Quick question: Currently the API abstracts most of the state machine processing of the protocol handling.

Reading through some of these comments it seems like you are looking for something else. Instead, should the API just be an encoder/decoder for the protocol and leave the FSM up to the user?
Should it be more like sendMessageXXX, receiveMessageXXX, etc. - and all of the logic and statefulness would be handled by the user/developer?

Or, is the current abstracted interface the correct API for a developer? In this case an application developer could handle some higher level state within their application. But, the bulk of the transactional state machine would be handled under the API.

Let's get clear on this.

Tim's version of the question above:

The current API and implementation are more of what would be considered a controller pub/sub API that implements MOQT.
It is not an MOQT protocol API. A MOQT protocol API would look different and would move most of the FSM to the application.
For example, the application would receive all MOQT protocols messages and would be able to send all protocol messages.
The API would perform the encoding/decoding and QUIC session state handling.

==============

Cullen Jennings • Some notes on code review of new API. I probably need to spend more time looking at .h files but from first pass over doxygen files.
Few thoughts at casual read. Some of my comments are probably wrong so help me understand but I do think there are also probably a bunch of things

that need some changes.

1.) I don't get the need for both MoQClient and MoQClientDelegate because it seems there is always 1:1 binding between the two instances. Could we just put what is in the delegate into the Client class? Similar to what you did with Publish and Subscriber handler classes.

[Issue: #178]
Combine into 1…will do (Tim)

2.) I would like to get rid of cantina::LoggerPointer, is there some SpdLog would not work ? That would get rid of passing in a logger and make it easier for users of the library to use different logging strategies like syslog, files, or MoQ.

Issue: #178
Remove the logger - but get an idea how much time to replace (WILL DO - Tomas will look) -

3.) On Client config, I think we should pass is a moqt URL which will have the host, port, and other path information needed for the path parameters. I think the config also needs the role. Seems like we could get rid of the TransportProtocol.

I like the URL idea. We've talked about this in the past. (Will look at - resolving is an issue. Looks simple - and maybe we overloaded method. But, what do we do about DNS responses?. Tim/Scott/Mo).
- Mo says - MOQT has a path part of a URI. Where is this defined? May all go away - like a lot of other things in the changing MOQT drafts.

- Will have to dig into what this defined to now and what it might be defined like later…
- Currently our server has no notion of "path".
- This should be applied to the setup message.
- Role - we only support bi-directional today. Is there any reason for us to support any other role? Mo - says don't do anything with it. It may go away.
- MOQT is changing - will need to follow to see what changes are required.

4.) I don't understand endpoint_id and think we should get rid of this because no reasonable way to generate it. If we want to use something for metrics, we should have an way for app to generate it or get it like generate from the MAC. Seems like transport_config should be private or note here at all.

- MAC is a PITA. But, is a fair point for 'general' use. Since this is about tracking for metrics, might not need for general use. (Will discuss what to do here - Scott/Tim)
- Will probably change the name of the variable to something like "{somekindofkey}".

5.) Seems wrong that metrics_export is includes in moqt_config.h - we certainly want people to be able to build an app without use of metrics.

Issue: #178
Tomas can take a look. Rich will look at automating not including includes that are not used.

6.) I remain vary confused about the in Client. Run makes me thing of something that never returns while it is running. Perhaps connect would be a better name. I also don't understand why the constructor does not just call this.

Issue: #178
Another method name to make this more clear might help.

- Connect and disconnect might make sense.
- (Scott) Do not agree that the constructor should do this. Performing operations that might (or are very likely to) fail in the constructor is wrong.

7.) Seems like client needs a method to get the current status

The base class has a public method: Status status();

In the client callbacks:

8.) How about renaming connectionStatus to statusChanged(). I don't see why it needs conn_id, endpoint_Id, or TranportStatus. It should just have the pointer to the client it is for. It seems like connectionStatus and serverSetup could all be wrapped up in a single statusChanged callback.

When client and client callbacks are combined (as per 1 - above) there is no need to pass client - since the callback is a member of the client object.

- We discussed possible name changes for callbacks. Refer to 13 below.

- For server api the conn_id is required. Can remove from client API.
- Connections are treated separately from “server” status - a single handler could handle all connections and needs a way to differentiate.

