RRC Connection Establishment Process Objective

Overview of RRC States-

In cellular networks, the RRC layer controls the control plane signalling between the network (e.g., eNodeB in LTE or gNodeB in 5G) and the UE (e.g., your smartphone). The UE may be in one of two main RRC states:

- **RRC_IDLE:** The UE is not currently communicating with the network. It listens to paging channels for calls or system messages but has no resources assigned to it. The UE is "asleep" in this state.
- **RRC_CONNECTED:** The UE is connected to the network with an active connection, and dedicated radio resources are reserved for data transfer (uplink and downlink).
- The RRC Connection Establishment process is how the UE makes the transition from RRC_IDLE to RRC_CONNECTED when it wishes to originate a communication (e.g., placing a call, sending data) or reply to a request from the network (e.g., incoming call).

Key Concepts

1. **RRC States**:
   - **RRC_IDLE**: No signalling connection, no dedicated bearers, UE tracks the network via cell reselection.
   - **RRC_CONNECTED**: Active signalling and data connection, UE is assigned a Cell Radio Network Temporary Identifier (C-RNTI).

2. **RRC Signalling Messages**:
   - **RRC Connection Request**: Sent by the UE to request a connection.
   - **RRC Connection Setup**: Sent by the network to configure the connection.
   - **RRC Connection Setup Complete**: Sent by the UE to confirm the connection.

3. **Objective**: Simulate this handshake and state transition.

Detailed Procedure

Let's break this down into steps:

Step 1: UE in RRC_IDLE State

- The UE is powered on and camped on a cell after performing cell selection/reselection.
- It listens to system information (e.g., Master Information Block, System Information Blocks) and paging messages.
- No dedicated resources are allocated yet.

Step 2: Trigger for Connection

- The UE decides to establish a connection, triggered by:
  - Mobile Originated (MO): User initiates a call or data session.
  - Mobile Terminated (MT): Network pages the UE for an incoming call.
- For this simulation, we'll assume an MO trigger (e.g., user opens an app).

Step 3: Random Access Procedure

- Before sending the RRC Connection Request, the UE performs the Random Access (RA) procedure to synchronize with the network and request resources:
  1. Preamble Transmission: UE sends a random access preamble on the Physical Random Access Channel (PRACH).
  2. Random Access Response (RAR): Network responds with timing advance (TA) and an uplink grant.
  3. This step ensures the UE is synchronized and has resources to send the RRC message.

Step 4: RRC Connection Request

- The UE sends an RRC Connection Request message to the network over Signaling Radio Bearer 0 (SRB0), which uses the Common Control Channel (CCCH).
- Contents:
    - UE Identity (e.g., S-TMSI or a random value).
    - Establishment Cause (e.g., MO-Data, MT-Access).

Step 5: RRC Connection Setup

- The network (eNodeB) processes the request, allocates a C-RNTI, and configures dedicated resources (e.g., SRB1 for further signaling).
- The network sends an RRC Connection Setup message back to the UE on SRB0.
- Contents:
    - C-RNTI.
    - Configuration for SRB1 (e.g., RLC and MAC parameters).

Step 6: RRC Connection Setup Complete

- The UE applies the configuration, transitions to RRC_CONNECTED, and sends an RRC Connection Setup Complete message on SRB1.
- Contents:
    - Confirmation of successful setup.
    - Optionally, NAS (Non-Access Stratum) messages (e.g., Attach Request).

Step 7: RRC_CONNECTED State

- The UE is now in RRC_CONNECTED with dedicated resources.
- Data transfer can begin (e.g., via Data Radio Bearers).

Python Simulation

Let’s simulate this process in Python. We’ll create classes for the UE and Network, and use a simple messaging system to mimic the signaling exchange. I’ll include code snippets for each step and then provide the full code.

Code Snippet 1: Define Classes and States

```
from enum import Enum
import time

# Define RRC States
class RRCState(Enum):
    IDLE = "RRC_IDLE"
    CONNECTED = "RRC_CONNECTED"

# UE class to simulate the device
class UE:
    def __init__(self, ue_id):
        self.ue_id = ue_id
        self.state = RRCState.IDLE
        self.c_rnti = None

    def initiate_connection(self, network):
        print(f"UE [{self.ue_id}] in {self.state.value}: Initiating connection...")
        network.receive_rrc_connection_request(self)

    def receive_rrc_connection_setup(self, c_rnti, config):
        print(f"UE [{self.ue_id}]: Received RRC Connection Setup, C-RNTI:
{c_rnti}")
        self.c_rnti = c_rnti
        self.state = RRCState.CONNECTED
        print(f"UE [{self.ue_id}]: Transitioned to {self.state.value}")
        return {"message": "RRC Connection Setup Complete", "ue_id":
self.ue_id}

# Network class to simulate eNodeB
class Network:
    def receive_rrc_connection_request(self, ue):
        print(f"Network: Received RRC Connection Request from UE
[{ue.ue_id}]")
        time.sleep(1)  # Simulate processing delay
        c_rnti = "C-RNTI-1234"  # Assign a temporary identifier
        config = {"SRB1": "Configured"}
        response = {"c_rnti": c_rnti, "config": config}
        ue.receive_rrc_connection_setup(response["c_rnti"], response["config"])
        self.receive_rrc_connection_setup_complete(ue)

    def receive_rrc_connection_setup_complete(self, ue):
        print(f"Network: Received RRC Connection Setup Complete from UE 
[{ue.ue_id}]")
        print(f"Network: UE [{ue.ue_id}] is now in {ue.state.value}")
```

Explanation:

- **RRCState**: Enum for RRC states.
- **UE**: Represents the device with an ID, state, and C-RNTI.
- **Network**: Simulates the eNodeB, handling requests and responses.
- time.sleep(1): Adds a delay to mimic real-world processing.

