Updates for Message Framers

Tommy Pauly
TAPS
IETF 105, July 2019, Montreal

Status Updates

Merged PR for framer updates

https://github.com/ietf-tapswg/api-drafts/pull/323

Some discussion items remain open

Framing API made available in iOS/macOS beta

https://developer.apple.com/documentation/network/nwprotocolframer

Adding Message Framers

First, create a Message Framer

```
framer := NewMessageFramer()
```

Add the Message Framer to a preconnection; last added runs “closest” to application

```
Preconnection.AddFramer(framer)
```

Issue #340: Clarify ordering of events for Start/Stop with layered framers

Framer Metadata

Add framer-specific values to message context when sending

```
messageContext := NewMessageContext()
messageContext.add(framer, key, value)
Connection.Send(messageData, messageContext)
```

Access framer-specific values when receiving

```
messageContext.get(framer, key) -> value
```

Message Framer Lifetime

Events are delivered to indicate a framer becoming active or inactive on a given Connection

```
MessageFramer -> Start(Connection)
MessageFramer -> Stop(Connection)
```

Functions can be called relative to a specific Connection instance to become Ready or Closed

```
MessageFramer.MakeConnectionReady(Connection)
MessageFramer.FailConnection(Connection, Error)
```

Issue #340: Need to be able to close a connection without an error

Datapath Handling

Sending frames can be reduced to be very simple

```
MessageFramer -> NewSentMessage<...>

MessageFramer.Send(Conn, Data)
```

Receiving is an event loop of parsing and delivering

```
MessageFramer -> HandleReceivedData<Conn>

MessageFramer.Parse(...)
MessageFramer.AdvanceReceiveCursor(Conn, Length)
MessageFramer.Deliver(Conn, Context, Data, EOM)
```

Questions Raised

PR #323

Are Message Framers “protocols”?

How much should the Interface document refer to “objects” for concepts like Message Framers? Can we clarify the object model?

Is there a clearer way to refer to storage of metadata for Message Framers? Can we unify this with other protocol metadata?

Examples

Framer Setup

```
// Add your custom game protocol to support game messages.
let gameOptions = NWProtocolFramer.Options(definition: GameProtocol.definition)
self.defaultProtocolStack.applicationProtocols.insert(gameOptions, at: 0)
```

```
// Create a class that implements a framing protocol.
class GameProtocol: NWProtocolFramerImplementation {

    func handleOutput(...)

    func handleInput(...)

}
```

```
// Define a protocol header struct to help encode and decode bytes.
struct GameProtocolHeader: Codable {
    let type: UInt32
    let length: UInt32
}
```

https://developer.apple.com/documentation/network/building_a_custom_peer-to-peer_protocol

Examples

Sending TLVs

```
// Whenever the application sends a message, add your protocol header
// and forward the bytes.
func handleOutput(framer: NWProtocolFramer.Instance, message: NWProtocolFramer.Message,
messageLength: Int, isComplete: Bool) {
    // Extract the type of message.
    let type = message.gameMessageType

    // Create a header using the type and length.
    let header = GameProtocolHeader(type: type.rawValue, length: UInt32(messageLength))

    // Write the header.
    framer.writeOutput(data: header.encodedData)

    // Ask the connection to insert the content of the application message after your
    // header.
    do {
        try framer.writeOutputNoCopy(length: messageLength)
    } catch let error {
        print("Hit error writing \(error)")
    }
}
```

https://developer.apple.com/documentation/network/building_a_custom_peer-to-peer_protocol

Examples

Parsing TLVs

```
// Whenever new bytes are available to read, try to parse out your message format.
func handleInput(framer: NWProtocolFramer.Instance) -> Int {
    while true {
        // Try to read out a single header.
        var tempHeader: GameProtocolHeader? = nil
        let headerSize = GameProtocolHeader.encodedSize
        let parsed = framer.parseInput(minimumIncompleteLength: headerSize,
                                       maximumLength: headerSize) { (buffer, isComplete) -> Int in
            tempHeader = GameProtocolHeader(buffer)
            return headerSize
        }

        // If you can't parse out a complete header, stop parsing and ask for headerSize more bytes.
        guard parsed, let header = tempHeader else {
            return headerSize
        }

        // Create an object to deliver the message.
        var messageType = GameMessageType.invalid
        if let parsedMessageType = GameMessageType(rawValue: header.type) {
            messageType = parsedMessageType
        }
        let message = NWProtocolFramer.Message(gameMessageType: messageType)

        // Deliver the body of the message, along with the message object.
        if !framer.deliverInputNoCopy(length: Int(header.length), message: message, isComplete: true) {
            return 0
        }
    }
}
```

https://developer.apple.com/documentation/network/building_a_custom_peer-to-peer_protocol