- endpoint_id is a mistake and will be removed - this is part of server setup and client setup

- TransportStatus - is required - it indicates the current “status” that changed. All events callbacks include the event as an argument of the callback.

This seems like you would like to use the call back as a trigger -without any data. And that it is up to the client to access the current state of the “client” connection. I (Scott) consider this problematic if you as an application API user are keeping track of a connection FSM and the FSM is defined to go from CONNECTING, CONNECTED, DISCONNECTED, etc. - there may be glare condition between receiving the trigger and checking the current stats of the object. The connection may have skipped CONNECTING and seeming jumped to CONNECTION - because that was the state when you checked. But, if you were sent each status as a parameter of the callback the states would be seen as CONNECTING then CONNECTED.

We covered some more topics when reviewing this comment:

- will have working meeting to go through status/state changes.

- also include in the discussion what the application should do about re-connecting - should the app have to handle it or have it be “automatic”

- go away handling - TBD - beginning of connection, or mid-call, etc.

- do we want to support connect, disconnect and connect again? Or do we required a totally new client? Much easier to do the latter.

9.) The statusChanged stuff needs to also deal with the states when there is a temporary disconnect due to network braking and it is reconnecting. It also need to deal with temporary gone due to goaway message from server moving the connection.

This we can take a look at.

- need to discuss how we want to handle the “go away”
  - app status change, app gets info on where to go to, app tears down the current client and creates a new one
  - or - the apps sets a follow “go away” and the client automatically handles the re-connect. But, the client will have to tear down “handlers” and then restart them.
  - Do we minimize “hiding” state from the app?

10.) As a client using this,, the current status messages don't seem quite right. Perhaps we could outline the state machine then make sure we have a status for each state the software using the client needs to know. I think it need to be design from the what the app using the library need to do something different and then make sure the status gives the info to do that. I am sure there will be lots of changes in status info as we get more errors in the defined in the protocol

Issue: #178
Can look at some examples. Will document state machines related to status changes. (Tim/Scott)

11.) There does not seem to be a way to announce a namespace separate from knowing what tacks will be published. I think the API should allowing an announce of namespace then get subscribes for full track names. (perhaps there is a way to do this and I am just missing it)

Issue: #178
Sure - (Tim) - add a new method to send an announce.

12.) I think I lean towards adding a method on the client to send zero or more announce plus way of monitoring responses and then on server ways of receiving this but perhaps it needs a whole AnouceHandler class. Part of this depends on if you think a server can ever send an Annouce or not. Thoughts on that.

Let's discuss - MOQT future versions might need this on the server. (Tim/Scott/Mo)

- client already supports "READY" - announceOK triggers this status
- should have auth added.
- today - relay does use this (Peering is different)
- There is an announce callback on the server side

- We can add client without a subscribe. *Today this will cause an error to be reported reported on the track. But, the app could ignore that error.

13.) for all the statusCallback, I think a better name would be statusChangedCallback.

Fine. Or - what about using "statusChanged"?

- rule - xxxChanged for status events (e.g. statusChanged, connectionChanged, etc.)
- rule - xxxReceived for message/object/etc. - (e.g. objectReceived, announceReceived, etc.)

14.) Similar with congestedCallback , I would go with name more like congestionChaangedCallback because I assume get this both when something gets congested and when it stops being congested. We should also be able to see the congestion status with something like "bool isCongested"

Will could change to match above rule (“congestionChanged”).

-there are a wide variety of transmission states we should consider and maybe coalesce multiple possible callbacks. We should discuss.

- for now we can remove the congested callback until we figure out how we want to handle the information needed by the client to adjust frame-rate/bit-rate/etc.

15.) Ton of this stuff probably needs const added to it.

Issue: #178
Tomas/Tim/etc. - Rich and Tim are looking at automating some of this.

16.) I did not look at any of the protected stuff but at casual glance it looked like some of it needs to be public for the API to work.

Keep as is for now. Will see later when get further along what needs to be exposed.

17.) It seems like the PUblichTrackHandler (and subscribe handler) need a MoqTClient or MoqTRServer passed into the constructor to know what connection to use.

