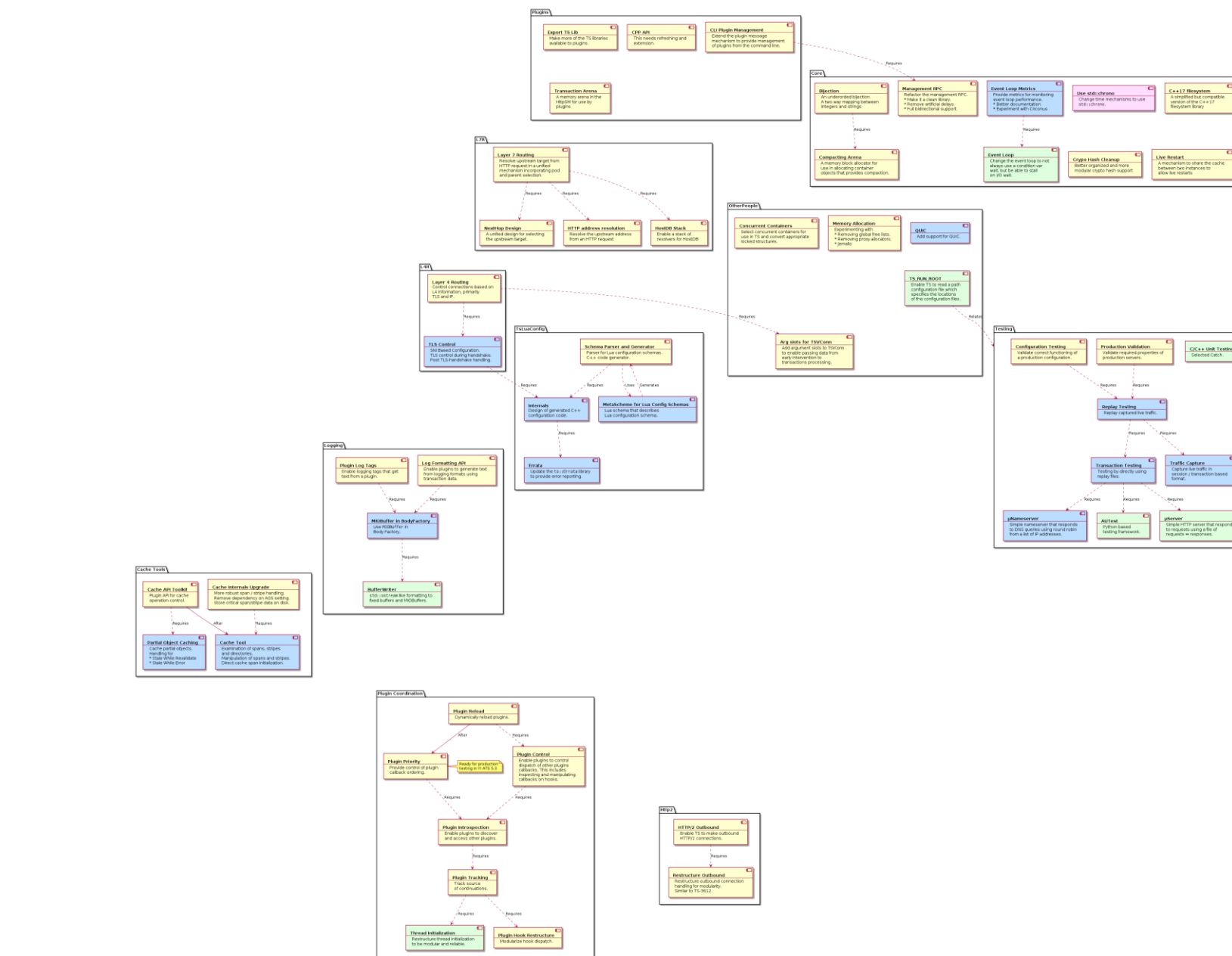
Long Range Planning

▶ Add functionality to address shortcomings.

▶ Frequently missing features that are just not quite there.

    ▶ Need to find these and build them out to usability.

▶ Need to stay competitive.

Layer 7 Routing

- Route based on layer 7 data – the HTTP request.

- Fundamentally: generate stream of IP addresses based on HTTP request and configuration.

- Make HostDB/DNS modular and pluggable.

  - Make NextHop logic a module in the HostDB stack.

  - Enable load feedback / load balancing.

Layer 4 Routing

- This is a thing now apparently.

- Already moving in this direction with Persia's work on SNI remap / configuration.

- Need to be able to control
  - Inbound SSL negotiations (ALPN, cyphers, certificate verification)
  - Tunneling, upstream target.
  - Access.

- Potentially foreign protocol observers.

- Tunneled transactions still get TXN_START and TXN_CLOSE

- Need TSVConn arguments to make this work well.

Outbound Restructuring

- Restructure outbound classes and operations to parallel the inbound structures.
  - E.g. TS-3612.
  - ServerSession, ServerTransaction
- HTTP/2 outbound
  - Requires outbound restructuring.
  - HTTP/2 outbound streams become ServerTransaction instances.
  - Need configuration to control
    - Low water / minimum outbound HTTP/2 to an upstream.
    - Load balancing / maximum streams per outbound HTTP/2.

