Actor Bug: Communication - Connection

Sometimes it is required for actors to form a connection, such as a data stream included in Akka's library, to share data among themselves beyond sending a message to an actors queue. This is seen in streaming services or file requests where pipes are often utilized.

A bug of this nature occurred in Colossus when it was reported that a client that should be disconnected would instead continuously try to reconnect to a service anyway. The problem was that if disconnect was called while the client was in its connecting state, there was no way for the system to process this change of state and so wouldn't recognize that it happened. This was fixed by changing the code to allow for this state of disconnecting before the attempted reconnect call.

This bug is highlighted in the code below. There are three main parts that needed work for this fix. For the Controller, a case needed to be considered when checkControllerGracefulDisconnect was called and the senders connection status was before an initial connection. By the same token, CoreHandler also needed a NotConnected case for shutdownRequests. The ServiceClient had a check on line 43 for the function attemptReconnect that caused the trouble with the continuous reconnect attempts. Without checking if the user had chosen to manually disconnect the reconnect option was constantly defaulted to instead. By adding this consideration of being disconnected before the initial connection, the program was able to handle this state and function appropriately.

Link: https://github.com/tumblr/colossus/issues/278

Code:

colossus/controller/Controller.scala

```
...
  private[controller] def checkControllerGracefulDisconnect() {
  (connectionState, inputState, outputState) match {
      case (ShuttingDown(endpoint), InputState.Terminated, OutputState.Terminated) => {
        super.shutdown()
      }
+     case (NotConnected, _, _) => {
+     //can happen when disconnect is called before being connected
+       super.shutdown()
+     }
      case _ => {}
     }
...
```

colossus/core/CoreHandler.scala

```
...
```

```
  final def shutdownRequest() {
    case Connected(endpoint) => {
      _connectionState = ShuttingDown(endpoint)
      shutdown()
    }
+   case NotConnected => shutdown()
    case _ => {}
  }
...
```

colossus/service/ServiceClient.scala

```
...
  private def attemptReconnect() {
    connectionAttempts += 1
-   if(!disconnecting) {
+     if(!disconnecting && !manuallyDisconnected) {
        if(canReconnect) {
…
```