TEST 2: SMF PDU SESSION ESTABLISHMENT

REFERENCE STANDARD: 3GPP TS 23.502 ,TS 29.502

LAYER: 5G CORE (SMF)

DIFFICULTY LEVEL: INTERMEDIATE

SUBMITTED BY:

GROUP 3 - TEAM MEMBERS:

- ***55984_Vikas Srivastava***
- ***58622_Harshinie M***
- ***58623_Shreyash Bhatt***
- ***58624_Jayavarshini G***
- ***58635_Yuvaraj P***

TABLE OF CONTENTS

S.NO	DESCRIPTION	PAGE NO
1	Introduction to SMF & PDU Sessions	3
2	5G Core Architecture	4
3	PDU Session Establishment Signaling Flow	6
4	SMF State Machine	8
5	C++ Implementation	9
6	Sample Output	20
7	Key Concepts & 3GPP References	22
8	Compilation Guide	24
9	Conclusion	25

1. INTRODUCTION TO SMF & PDU SESSIONS IN 5G NR

The Session Management Function (SMF) is a key control-plane Network Function in the 5G Core (5GC) architecture. It is responsible for managing the complete lifecycle of PDU (Protocol Data Unit) Sessions, which provide end-to-end data connectivity between the User Equipment (UE) and external Data Networks (DNs) such as the Internet, IMS, and enterprise services.

A PDU Session represents a logical association between the UE and a specific Data Network, identified by a Data Network Name (DNN) and a Network Slice (S-NSSAI). The SMF establishes, modifies, and releases these sessions, while coordinating with other core network functions including the AMF (Access and Mobility Management Function), UPF (User Plane Function), PCF (Policy Control Function), and UDM (Unified Data Management).

Through its central role in session control, the SMF ensures proper IP address allocation, QoS enforcement, traffic steering, and seamless connectivity for 5G services across different slices and access technologies.

3GPP Reference

The procedures governing PDU Session establishment, modification, and release in the 5G System are specified in 3GPP TS 23.502, which defines the end-to-end control-plane signaling flows and interactions among core network functions such as AMF, SMF, and UPF. The service-based architecture and Session Management Function (SMF) operations are further detailed in 3GPP TS 29.502, which specifies the Service-Based Interface (SBI) APIs and service operations exposed and consumed by the SMF for PDU Session lifecycle management within the 5G Core network.

1.1 SMF Key Responsibilities

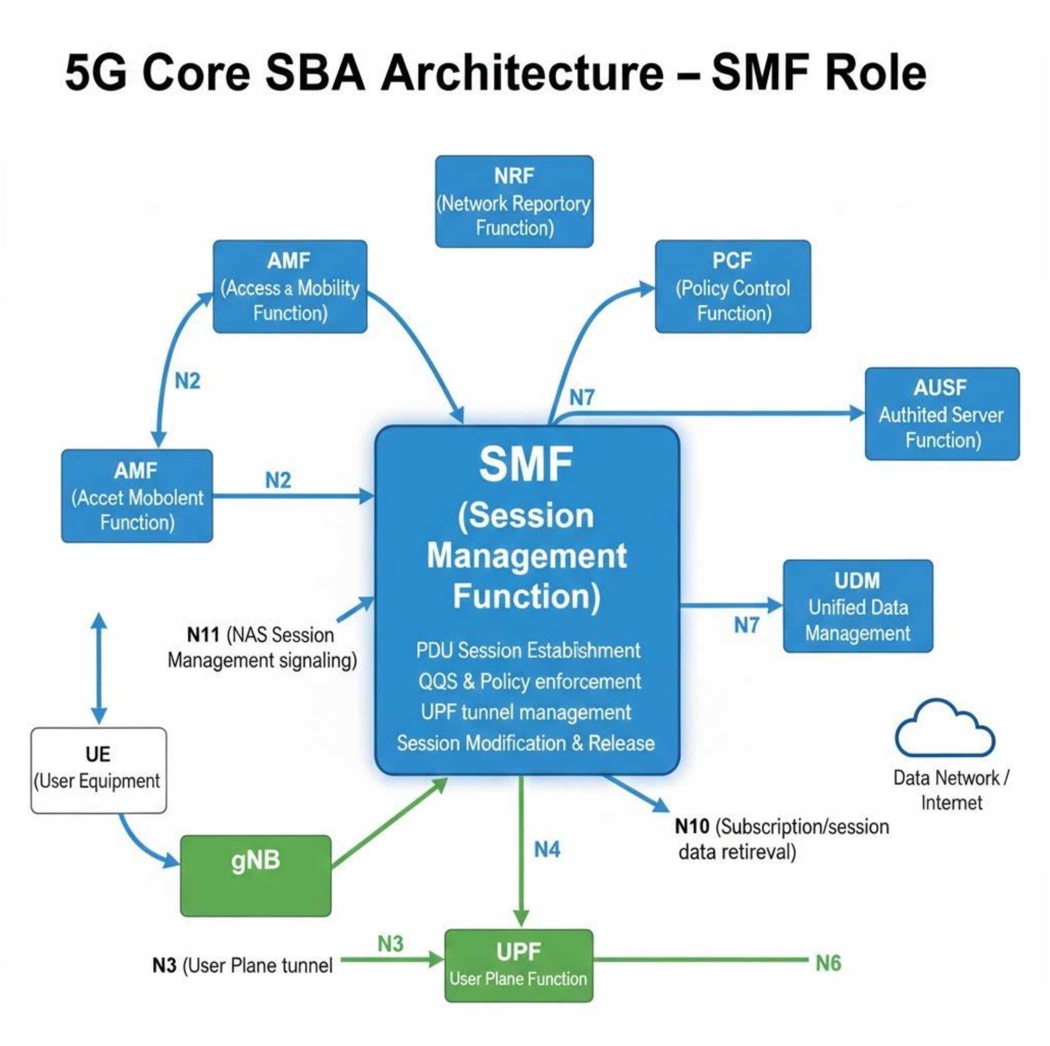
The Session Management Function (SMF) is responsible for the complete control-plane management of PDU Sessions in the 5G Core. It handles the entire session lifecycle from creation to release and ensures that user data flows are correctly established, controlled, and optimized. One of its primary responsibilities is PDU Session establishment, modification, and release, including the handling of UE requests and network-initiated changes. The SMF allocates and manages UE IP addresses (IPv4, IPv6, or IPv4v6) and selects the appropriate User Plane Function (UPF) based on the requested Data Network (DNN), Network Slice (S-NSSAI), UE location, and network topology.

The SMF also performs PFCP session management over the N4 interface, installing and updating Packet Detection Rules (PDRs), Forwarding Action Rules (FARs), QoS Enforcement Rules (QERs), and Usage Reporting Rules (URRs) in the UPF. It creates and manages QoS Flows, maps them to Data Radio Bearers, and enforces Session AMBR (Aggregate Maximum Bit Rate) limits. In addition, the SMF handles Downlink Data Notification (DDN) procedures, supports roaming scenarios using the V-SMF / H-SMF architecture, and integrates with charging, lawful intercept, and policy control functions to ensure compliant and optimized service delivery.

2. 5G Core Architecture - SMF Role

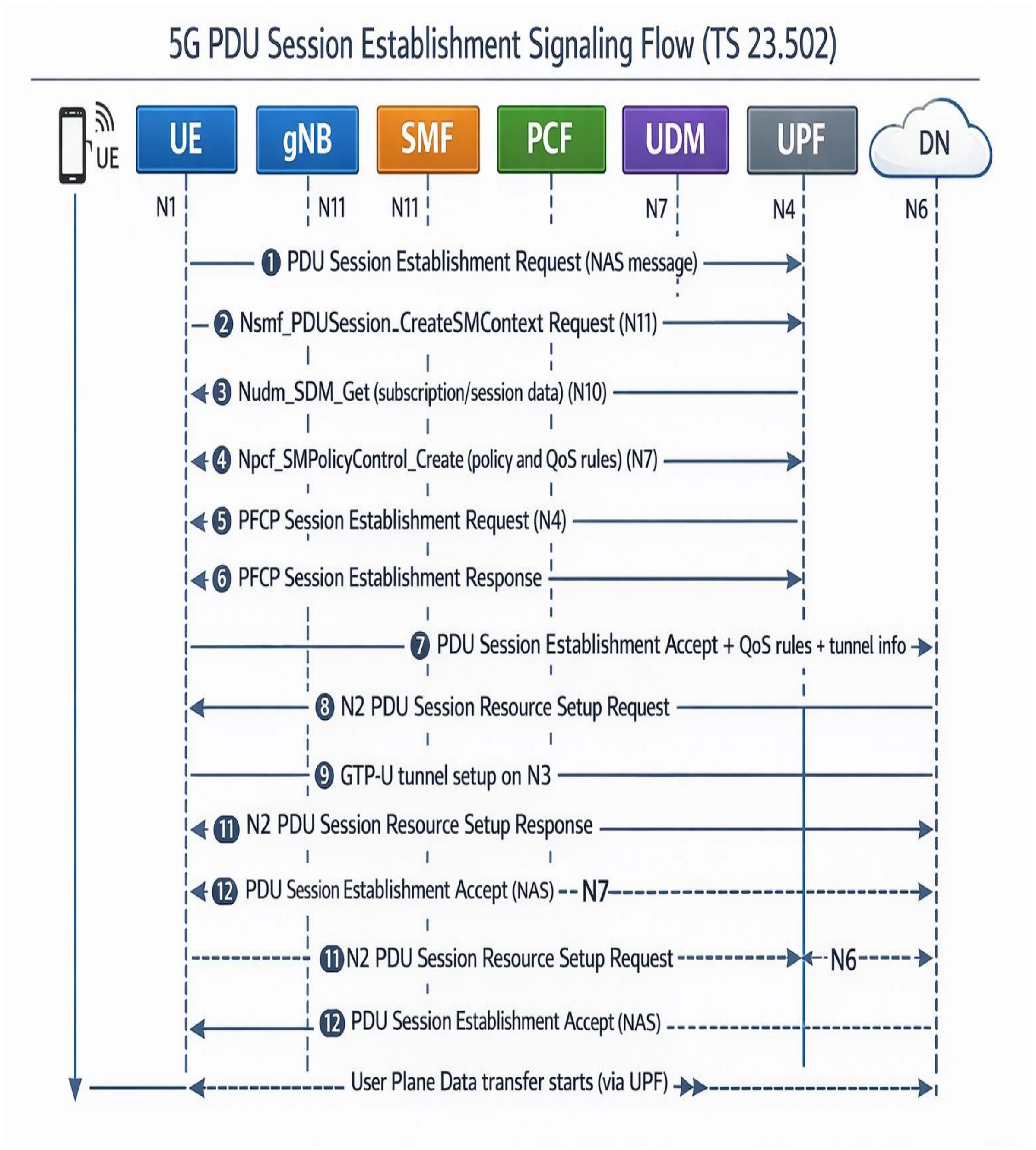
In the 5G Core Service-Based Architecture (SBA), the SMF acts as the central session control entity that orchestrates all user-plane behaviour. It sits between the access and user-plane functions and coordinates with multiple network functions to deliver end-to-end connectivity. The SMF communicates with the AMF over the N11 interface for NAS Session Management signaling, with the PCF over the N7 interface for policy and QoS rules, and with the UDM over the N10 interface to retrieve subscription and session-related data. It controls the UPF via the N4 interface using the PFCP protocol, enabling the setup, modification, and teardown of user-plane tunnels and forwarding rules.