[Issue: #178]
There is no need to pass the client/server to handlers in a constructor for a handler.

This is problematic.

Instead there will be a mechanism to pass access to the client/server object as part of a callback.

The mechanics are being worked.

18.) I don't understand why the client inherits from the transportDelegate but did not dig into understanding this as does not seem part of public part of API.

The ClientCore inherits and implements the transportDelegate callbacks.

- But, it does seem worthwhile to merge transport directly into “libquicr” (name to be changed).
- This would resolve this inherited dependency.

19.) On Publish API, we nee to deal with errors of trying to publish to an non authorized space or a space that was non announced.
I don't think the TrackMode can be changed once the publish is started so I think we should remove the setTrackMode - I could easily be convinced I was wrong on this.

[Issue: #178]
Sure - we can remove “setTrackMode”.

20.) I think we should have an Object class that has a ref to full track name, group id , object id , priority, ttls and the data. Feel pretty strongly about this as this is how the application is going to want to handle the data in many cases and it allows us to extend the object data model without breaking all the APIs.

[Issue: #178]

Just checking - is this just to not have all of the overloads for publishing an object and instead fill in the fields in a class/struct?

We will look into this.

21.) I think we need a FullTrackName class that can hold the Namespace, TrackName and optional track alias. Again feel pretty strongly about this. This will allow us to do things like change the Namespace to an array of tuples without changing all the APIs which is a current proposal.

[Issue: #178]
We can dig into this. There is struct “TrackFullName” - with Namespace and Name. But, we can expand with tuples, etc. But, yep - we can rename the current struct.

22.) More disctution needed on all the Error enums and status enums.

Agreed

23.) TrackSubscribneHandler seems better name to me than SubscirbeTrackHanlder - same for publisher.

[Issue: #178]
Okay

24.) I would be fine to change the namespace to quicr-MoQT (or whatever naming convention is) but I don't think I like the MOQT at start of all things.

Okay - I think. Just shifting the name. “namespace quicr::moqt”

- we should remove ‘quicr’

- instead shouldn’t it be “namespace moq::transport”?
- src files will be to a directory ./moq/transport?

25.) Seems like Server and SeverCallbacks could be merged into one class.

[Issue: #178]
Yep - just like client.

26.) Get rid of Cantina logger.

We know - from 2. above.

27.) So when the server get a new connection. It seems like it should create some object that represents a Connection to the client. I think we need a new class for this. However, most the callback can be on that.

We will take a look. Current the ID is called and that is enough to keep track of the data you want to track based on the connection.

- But, we can look at this complication. This will required a templated factory for constructing the ‘handler’.

28.) Naming of callback in server callbacks seems not aligned with other ones.

[Issue: #178]
Will take a look.

29.) The announce Verify and announce ok callback seem like the wrong abstraction. Walk me thought this.

Sure - we can review. We can write out the thinking here.

30.) This all seems like it needs ServerAnnouceHandler class, ServerSubscribeHandler class, ServerPublishHandler class.

SubscribeHandler (not used by Server) and PublishHandler are already defined. Currently there doesn't seem to be a need for a ServerAnnouceHandler. All callbacks are already part of the handler without the need for a separate AnnounceHandler.

31.) A relay would use a Subscriber class when it acted as a client and subscribed to upstream relay. But it would use a ServerSubscribe class when it received a subscribe from a downstream client. I think we need some refactoring then can comment on actual methods in those classes.

Yes - to the first part - Relay subscribes just like a client.

- When the relay receives a subscribe. The relay ultimately published to the subscribe. So, that is why the current implement uses a PublisherHandler. This has been discussed between Tim and Mo.

32.) In TrackFullName. I think this should be a class not a struct and call FullTrackName. the name element should be trackName not name just to line up with spec. It should have an optional trackAlias.

[Issue: #178]
Okay.

33.) I am super negative on the TrackHash. I think this represents a confusion about what TrackAlias is for. I would like to work through how we use then then get rid of it. I don't think it will work for large scale CDNs.

If TrackAlias can be global then fine. There are other ways to make this unique. Or, we could just use the entire vector - but performance on matching may not be performant.

34.) I think I would have preferred to provide this review as a github review on a PR for the .h files.

Sure.