Plugin Coordination

▶ A generalization of earlier plugin priority work.

▶ Goal: control of callback dispatch, ordering and enabling.

▶ Phases:

1. Thread initialization via generic continuations. [DONE]

2. Hook dispatch modularization.

3. Plugin continuation tracking.

4. Plugin introspection.

5. Hook introspection and control – priorities and/or partial ordering.

6. Configuration (static ordering control).

Plugin Coordination

- Plugin tracking should make plugin debugging much easier.
  - Core error messages can specify which plugin.
  - E.g. "continuation used after deleted"
- Long term – plugin reload.
  - Old plugin can be marked 'dead' and events for continuations from that plugin discarded by the event loop.

Cache : Partial Object Caching

- Chunked caching.

- Hopefully better performance due to less lock contention, no flying Dutchman retries.

- Stale while revalidate.

  - This required changes in the basic live cache object structures.

  - Now there are "slices", time based versions of alternates.

  - Only latest slice is written to disk, others are discarded when the object becomes inactive.

  - Also needed for handling server based updates to partially cached objects.

- Thundering herd mitigation.

  - May have some performance impact, need to investigate.

Cache Tool

- Command line access to cache information.
  - Replaces internal 'stripe inspector'.
  - Spans, stripes, directory data.
  - Detect unused spans.
- Manipulation of storage setup.
  - Clear spans, stripes.
  - Allocate stripes from spans.
- Updates to cache storage structure required for stripe management.
  - Want to change stripe size, directory with minimal cache data loss.

Cache Storage Updates

- Store stripe metadata sizing information.
  - Currently recomputed from scratch every process start based on configuration.
  - Fragile and unchangeable.
- Exploring ways to make it forward & backward compatible.
  - Initially store extra data as extra.
  - Cleared stripes will new updated stripe header with data directly built in.

Technical Advances

Baby steps forward

Forwarded Header
(Walt Karas)

▶ RFC-7923 compliant implementation of Forwarded header in core.

▶ Replaces X-Forwarded-For.

▶ Very configurable.

    ▶ Each value can be enabled/disabled independently.

    ▶ Extension to track protocol or protocol stack on inbound connection.

    ▶ Host can be configured as with Via.

Event Loop Changes (w/Fei Deng)

- Finally committed (thanks OKNet!)

- Changes event loop so that it can stall on I/O instead of a condition variable.

- By default maintains previous behavior.

  - Changed for ET_NET threads to wait on sockets.

- Run in production at Yahoo! for over two years.

- Major reduction in latency for rapid, tiny transaction processing.

  - 99% latency under 20ms, average under 6ms.

- Metrics added to track event loop performance.

  - Watch for outliers.

BufferWriter
(Walt Karas)

▶ Type safe C++ stream style writing to fixed buffers.

▶ No more
```
memcpy(str, field, strlen(field));
str += strlen(field);
```

▶ Buffer size checks on every write.

▶ Easily add formatting for classes.

▶ Code is cleaner and more robust without loss of performance.

▶ Time to talk about replacing Debug() with something better.

header_rewrite INBOUND

▶ The header_rewrite plugin now has a set of "INBOUND" conditions and substitutions.

▶ Data about the inbound connection.

▶ Much more reliable for detecting SSL vs. plain text connections.

▶ Detection of HTTP/2 vs. HTTP/1.

▶ Some day will do "OUTBOUND" too.

TextView

- Replaces `StringView` – named changed to avoid confusion.

- Subclass of `string_view` and therefore has those methods.

- Same footprint as `string_view` and just as fast to construct / copy.

- Interchangeable – conversion to and assignment from `string_view`.

- Used when contents of the view need to be examined / parsed.

- Easy replacement for `strtok` and Tokenizer.

  - No modification of source string.