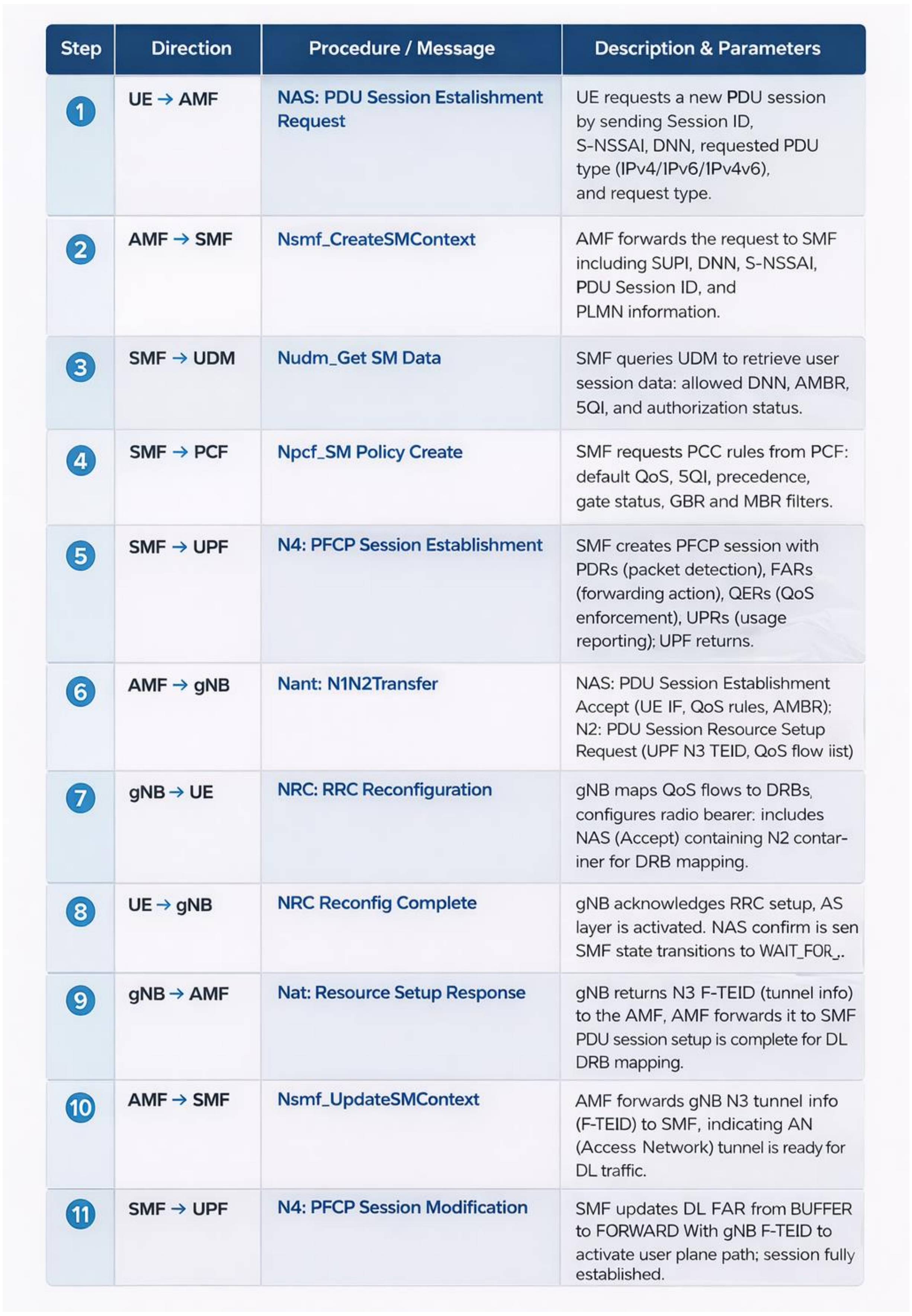
Through these interactions, the SMF ensures that traffic is routed correctly, QoS requirements are enforced, and PDU Sessions are maintained across mobility events and service changes. Its central role allows the 5G Core to support advanced features such as network slicing, edge computing (MEC), and service continuity, making the SMF a key enabler of flexible, scalable, and high-performance 5G services.


5G Core SBA Architecture – SMF Role
NRF (Network Reportory Frunction)
AMF (Access a Mobility Function)
PCF (Policy Control Function)
AUSF Authited Server Function)
N2
N7
AMF (Accet Mobolent Function)
N2
SMF (Session Management Function)
PDU Session Estabishment
QQS & Policy enforcement
UPF tunnel management
Session Modification & Release
UDM Unified Data Management
N7
N11 (NAS Session Management signaling)
UE (User Equipment
Data Network / Internet
N10 (Subscription/session data retireval)
gNB
N4
N3 (User Plane tunnel
N3
UPF User Plane Function
N6

3. PDU Session Establishment Signaling Flow

The PDU Session Establishment procedure defines how a UE requests and receives data connectivity to a specific Data Network (DN) in the 5G system. This procedure is standardized in 3GPP TS 23.502 and involves coordinated signaling between the UE, AMF, SMF, PCF, UDM, and UPF.

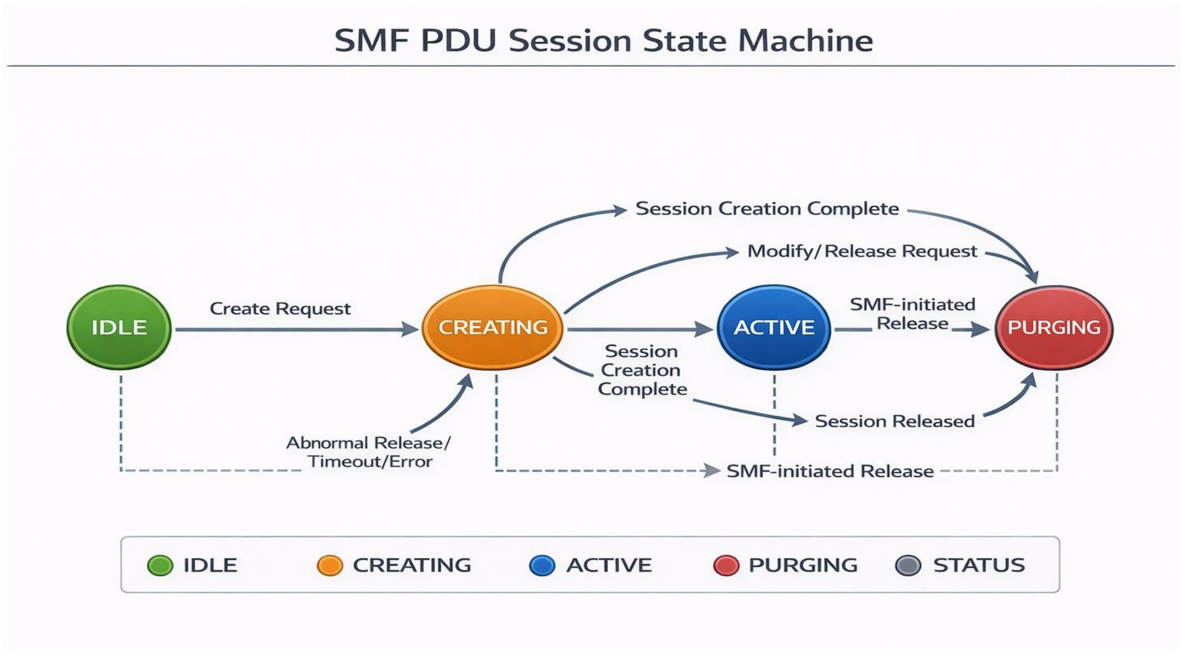

5G PDU Session Establishment Signaling Flow (TS 23.502)
UE
gNB
SMF
PCF
UDM
UPF
DN
N1
N11
N11
N7
N4
N6
1 PDU Session Establishment Request (NAS message)
2 Nsmf_PDUSession_CreateSMContext Request (N11)
3 Nudm_SDM_Get (subscription/session data) (N10)
4 Npcf_SMPolicyControl_Create (policy and QoS rules) (N7)
5 PFCP Session Establishment Request (N4)
6 PFCP Session Establishment Response
7 PDU Session Establishment Accept + QoS rules + tunnel info
8 N2 PDU Session Resource Setup Request
9 GTP-U tunnel setup on N3
11 N2 PDU Session Resource Setup Response
12 PDU Session Establishment Accept (NAS) N7
11 N2 PDU Session Resource Setup Request N6
12 PDU Session Establishment Accept (NAS)
User Plane Data transfer starts (via UPF)

