Origin connection issues: Dead, down, retry

Susan Hinrichs
ATS Summit June 2021

Origin connection retry

If a request to origin fails, ATS will retry

- Controlled by `proxy.config.http.connect_attempts_max_retries`
- ATS will always try once, and if that fails retry the number of times specified in the setting.

When does ATS retry the request?

- If ATS has sent no data to the origin (e.g. the TLS connection failed)
- If the request is idempotent (e.g. a GET but not a POST)
  - What if ATS has received data back?

Marking a server dead or down

First a philosophical discussion. Should we be talking about dead servers or down servers? Is there a difference? We seem to use the terms interchangeably

Dave Carlin filed https://github.com/apache/trafficserver/issues/7283

```
-bash-4.2$ traffic_ctl config match dead
NOTE: using the environment variable TS_RUNROOT
proxy.config.http.connect_attempts_max_retries_dead_server: 1
proxy.config.http.connect.dead.policy: 2
-bash-4.2$ traffic_ctl config match down
NOTE: using the environment variable TS_RUNROOT
proxy.config.http.down_server.cache_time: 180
proxy.config.http.parent_proxy.mark_down_hostdb: 0
```

How does a server get marked dead/down

Only considering the non-parent path. The parent path may be similar. (I hope.)

Only connection failures contribute to the down count. Once data has been sent, it is no longer a connection failure.

- TCP failure
  - Sent SYN but got no SYN-ACK (timeout)
  - Sent SYN but got immediate RST or ICMP error
- TLS Failure
  - Bad handshake packets
  - Failure to validate certificate on ATS or origin side
  - Failure to negotiate common protocols

Refining failures for deadness/downness

Particularly in cases where the origin handles multiple services, a bad certificate for one service should not cause the whole origin to be marked dead/down

In master (and hopefully 9.1) PR #7757 added `proxy.config.http.connect.dead.policy` to add some control for the kind of failures that count for dead/down

- 2 - Default original behavior. Both TLS and TCP failures count for deadness
- 1 - Only TCP failures count for deadness

How many times does a connection fail before the server is marked dead/down?

Currently `proxy.config.http.connect_attempts_max_retries` does double duty. Tracking both the number of connect retries, and the number of total failures it takes for a server to be marked dead/down

Should probably have a separate setting. May want to have more failure attempts than a single transaction before marking a server dead/down

Until PR #7291 landed (master and 9.1), a single failure with a non-retryable post would get a server address marked down.

What does it mean for a server to be dead/down?

`proxy.config.http.connect_attempts_max_retries_dead_server` takes effect during the dead time
A server stays dead/down until `proxy.config.http.down_server.cache_time` seconds have passed
Before Bryan Call landed PR #7142 (master/9.0), ATS would always try a dead origin at least once (even if `connect_attempts_max_retries_dead_server` was 0)
Now if `connect_attempts_max_retries_dead_server` is 0, the origin address is not tried at all during the dead period.

- This change in behaviour caused some exciting outages for us which led to the motivation for the `proxy.config.http.connect.dead.policy`

What does it mean for a server to be dead/down?

Moving forward, Alan's HostDB rework PR #7874 will never retry during the dead period.

- There is a zombie period after the dead time expires
- One origin to that address is tried at a time until a successful connection is made
- The `connect_attempts_max_retries_dead_server` will be used during the zombie period after the dead period where ATS is determining whether the origin is alive.

One more setting affecting retries

If there are multiple IP addresses for a origin, `proxy.config.http.connect_attempts_rr_retries` also impacts the connection retries.

- The counter for `connect_attempts_rr_retries` applies per address
- If connect_attempts_rr_retries is >= `connect_attempts_max_retries` then only one address will be tried.

PR #7288 (9.0.1) actually fixes the retry of the the first address.
This also prints up a wording choice for retry.

- For `connect_attempts_max_retries`, it is the number of retries beyond the initial attempt
- For `connect_attempts_rr_retries` it is the total number of tries for an address before moving to the next one.

Next steps

- What does this all mean on the parent path?
- Do we want to review/refine our terminology moving forward?
- Moving to Alan’s Layer 7 working group talk.