  - No memory allocation.
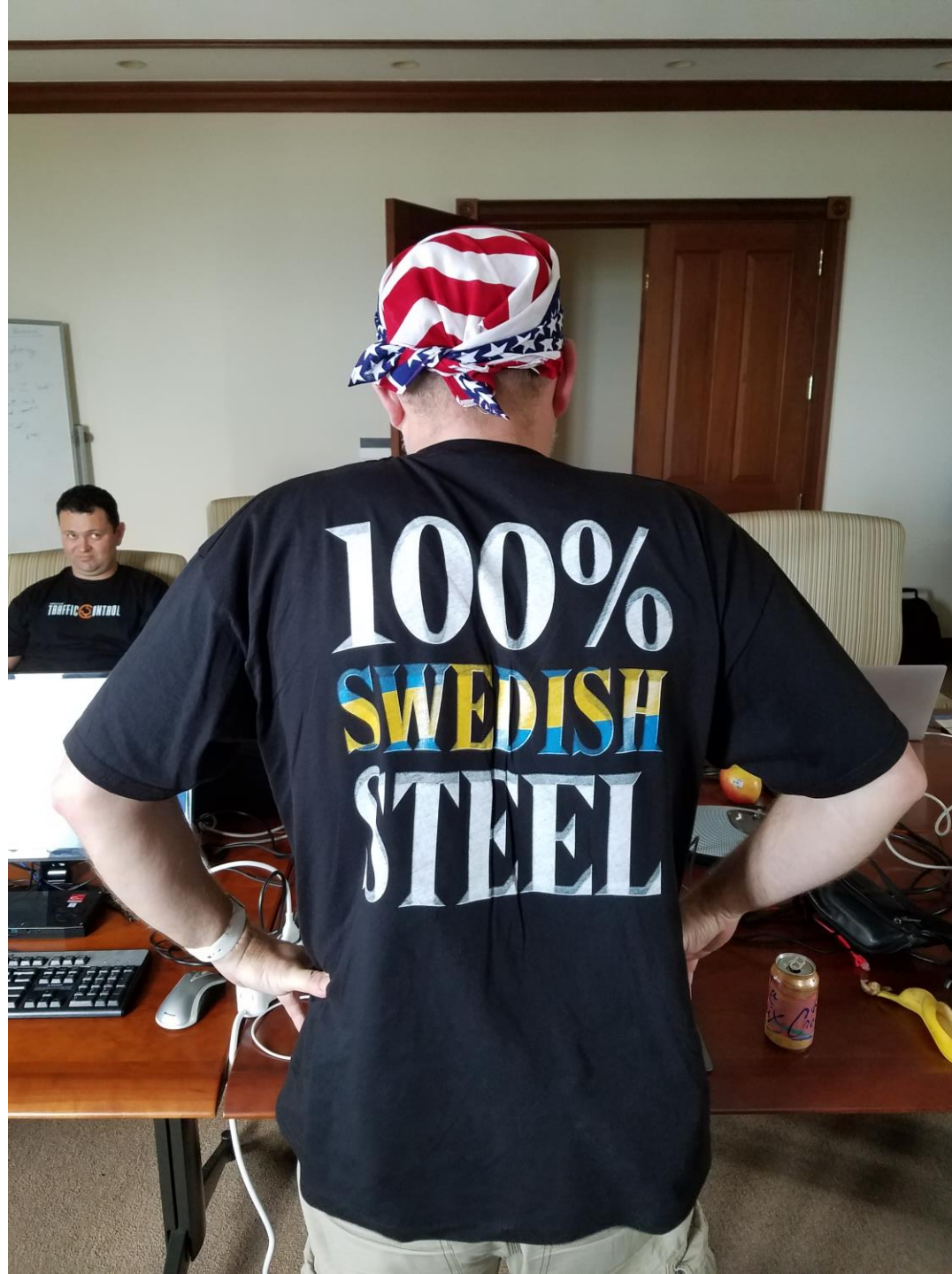
Briefings.

C++ Eleventy

C++ Eleventy
Guide for Contributors

▶ I talked about C++ techniques at two previous summits, read the slides.

▶ Containers

  ▶ Current only std::vector is approved for general use.

  ▶ The concern is excessive and uncontrolled memory allocation. I have seen Traffic Server deployments brought low by exactly from plugins.

  ▶ The issue cannot be entirely solved via better allocators. Fixed sized buffers or pre-allocated memory will always be faster than allocated memory.

  ▶ There is ongoing work in this area, but it proceeds slowly.

Parsimony

There is a general concern about code bloat and much effort has been made in the past to decrease   the over all code size. For what Traffic Server can do it has a remarkably small code base. This is not an accident.

For this reason think twice, then reconsider again, if you really need to add a support class. Avoid this if you can, especially if the class is used only once or in limited circumstances. Also avoid putting classes in separate files if the class isn't used in other file scopes. See if there are existing classes or other techniques which, with a bit of work, could be made to suffice.

If the support class is really needed, see if it can be made a bit more general to cover other uses.

Separability

Pull requests should be as small as possible. This yields the important result that isolatingchanges that cause problems and recovering from them is much easier. This means that if new classes / technology / support is required for a pull request, it is almost always better to contribute it in a separate, prior pull request. Pull requests that have a lot of new classes and other things have a very difficult time, both because the additional support obscures the realpoint of the pull request and that discussions can get bogged down on details in the support that aren't critical for the overall work.

Consistency

Traffic Server is an old project with people who have been working on it for over a decade. They are comfortable with where things are and how things are done. This doesn't mean it's perfect but unless there is a clear, strong reason to change, you should peruse the code and try to do things in a similar way.

One example would be initialization. There are various ways to do this in C++, mostof which result in the same machine code. Therefore it's a reasonable expectation that you will do this in the same way as the existing code. As a newcomer to the community, try to show a bit of respect for existing customs and habits, *especially* when such things have no real impact on code quality.

* Although we now recommend using braces for initialization as much as possible. But there is a strong reason for that.

Documentation

Documentation is critical to a successful project. Even the best technology will be abandoned if users, administrators, and contributors cannot understand it.

For this reason a pull request may be blocked if it adds new features without documentation for those features. For instance, a new configuration variable must always be accompanied by documentation for that variable.

Cache Write Lock

- No more missed cache write lock.

- Essential reason is cache writes must now create the alternate slice at the start of the write, not at write completion.