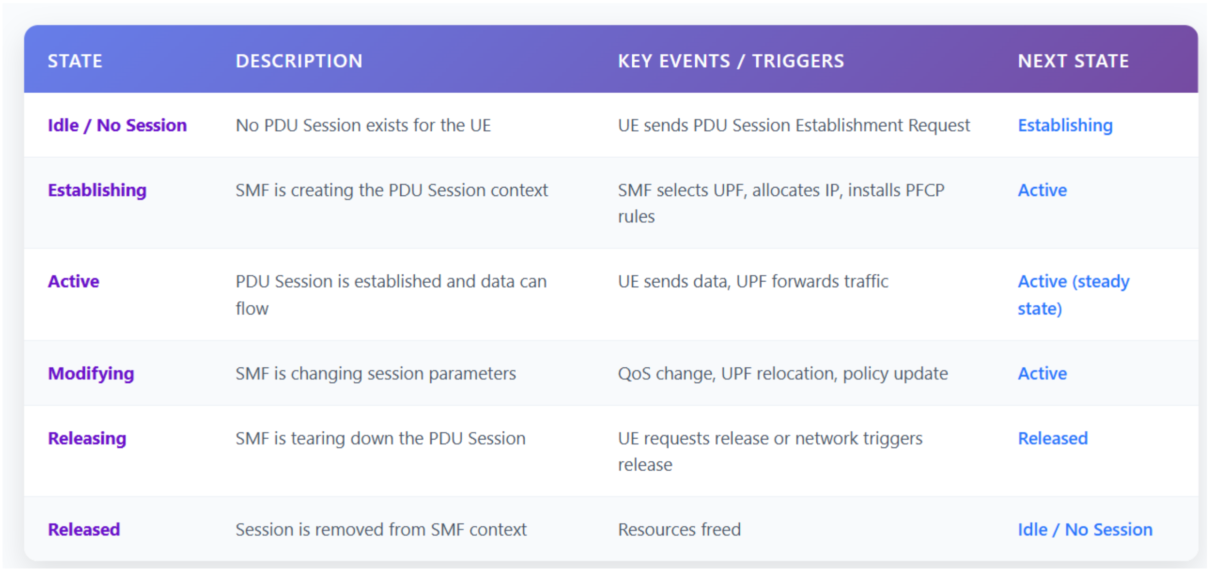


Step	Direction	Procedure / Message	Description & Parameters
1	UE → AMF	NAS: PDU Session Estalishment Request	UE requests a new PDU session by sending Session ID, S-NSSAI, DNN, requested PDU type (IPv4/IPv6/IPv4v6), and request type.
2	AMF → SMF	Nsmf_CreateSMContext	AMF forwards the request to SMF including SUPI, DNN, S-NSSAI, PDU Session ID, and PLMN information.
3	SMF → UDM	Nudm_Get SM Data	SMF queries UDM to retrieve user session data: allowed DNN, AMBR, 5QI, and authorization status.
4	SMF → PCF	Npcf_SM Policy Create	SMF requests PCC rules from PCF: default QoS, 5QI, precedence, gate status, GBR and MBR filters.
5	SMF → UPF	N4: PFCP Session Establishment	SMF creates PFCP session with PDRs (packet detection), FARs (forwarding action), QERs (QoS enforcement), UPRs (usage reporting); UPF returns.
6	AMF → gNB	Nant: N1N2Transfer	NAS: PDU Session Establishment Accept (UE IF, QoS rules, AMBR); N2: PDU Session Resource Setup Request (UPF N3 TEID, QoS flow iist)
7	gNB → UE	NRC: RRC Reconfiguration	gNB maps QoS flows to DRBs, configures radio bearer: includes NAS (Accept) containing N2 contar-iner for DRB mapping.
8	UE → gNB	NRC Reconfig Complete	gNB acknowledges RRC setup, AS layer is activated. NAS confirm is sen SMF state transitions to WAIT_FOR_,.
9	gNB → AMF	Nat: Resource Setup Response	gNB returns N3 F-TEID (tunnel info) to the AMF, AMF forwards it to SMF PDU session setup is complete for DL DRB mapping.
10	AMF → SMF	Nsmf_UpdateSMContext	AMF forwards gNB N3 tunnel info (F-TEID) to SMF, indicating AN (Access Network) tunnel is ready for DL traffic.
11	SMF → UPF	N4: PFCP Session Modification	SMF updates DL FAR from BUFFER to FORWARD With gNB F-TEID to activate user plane path; session fully established.

4. SMF PDU Session State Machine

The SMF PDU Session State Machine describes how a PDU Session transitions through different states during its lifecycle, from creation to termination. The SMF controls these state transitions based on UE requests, network policies, and control-plane signaling defined in 3GPP TS 23.502.





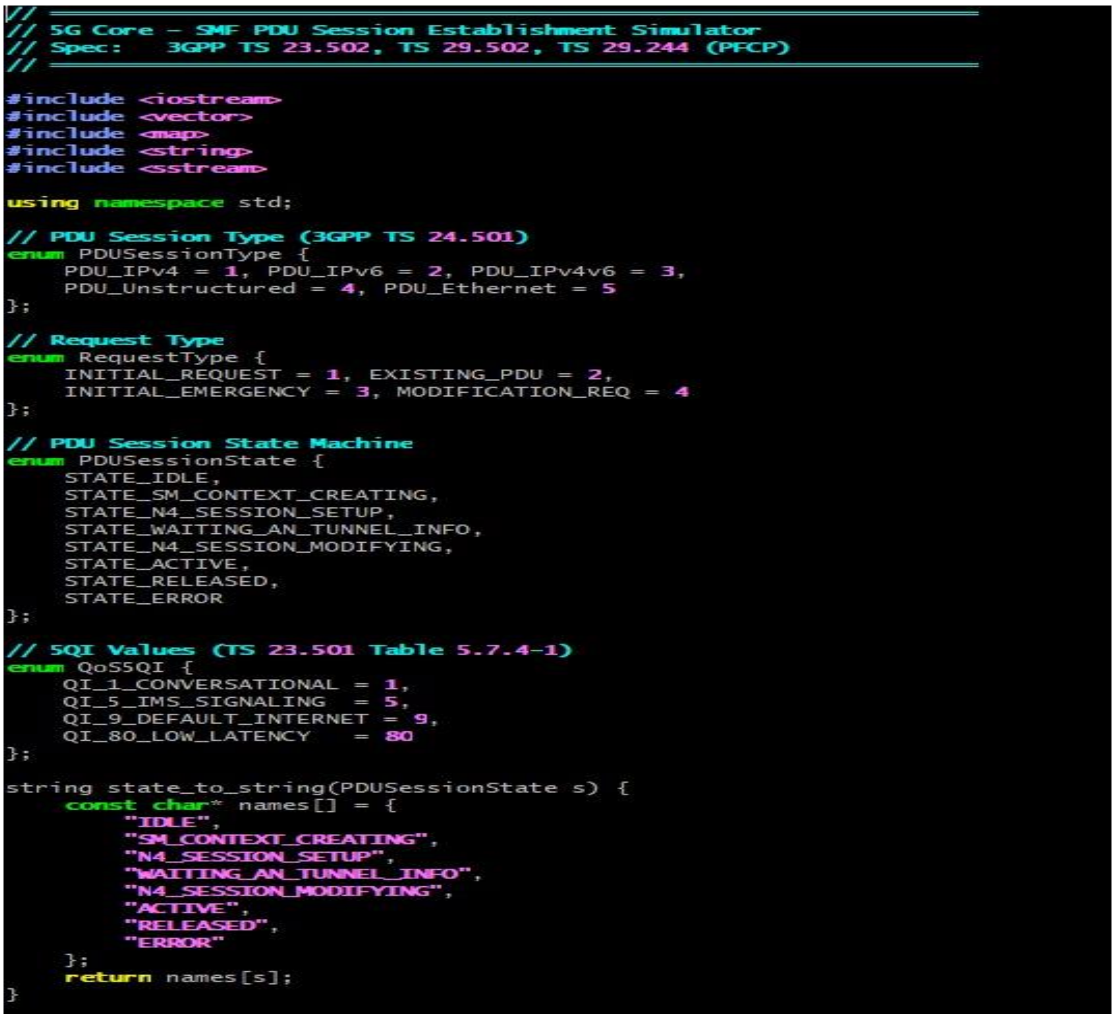
STATE	DESCRIPTION	KEY EVENTS / TRIGGERS	NEXT STATE
Idle / No Session	No PDU Session exists for the UE	UE sends PDU Session Establishment Request	Establishing
Establishing	SMF is creating the PDU Session context	SMF selects UPF, allocates IP, installs PFCP rules	Active
Active	PDU Session is established and data can flow	UE sends data, UPF forwards traffic	Active (steady state)
Modifying	SMF is changing session parameters	QoS change, UPF relocation, policy update	Active
Releasing	SMF is tearing down the PDU Session	UE requests release or network triggers release	Released
Released	Session is removed from SMF context	Resources freed	Idle / No Session

5. C++ IMPLEMENTATION

5.5 Step 1: Enumerations & Type Definitions

Defines the basic enumerations required for the SMF PDU Session Establishment simulation. It includes PDU Session types such as IPv4, IPv6, IPv4v6, Ethernet, and Unstructured sessions, request types used during session procedures, and the PDU Session state machine representing different SMF states from IDLE to ACTIVE and RELEASED.

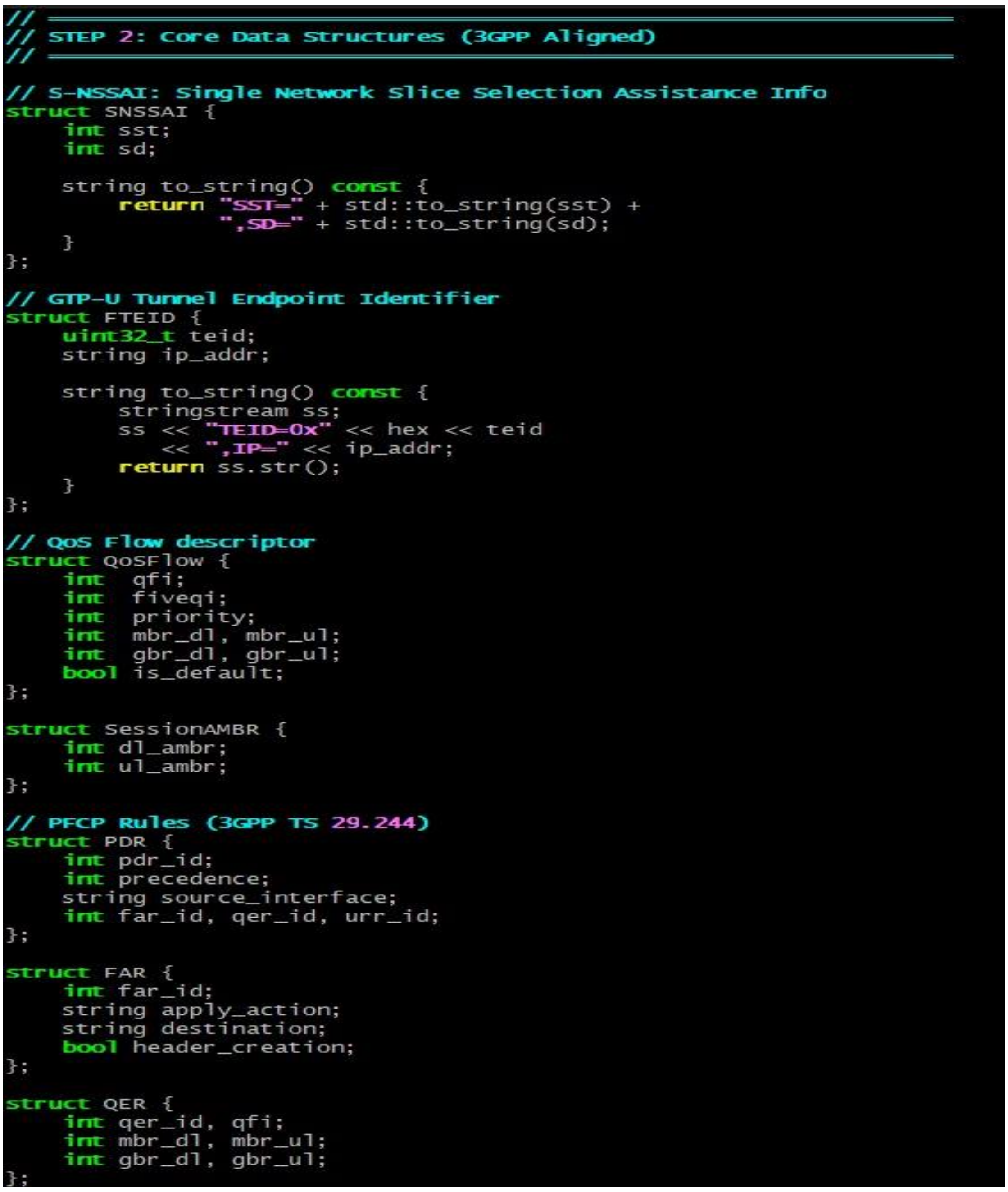
It also defines standard 5QI values based on 3GPP TS 23.501, which represent QoS characteristics like latency and priority for different service types. A helper function is implemented to convert session states into readable strings for logging and debugging during simulation.