Code Snippet 2: Random Access Procedure

Let's add a simplified Random Access step:

```
class UE:
    def random_access(self, network):
        print(f"UE [{self.ue_id}]: Sending Random Access Preamble...")
        time.sleep(0.5)
        network.receive_preamble(self)

    def receive_rar(self, timing_advance, uplink_grant):
        print(f"UE [{self.ue_id}]: Received RAR - TA: {timing_advance}, 
Grant: {uplink_grant}")
        self.initiate_connection(network)

class Network:
    def receive_preamble(self, ue):
        print("Network: Received Random Access Preamble")
        time.sleep(0.5)
        self.send_rar(ue)

    def send_rar(self, ue):
        timing_advance = "1.2ms"
        uplink_grant = "Allocated"
        ue.receive_rar(timing_advance, uplink_grant)
```

Explanation:

- The UE sends a preamble, and the network responds with a Random Access Response (RAR).

Code Snippet 3: Main Simulation

```
def simulate_rrc_connection():
    ue = UE("UE-001")
    network = Network()
    print("Starting RRC Connection Establishment Simulation...")
    ue.random_access(network)

# Run the simulation
if __name__ == "__main__":
    simulate_rrc_connection()
```

Explanation:

- This ties everything together, starting with the Random Access procedure followed by the RRC signalling.

Complete Python Code

Here's the full program combining all steps:

```
from enum import Enum
import time

# Define RRC States
class RRCState(Enum):
    IDLE = "RRC_IDLE"
    CONNECTED = "RRC_CONNECTED"

# UE class to simulate the device
class UE:
    def __init__(self, ue_id):
        self.ue_id = ue_id
        self.state = RRCState.IDLE
        self.c_rnti = None

    def random_access(self, network):
        print(f"UE [{self.ue_id}] in {self.state.value}: Sending Random Access
Preamble...")
        time.sleep(0.5)
        network.receive_preamble(self)

    def receive_rar(self, timing_advance, uplink_grant):
        print(f"UE [{self.ue_id}]: Received RAR - TA: {timing_advance},
Grant: {uplink_grant}")
        self.initiate_connection(network)

    def initiate_connection(self, network):
        print(f"UE [{self.ue_id}]: Sending RRC Connection Request...")
        network.receive_rrc_connection_request(self)

    def receive_rrc_connection_setup(self, c_rnti, config):
        print(f"UE [{self.ue_id}]: Received RRC Connection Setup, C-RNTI:
{c_rnti}")
        self.c_rnti = c_rnti
        self.state = RRCState.CONNECTED
        print(f"UE [{self.ue_id}]: Transitioned to {self.state.value}")
        return {"message": "RRC Connection Setup Complete", "ue_id":
self.ue_id}

# Network class to simulate eNodeB
class Network:
    def receive_preamble(self, ue):
        print("Network: Received Random Access Preamble")
        time.sleep(0.5)
        self.send_rar(ue)

    def send_rar(self, ue):
        timing_advance = "1.2ms"
        uplink_grant = "Allocated"
        ue.receive_rar(timing_advance, uplink_grant)

    def receive_rrc_connection_request(self, ue):
        print(f"Network: Received RRC Connection Request from UE
[{ue.ue_id}]")
```

```
        time.sleep(1)  # Simulate processing delay
        c_rnti = "C-RNTI-1234"  # Assign a temporary identifier
        config = {"SRB1": "Configured"}
        response = {"c_rnti": c_rnti, "config": config}
        ue.receive_rrc_connection_setup(response["c_rnti"], response["config"])
        self.receive_rrc_connection_setup_complete(ue)

    def receive_rrc_connection_setup_complete(self, ue):
        print(f"Network: Received RRC Connection Setup Complete from UE
[{ue.ue_id}]")
        print(f"Network: UE [{ue.ue_id}] is now in {ue.state.value}")

# Main simulation function
def simulate_rrc_connection():
    ue = UE("UE-001")
    network = Network()
    print("Starting RRC Connection Establishment Simulation...")
    ue.random_access(network)

# Run the simulation
if __name__ == "__main__":
    simulate_rrc_connection()
```

Expected Output-

When you run this code, you'll see something like:

```
Starting RRC Connection Establishment Simulation...
UE [UE-001] in RRC_IDLE: Sending Random Access Preamble...
Network: Received Random Access Preamble
UE [UE-001]: Received RAR - TA: 1.2ms, Grant: Allocated
UE [UE-001]: Sending RRC Connection Request...
Network: Received RRC Connection Request from UE [UE-001]
UE [UE-001]: Received RRC Connection Setup, C-RNTI: C-RNTI-1234
UE [UE-001]: Transitioned to RRC_CONNECTED
Network: Received RRC Connection Setup Complete from UE [UE-001]
Network: UE [UE-001] is now in RRC_CONNECTED
```

Conclusion

This simulation demonstrates the **RRC Connection Establishment process**, covering:

- The transition from **RRC_IDLE** to **RRC_CONNECTED**.
- The Random Access procedure for synchronization and resource allocation.
- The exchange of RRC signalling messages (Request, Setup, Complete).

The Python code simplifies some aspects (e.g., no real radio resource allocation), but it captures the essence of the state transition and signaling flow. You can extend this by adding error handling (e.g., contention resolution in RA) or more detailed configurations. Let me know if you'd like to dive deeper into any part!