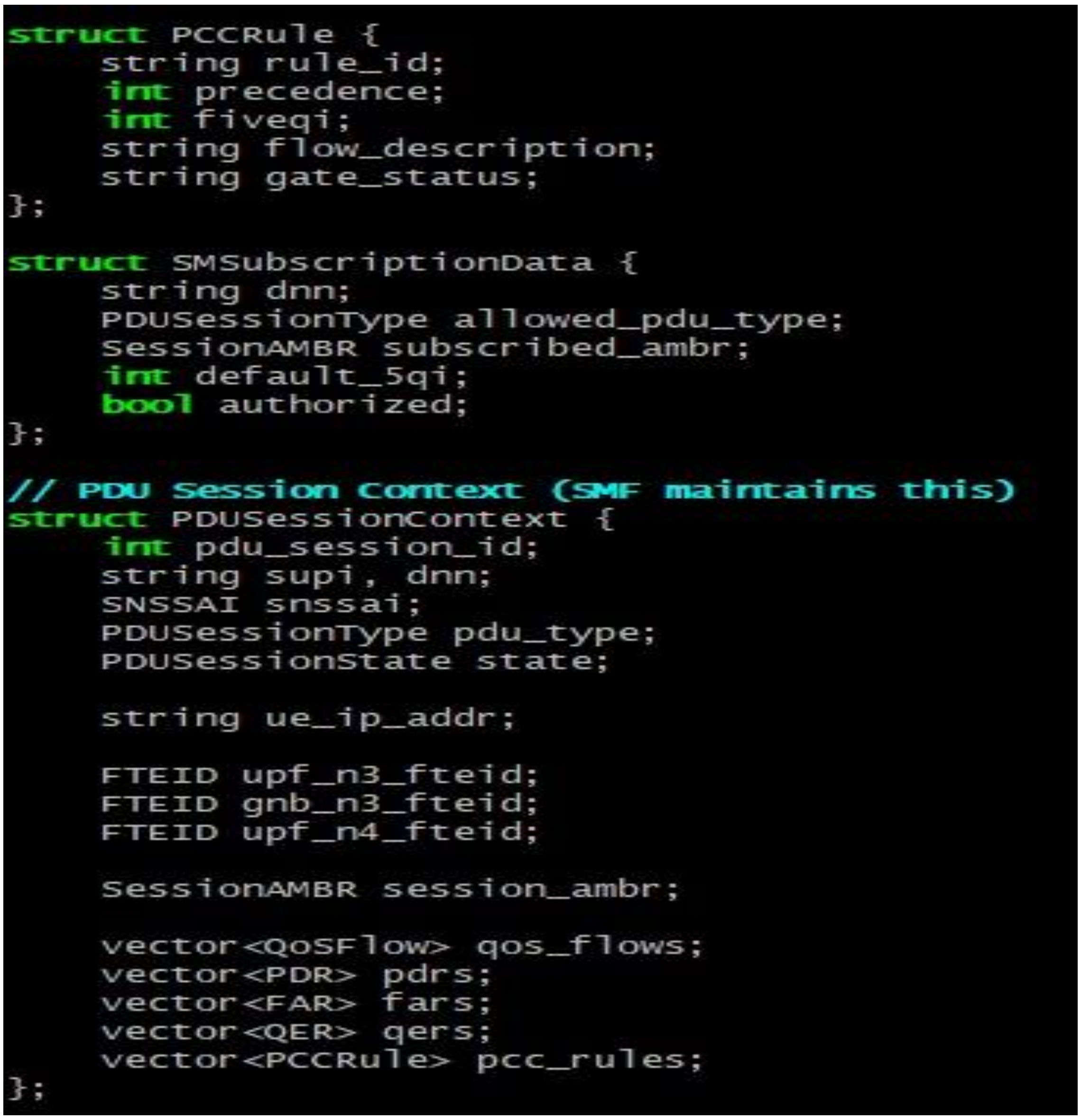
```cpp
// ════════════════════════════════════════════════════════════════
// 5G Core - SMF PDU Session Establishment Simulator
// Spec:   3GPP TS 23.502, TS 29.502, TS 29.244 (PFCP)
// ════════════════════════════════════════════════════════════════

#include <iostream>
#include <vector>
#include <map>
#include <string>
#include <sstream>

using namespace std;

// PDU Session Type (3GPP TS 24.501)
enum PDUSessionType {
    PDU_IPv4 = 1, PDU_IPv6 = 2, PDU_IPv4v6 = 3,
    PDU_Unstructured = 4, PDU_Ethernet = 5
};

// Request Type
enum RequestType {
    INITIAL_REQUEST = 1, EXISTING_PDU = 2,
    INITIAL_EMERGENCY = 3, MODIFICATION_REQ = 4
};

// PDU Session State Machine
enum PDUSessionState {
    STATE_IDLE,
    STATE_SM_CONTEXT_CREATING,
    STATE_N4_SESSION_SETUP,
    STATE_WAITING_AN_TUNNEL_INFO,
    STATE_N4_SESSION_MODIFYING,
    STATE_ACTIVE,
    STATE_RELEASED,
    STATE_ERROR
};

// 5QI Values (TS 23.501 Table 5.7.4-1)
enum QoS5QI {
    QI_1_CONVERSATIONAL  = 1,
    QI_5_IMS_SIGNALING   = 5,
    QI_9_DEFAULT_INTERNET = 9,
    QI_80_LOW_LATENCY    = 80
};

string state_to_string(PDUSessionState s) {
    const char* names[] = {
        "IDLE",
        "SM_CONTEXT_CREATING",
        "N4_SESSION_SETUP",
        "WAITING_AN_TUNNEL_INFO",
        "N4_SESSION_MODIFYING",
        "ACTIVE",
        "RELEASED",
        "ERROR"
    };
    return names[s];
}
```

5.2 Step 2: Core Data Structures

Defines the main data structures required for SMF PDU Session handling. It includes S-NSSAI for network slicing, F-TEID for GTP-U tunnel identification, QoS Flow parameters, Session AMBR, and PFCP rules such as PDR, FAR, and QER. The PDU Session Context acts as the central structure maintained by SMF to store session state, QoS flows, tunnel information, and policy rules during the session lifecycle.



```
//
// STEP 2: Core Data Structures (3GPP Aligned)
//

// S-NSSAI: Single Network Slice Selection Assistance Info
struct SNSSAI {
    int sst;
    int sd;

    string to_string() const {
        return "SST=" + std::to_string(sst) +
               ",SD=" + std::to_string(sd);
    }
};

// GTP-U Tunnel Endpoint Identifier
struct FTEID {
    uint32_t teid;
    string ip_addr;

    string to_string() const {
        stringstream ss;
        ss << "TEID=0x" << hex << teid
           << ",IP=" << ip_addr;
        return ss.str();
    }
};

// QoS Flow descriptor
struct QoSFlow {
    int  qfi;
    int  fiveqi;
    int  priority;
    int  mbr_dl, mbr_ul;
    int  gbr_dl, gbr_ul;
    bool is_default;
};

struct SessionAMBR {
    int dl_ambr;
    int ul_ambr;
};

// PFCP Rules (3GPP TS 29.244)
struct PDR {
    int pdr_id;
    int precedence;
    string source_interface;
    int far_id, qer_id, urr_id;
};

struct FAR {
    int far_id;
    string apply_action;
    string destination;
    bool header_creation;
};

struct QER {
    int qer_id, qfi;
    int mbr_dl, mbr_ul;
    int gbr_dl, gbr_ul;
};
```



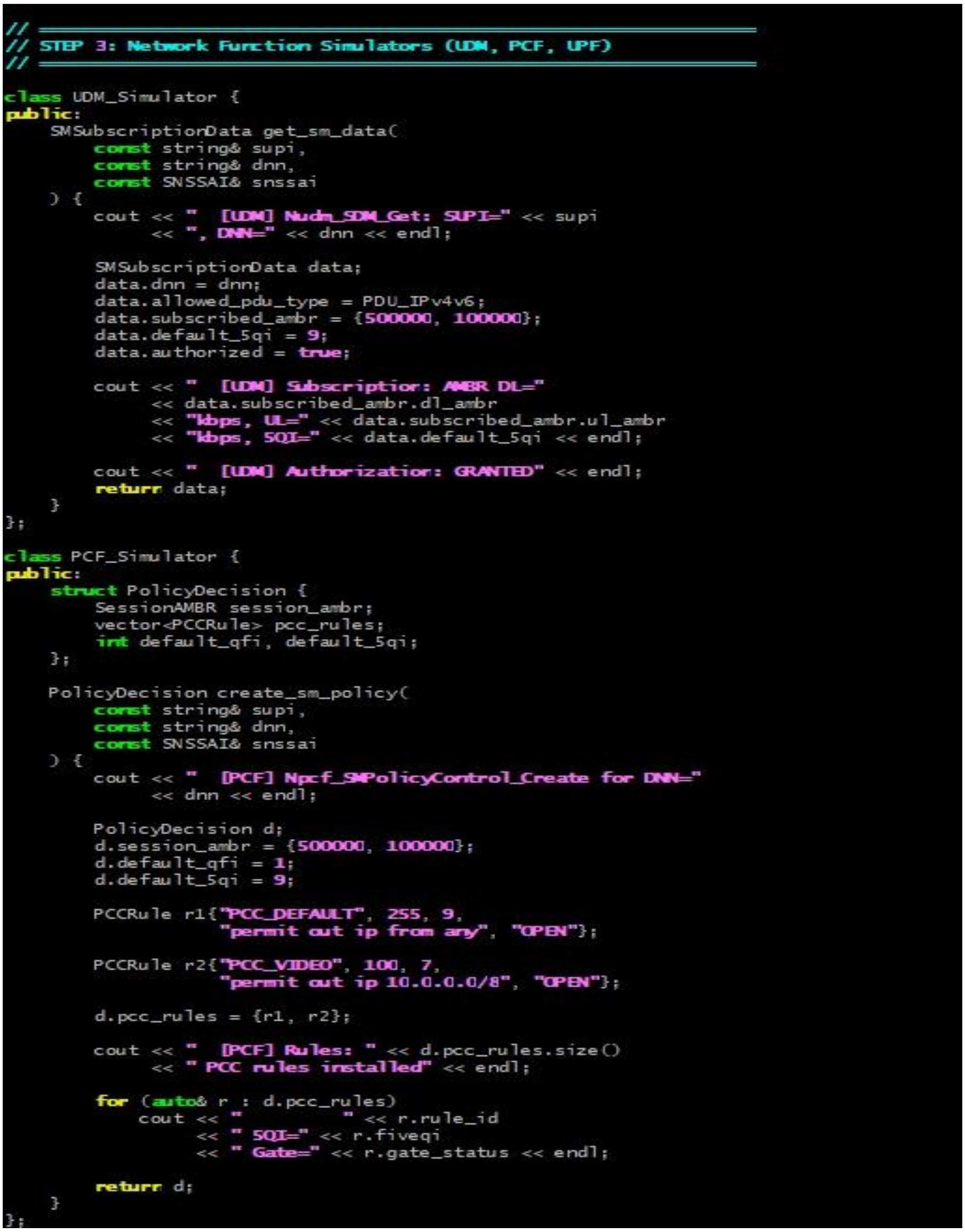
```
struct PCCRule {
    string rule_id;
    int precedence;
    int fiveqi;
    string flow_description;
    string gate_status;
};

struct SMSubscriptionData {
    string dnn;
    PDUSessionType allowed_pdu_type;
    SessionAMBR subscribed_ambr;
    int default_5qi;
    bool authorized;
};

// PDU Session Context (SMF maintains this)
struct PDUSessionContext {
    int pdu_session_id;
    string supi, dnn;
    SNSSAI snssai;
    PDUSessionType pdu_type;
    PDUSessionState state;

    string ue_ip_addr;

    FTEID upf_n3_fteid;
    FTEID gnb_n3_fteid;
    FTEID upf_n4_fteid;

    SessionAMBR session_ambr;

    vector<QoSFlow> qos_flows;
    vector<PDR> pdrs;
    vector<FAR> fars;
    vector<QER> qers;
    vector<PCCRule> pcc_rules;
};
```

5.3 Step 3: Network Function Simulators (UDM, PCF, UPF)

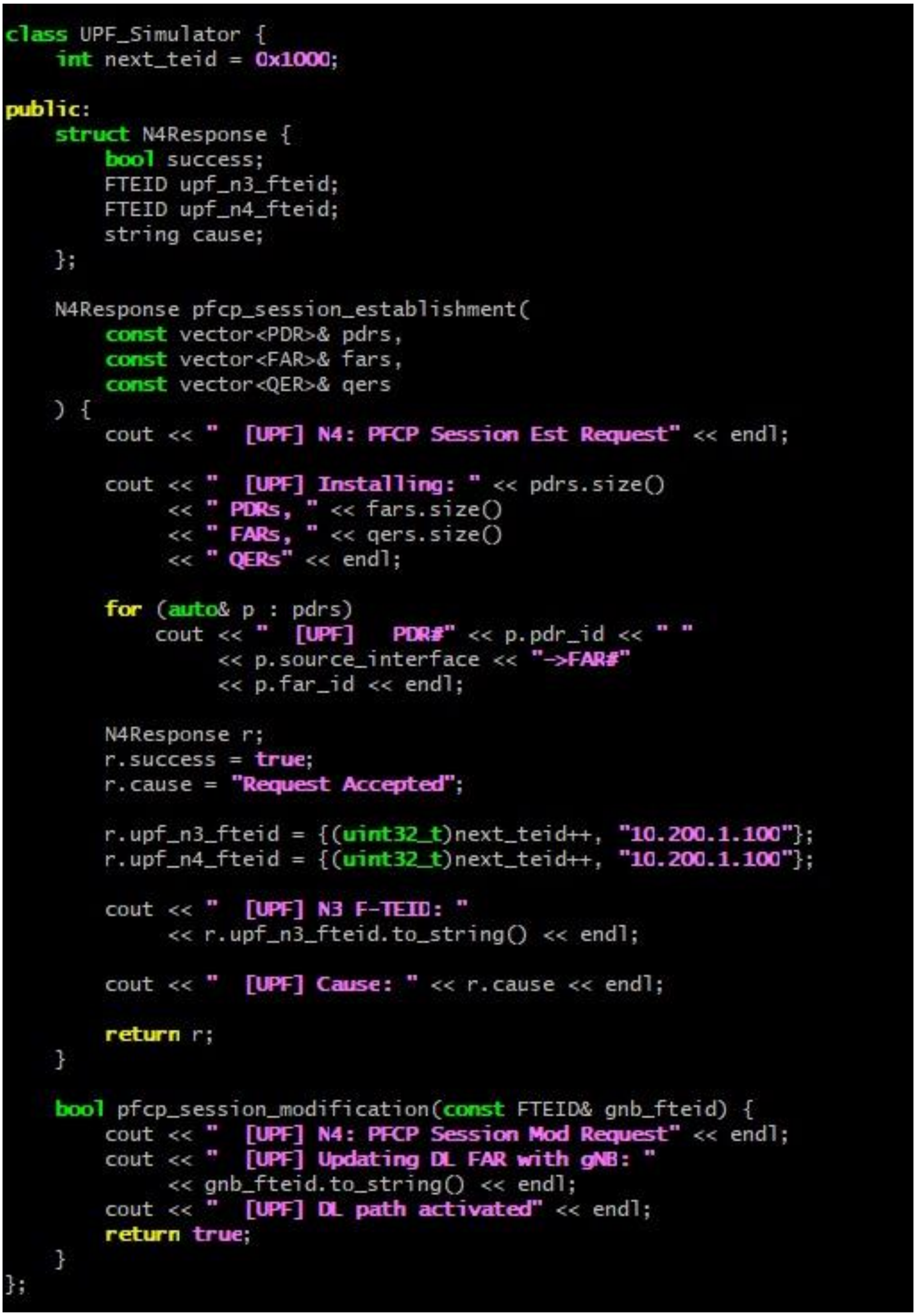
Simulates the interaction between SMF and other 5G Core network functions during PDU Session Establishment.

- UDM (Nudm_SDM_Get) – Provides subscriber data such as allowed PDU session type, subscribed AMBR, default 5QI, and authorization status.
- PCF (Npcf_SMPolicyControl_Create) – Generates policy decisions and PCC rules used for QoS flow creation and traffic control.

- UPF / PFCP – Performs PFCP session establishment and modification, installs PDR, FAR, and QER rules, and returns F-TEID tunnel information for user plane setup.



```
// ============================================================================
// STEP 3: Network Function Simulators (UDM, PCF, UPF)
// ============================================================================

class UDM_Simulator {
public:
    SMSubscriptionData get_sm_data(
        const string& supi,
        const string& dnn,
        const SNSSAI& snssai
    ) {
        cout << "  [UDM] Nudm_SDM_Get: SUPI=" << supi
             << ", DNN=" << dnn << endl;

        SMSubscriptionData data;
        data.dnn = dnn;
        data.allowed_pdu_type = PDU_IPv4v6;
        data.subscribed_ambr = {500000, 100000};
        data.default_5qi = 9;
        data.authorized = true;

        cout << "  [UDM] Subscription: AMBR DL="
             << data.subscribed_ambr.dl_ambr
             << "kbps, UL=" << data.subscribed_ambr.ul_ambr
             << "kbps, 5QI=" << data.default_5qi << endl;

        cout << "  [UDM] Authorization: GRANTED" << endl;
        return data;
    }
};

class PCF_Simulator {
public:
    struct PolicyDecision {
        SessionAMBR session_ambr;
        vector<PCCRule> pcc_rules;
        int default_qfi, default_5qi;
    };

    PolicyDecision create_sm_policy(
        const string& supi,
        const string& dnn,
        const SNSSAI& snssai
    ) {
        cout << "  [PCF] Npcf_SMPolicyControl_Create for DNN="
             << dnn << endl;

        PolicyDecision d;
        d.session_ambr = {500000, 100000};
        d.default_qfi = 1;
        d.default_5qi = 9;

        PCCRule r1{"PCC_DEFAULT", 255, 9,
                   "permit out ip from any", "OPEN"};

        PCCRule r2{"PCC_VIDEO", 100, 7,
                   "permit out ip 10.0.0.0/8", "OPEN"};

        d.pcc_rules = {r1, r2};

        cout << "  [PCF] Rules: " << d.pcc_rules.size()
             << " PCC rules installed" << endl;

        for (auto& r : d.pcc_rules)
            cout << "         " << r.rule_id
                 << " 5QI=" << r.fiveqi
                 << " Gate=" << r.gate_status << endl;

        return d;
    }
};
```

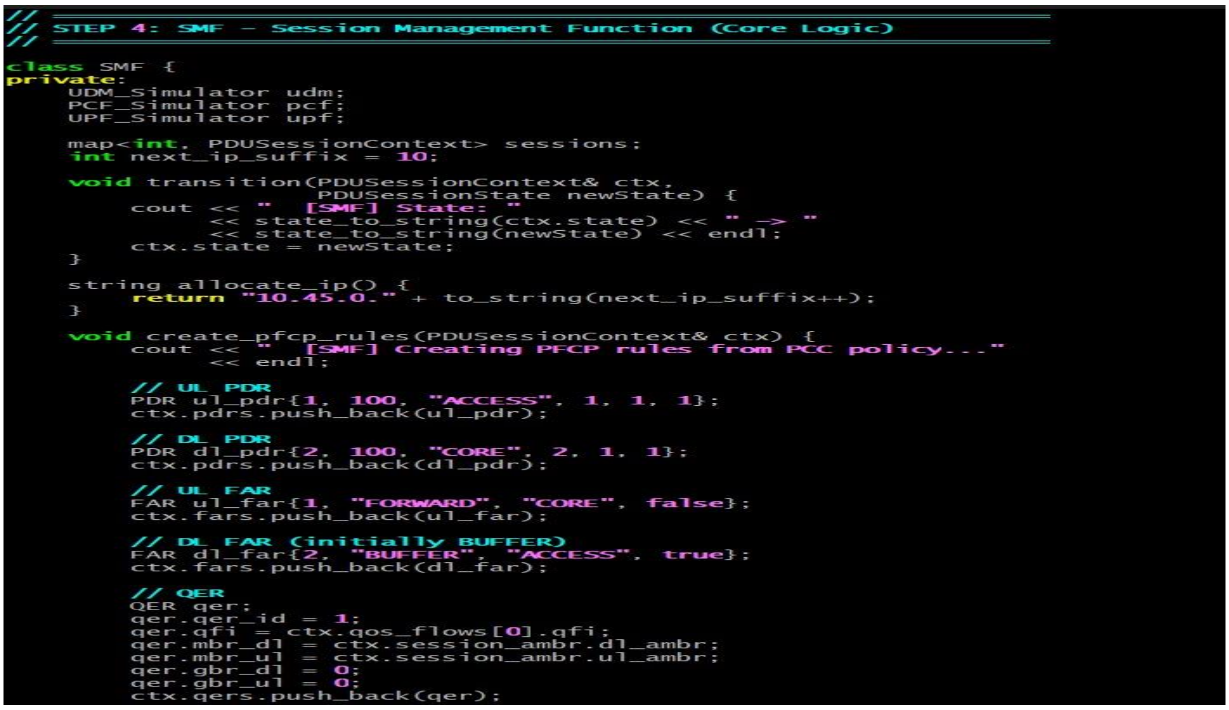


```
class UPF_Simulator {
    int next_teid = 0x1000;

public:
    struct N4Response {
        bool success;
        FTEID upf_n3_fteid;
        FTEID upf_n4_fteid;
        string cause;
    };

    N4Response pfcp_session_establishment(
        const vector<PDR>& pdrs,
        const vector<FAR>& fars,
        const vector<QER>& qers
    ) {
        cout << "  [UPF] N4: PFCP Session Est Request" << endl;

        cout << "  [UPF] Installing: " << pdrs.size()
             << " PDRs, " << fars.size()
             << " FARs, " << qers.size()
             << " QERs" << endl;

        for (auto& p : pdrs)
            cout << "    [UPF]   PDR#" << p.pdr_id << " "
                 << p.source_interface << "->FAR#"
                 << p.far_id << endl;

        N4Response r;
        r.success = true;
        r.cause = "Request Accepted";

        r.upf_n3_fteid = {(uint32_t)next_teid++, "10.200.1.100"};
        r.upf_n4_fteid = {(uint32_t)next_teid++, "10.200.1.100"};

        cout << "  [UPF] N3 F-TEID: "
             << r.upf_n3_fteid.to_string() << endl;

        cout << "  [UPF] Cause: " << r.cause << endl;

        return r;
    }

    bool pfcp_session_modification(const FTEID& gnb_fteid) {
        cout << "  [UPF] N4: PFCP Session Mod Request" << endl;
        cout << "  [UPF] Updating DL FAR with gNB: "
             << gnb_fteid.to_string() << endl;
        cout << "  [UPF] DL path activated" << endl;
        return true;
    }
};
```

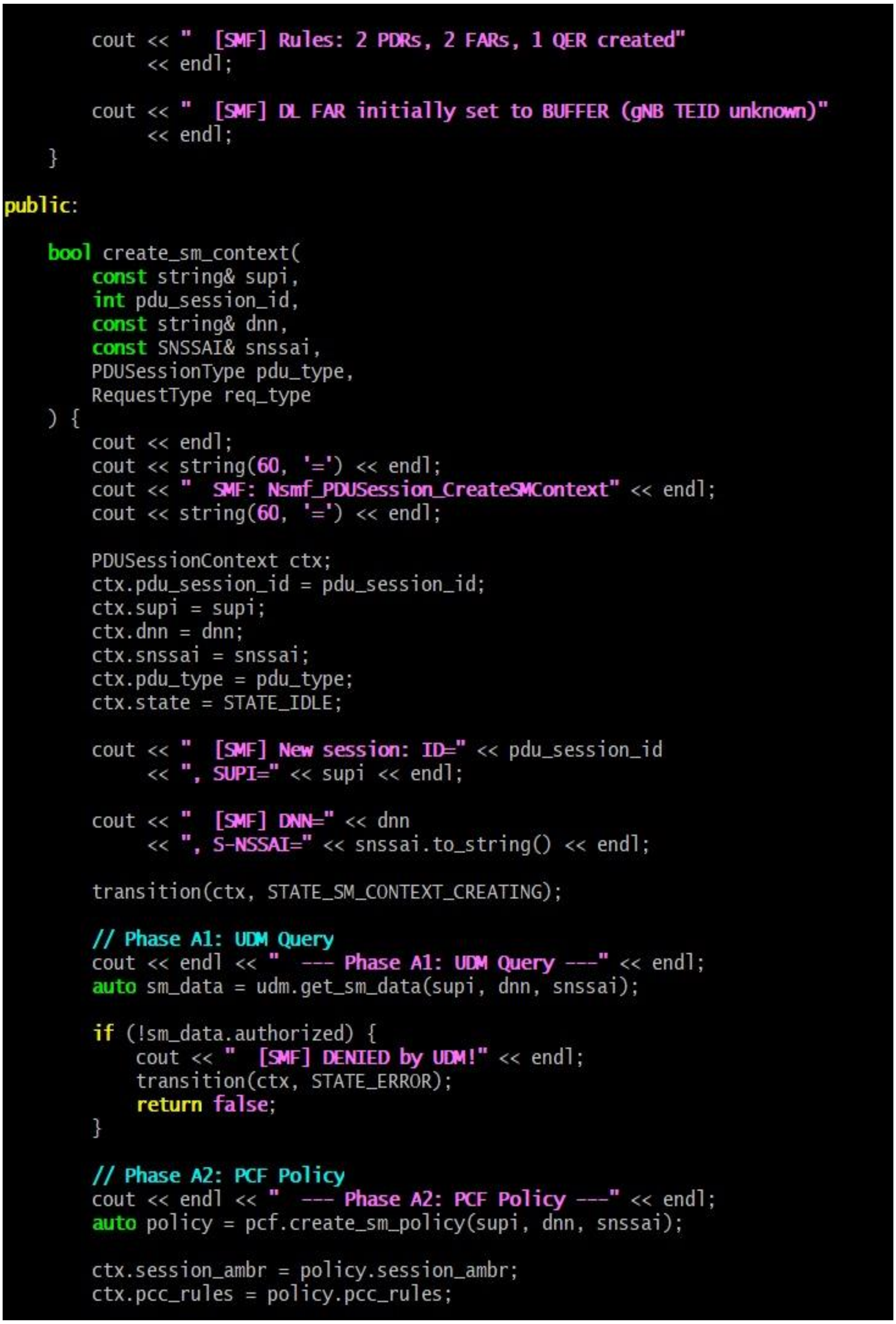
5.4 Step 4: SMF – Session Management Function (Core Logic)

Implements the main SMF logic responsible for handling the complete PDU Session Establishment procedure and managing session state transitions.

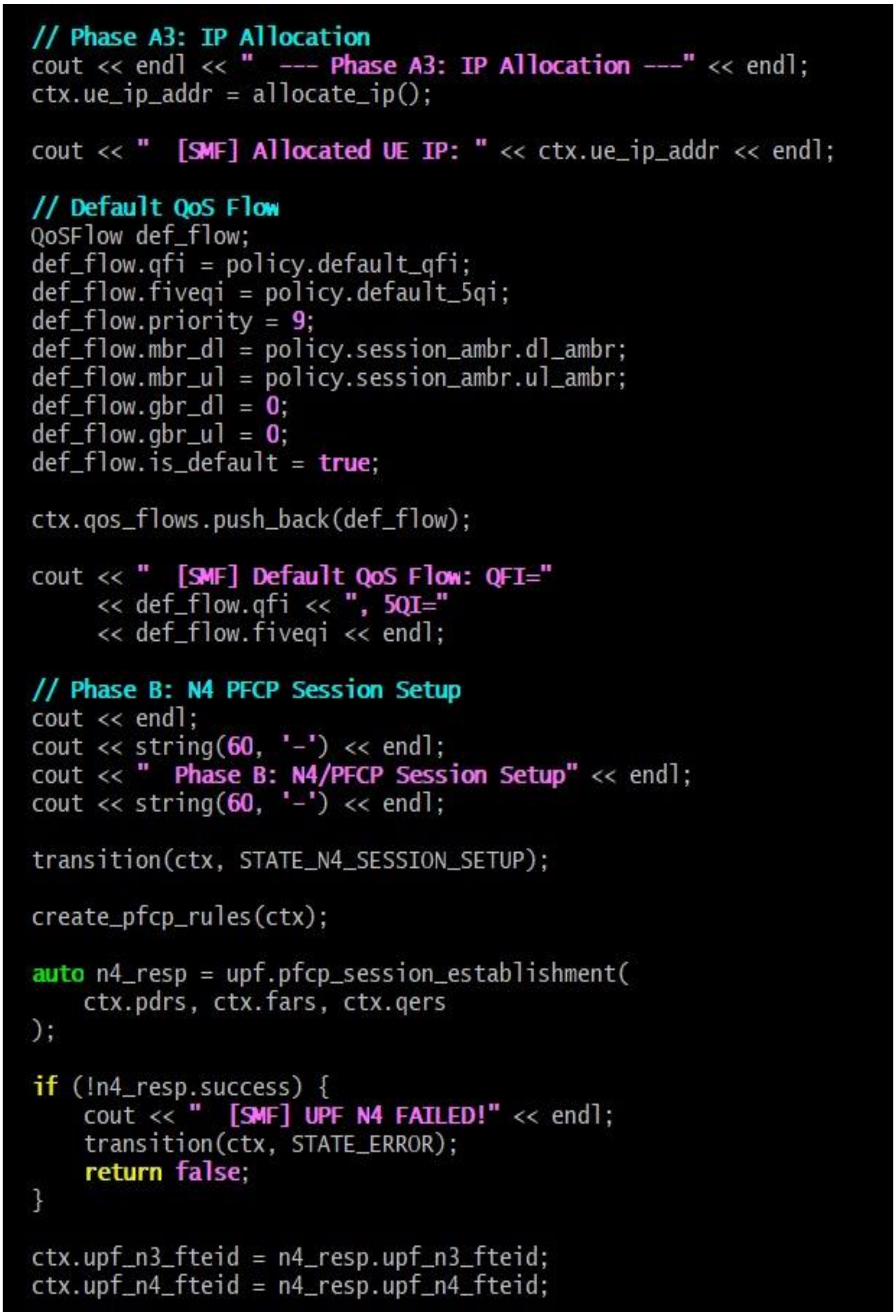
- SM Context Creation – Creates a new session context, stores subscriber and session details, and initiates the session state machine.
- Phase A (UDM & PCF Interaction) – Retrieves subscription data from UDM and policy rules from PCF, then creates default QoS flow and allocates UE IP address.
- Phase B (N4 / PFCP Session Setup) – Creates PFCP rules (PDR, FAR, QER) and establishes the session with UPF.
- Phase C (N1N2 Message Transfer) – Sends NAS PDU Session Establishment Accept and NGAP resource setup information to AMF.
- Phase D (Session Update) – After receiving gNB tunnel information, updates DL FAR from BUFFER to FORWARD using PFCP Session Modification and moves session to ACTIVE state.



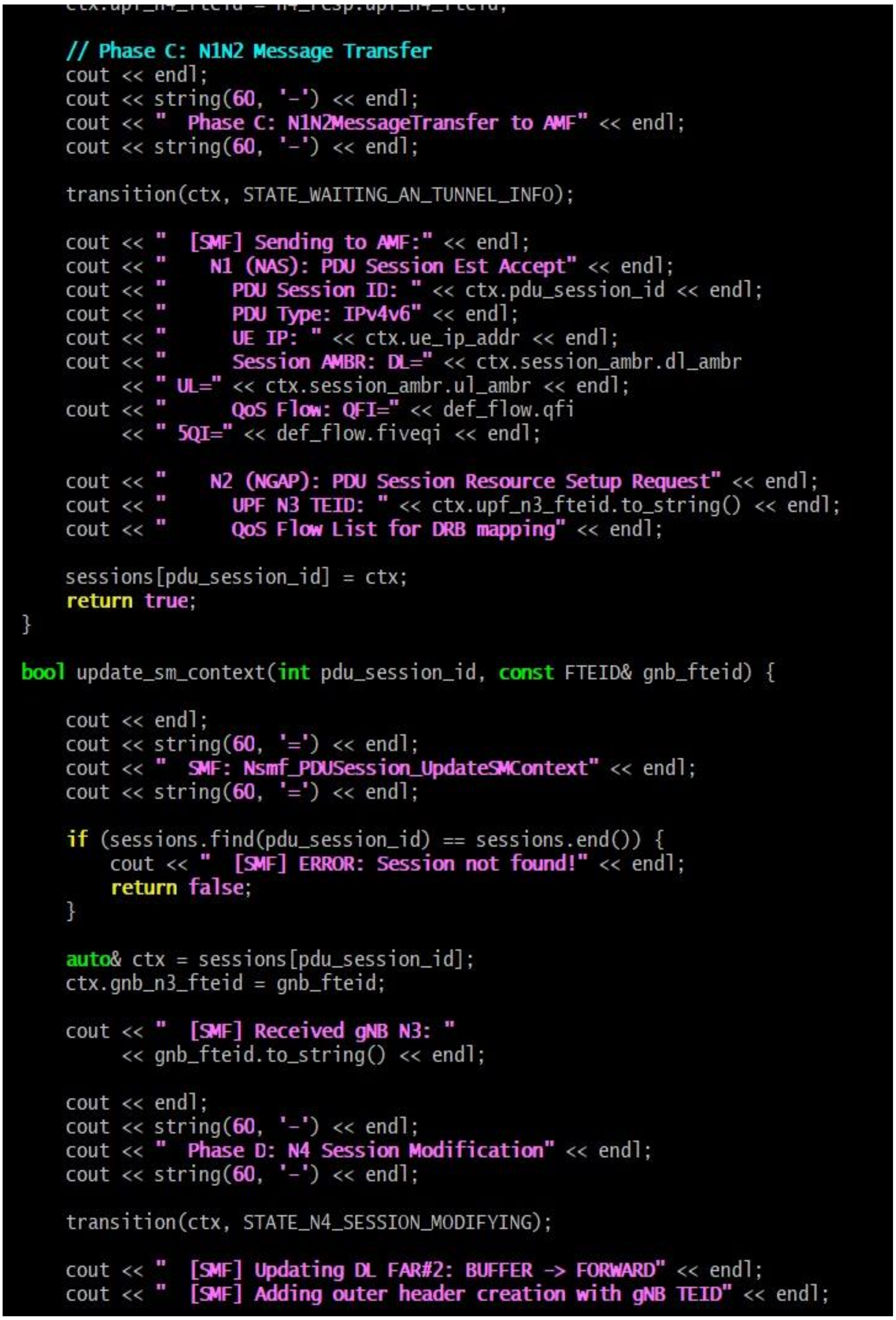
```
// ==============================================================
// STEP 4: SMF - Session Management Function (Core Logic)
// ==============================================================
class SMF {
private:
    UDM_Simulator udm;
    PCF_Simulator pcf;
    UPF_Simulator upf;

    map<int, PDUSessionContext> sessions;
    int next_ip_suffix = 10;

    void transition(PDUSessionContext& ctx,
                    PDUSessionState newState) {
        cout << "   [SMF] State: "
             << state_to_string(ctx.state) << " -> "
             << state_to_string(newState) << endl;
        ctx.state = newState;
    }

    string allocate_ip() {
        return "10.45.0." + to_string(next_ip_suffix++);
    }

    void create_pfcp_rules(PDUSessionContext& ctx) {
        cout << "   [SMF] Creating PFCP rules from PCC policy..."
             << endl;

        // UL PDR
        PDR ul_pdr{1, 100, "ACCESS", 1, 1, 1};
        ctx.pdrs.push_back(ul_pdr);

        // DL PDR
        PDR dl_pdr{2, 100, "CORE", 2, 1, 1};
        ctx.pdrs.push_back(dl_pdr);

        // UL FAR
        FAR ul_far{1, "FORWARD", "CORE", false};
        ctx.fars.push_back(ul_far);

        // DL FAR (initially BUFFER)
        FAR dl_far{2, "BUFFER", "ACCESS", true};
        ctx.fars.push_back(dl_far);

        // QER
        QER qer;
        qer.qer_id = 1;
        qer.qfi = ctx.qos_flows[0].qfi;
        qer.mbr_dl = ctx.session_ambr.dl_ambr;
        qer.mbr_ul = ctx.session_ambr.ul_ambr;
        qer.gbr_dl = 0;
        qer.gbr_ul = 0;
        ctx.qers.push_back(qer);
```



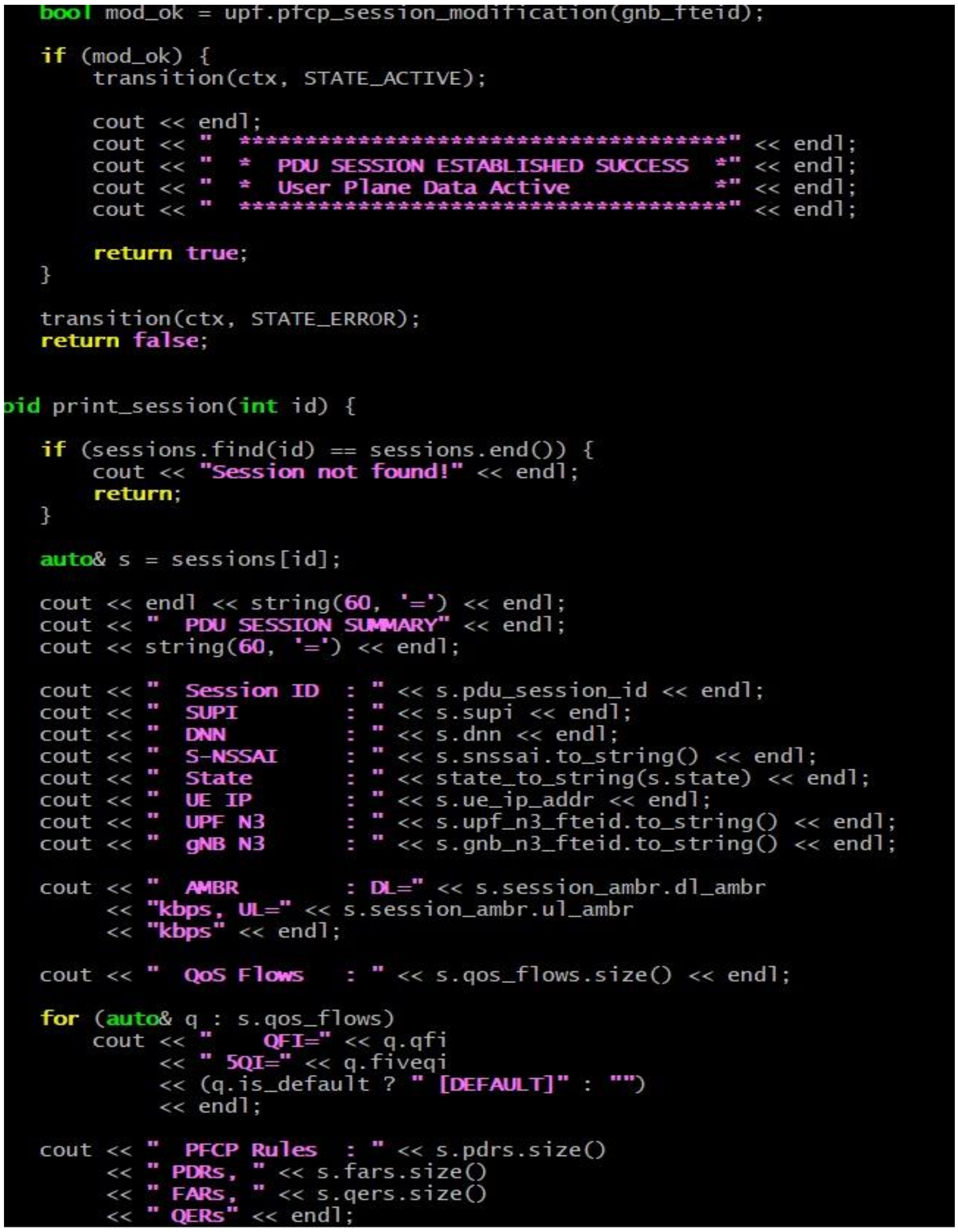
```
        cout << "  [SMF] Rules: 2 PDRs, 2 FARs, 1 QER created"
             << endl;

        cout << "  [SMF] DL FAR initially set to BUFFER (gNB TEID unknown)"
             << endl;
    }

public:

    bool create_sm_context(
        const string& supi,
        int pdu_session_id,
        const string& dnn,
        const SNSSAI& snssai,
        PDUSessionType pdu_type,
        RequestType req_type
    ) {
        cout << endl;
        cout << string(60, '=') << endl;
        cout << "  SMF: Nsmf_PDUSession_CreateSMContext" << endl;
        cout << string(60, '=') << endl;

        PDUSessionContext ctx;
        ctx.pdu_session_id = pdu_session_id;
        ctx.supi = supi;
        ctx.dnn = dnn;
        ctx.snssai = snssai;
        ctx.pdu_type = pdu_type;
        ctx.state = STATE_IDLE;

        cout << "  [SMF] New session: ID=" << pdu_session_id
             << ", SUPI=" << supi << endl;

        cout << "  [SMF] DNN=" << dnn
             << ", S-NSSAI=" << snssai.to_string() << endl;

        transition(ctx, STATE_SM_CONTEXT_CREATING);

        // Phase A1: UDM Query
        cout << endl << "  --- Phase A1: UDM Query ---" << endl;
        auto sm_data = udm.get_sm_data(supi, dnn, snssai);

        if (!sm_data.authorized) {
            cout << "  [SMF] DENIED by UDM!" << endl;
            transition(ctx, STATE_ERROR);
            return false;
        }

        // Phase A2: PCF Policy
        cout << endl << "  --- Phase A2: PCF Policy ---" << endl;
        auto policy = pcf.create_sm_policy(supi, dnn, snssai);

        ctx.session_ambr = policy.session_ambr;
        ctx.pcc_rules = policy.pcc_rules;
```



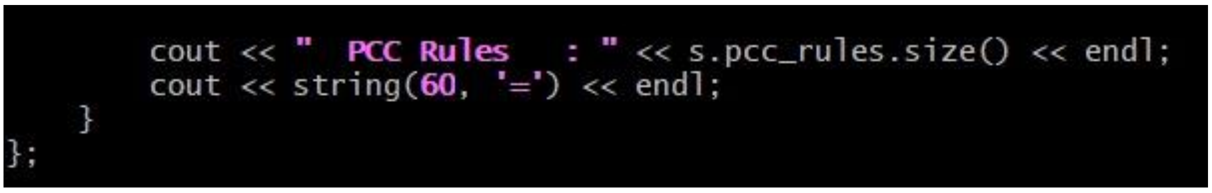
```
// Phase A3: IP Allocation
cout << endl << "  --- Phase A3: IP Allocation ---" << endl;
ctx.ue_ip_addr = allocate_ip();

cout << "  [SMF] Allocated UE IP: " << ctx.ue_ip_addr << endl;

// Default QoS Flow
QoSFlow def_flow;
def_flow.qfi = policy.default_qfi;
def_flow.fiveqi = policy.default_5qi;
def_flow.priority = 9;
def_flow.mbr_dl = policy.session_ambr.dl_ambr;
def_flow.mbr_ul = policy.session_ambr.ul_ambr;
def_flow.gbr_dl = 0;
def_flow.gbr_ul = 0;
def_flow.is_default = true;

ctx.qos_flows.push_back(def_flow);

cout << "  [SMF] Default QoS Flow: QFI="
     << def_flow.qfi << ", 5QI="
     << def_flow.fiveqi << endl;

// Phase B: N4 PFCP Session Setup
cout << endl;
cout << string(60, '-') << endl;
cout << "  Phase B: N4/PFCP Session Setup" << endl;
cout << string(60, '-') << endl;

transition(ctx, STATE_N4_SESSION_SETUP);

create_pfcp_rules(ctx);

auto n4_resp = upf.pfcp_session_establishment(
    ctx.pdrs, ctx.fars, ctx.qers
);

if (!n4_resp.success) {
    cout << "  [SMF] UPF N4 FAILED!" << endl;
    transition(ctx, STATE_ERROR);
    return false;
}

ctx.upf_n3_fteid = n4_resp.upf_n3_fteid;
ctx.upf_n4_fteid = n4_resp.upf_n4_fteid;
```



```cpp
    ctx.upf_n4_fteid = n4_resp.upf_n4_fteid;

    // Phase C: N1N2 Message Transfer
    cout << endl;
    cout << string(60, '-') << endl;
    cout << "  Phase C: N1N2MessageTransfer to AMF" << endl;
    cout << string(60, '-') << endl;

    transition(ctx, STATE_WAITING_AN_TUNNEL_INFO);

    cout << "  [SMF] Sending to AMF:" << endl;
    cout << "    N1 (NAS): PDU Session Est Accept" << endl;
    cout << "      PDU Session ID: " << ctx.pdu_session_id << endl;
    cout << "      PDU Type: IPv4v6" << endl;
    cout << "      UE IP: " << ctx.ue_ip_addr << endl;
    cout << "      Session AMBR: DL=" << ctx.session_ambr.dl_ambr
         << " UL=" << ctx.session_ambr.ul_ambr << endl;
    cout << "      QoS Flow: QFI=" << def_flow.qfi
         << " 5QI=" << def_flow.fiveqi << endl;

    cout << "    N2 (NGAP): PDU Session Resource Setup Request" << endl;
    cout << "      UPF N3 TEID: " << ctx.upf_n3_fteid.to_string() << endl;
    cout << "      QoS Flow List for DRB mapping" << endl;

    sessions[pdu_session_id] = ctx;
    return true;
}

bool update_sm_context(int pdu_session_id, const FTEID& gnb_fteid) {

    cout << endl;
    cout << string(60, '=') << endl;
    cout << "  SMF: Nsmf_PDUSession_UpdateSMContext" << endl;
    cout << string(60, '=') << endl;

    if (sessions.find(pdu_session_id) == sessions.end()) {
        cout << "  [SMF] ERROR: Session not found!" << endl;
        return false;
    }

    auto& ctx = sessions[pdu_session_id];
    ctx.gnb_n3_fteid = gnb_fteid;

    cout << "  [SMF] Received gNB N3: "
         << gnb_fteid.to_string() << endl;

    cout << endl;
    cout << string(60, '-') << endl;
    cout << "  Phase D: N4 Session Modification" << endl;
    cout << string(60, '-') << endl;

    transition(ctx, STATE_N4_SESSION_MODIFYING);

    cout << "  [SMF] Updating DL FAR#2: BUFFER -> FORWARD" << endl;
    cout << "  [SMF] Adding outer header creation with gNB TEID" << endl;
```



```
    bool mod_ok = upf.pfcp_session_modification(gnb_fteid);

    if (mod_ok) {
        transition(ctx, STATE_ACTIVE);

        cout << endl;
        cout << "  ***************************************" << endl;
        cout << "  *  PDU SESSION ESTABLISHED SUCCESS    *" << endl;
        cout << "  *  User Plane Data Active             *" << endl;
        cout << "  ***************************************" << endl;

        return true;
    }

    transition(ctx, STATE_ERROR);
    return false;


oid print_session(int id) {

    if (sessions.find(id) == sessions.end()) {
        cout << "Session not found!" << endl;
        return;
    }

    auto& s = sessions[id];

    cout << endl << string(60, '=') << endl;
    cout << "  PDU SESSION SUMMARY" << endl;
    cout << string(60, '=') << endl;

    cout << "  Session ID  : " << s.pdu_session_id << endl;
    cout << "  SUPI        : " << s.supi << endl;
    cout << "  DNN         : " << s.dnn << endl;
    cout << "  S-NSSAI     : " << s.snssai.to_string() << endl;
    cout << "  State       : " << state_to_string(s.state) << endl;
    cout << "  UE IP       : " << s.ue_ip_addr << endl;
    cout << "  UPF N3      : " << s.upf_n3_fteid.to_string() << endl;
    cout << "  gNB N3      : " << s.gnb_n3_fteid.to_string() << endl;

    cout << "  AMBR        : DL=" << s.session_ambr.dl_ambr
         << "kbps, UL=" << s.session_ambr.ul_ambr
         << "kbps" << endl;

    cout << "  QoS Flows   : " << s.qos_flows.size() << endl;

    for (auto& q : s.qos_flows)
        cout << "     QFI=" << q.qfi
             << " 5QI=" << q.fiveqi
             << (q.is_default ? " [DEFAULT]" : "")
             << endl;

    cout << "  PFCP Rules  : " << s.pdrs.size()
         << " PDRs, " << s.fars.size()
         << " FARs, " << s.qers.size()
         << " QERs" << endl;
```

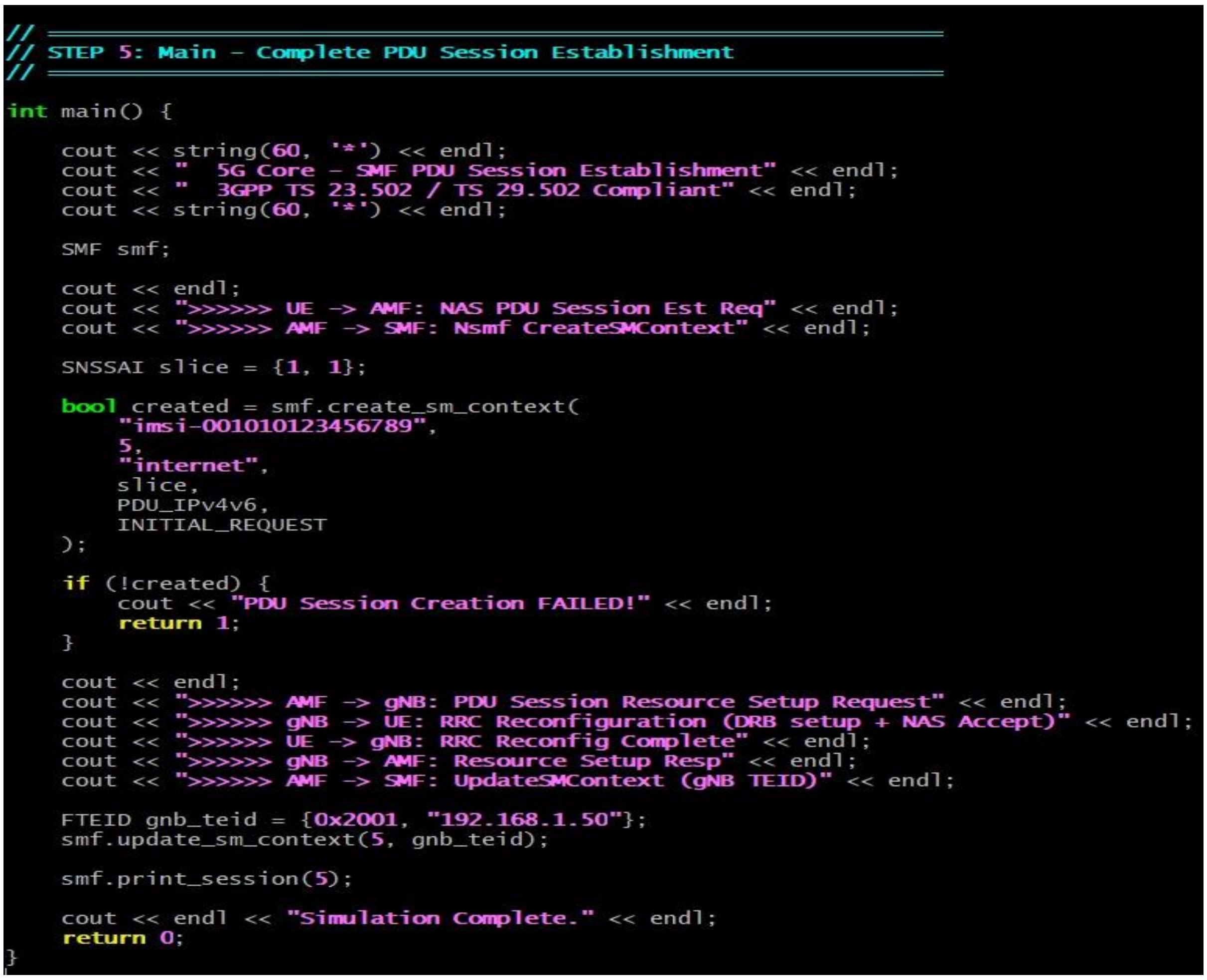


```
        cout << "  PCC Rules   : " << s.pcc_rules.size() << endl;
        cout << string(60, '=') << endl;
    }
};
```

5.5 Step 5: Main – Complete PDU Session Establishment

Implements the main simulation flow that executes the complete PDU Session Establishment procedure. The main function creates the SMF instance, initializes the session parameters, and simulates the UE request flow from UE to AMF and then to SMF for session creation. It triggers the SMF context creation process, which performs subscription validation, policy handling, IP allocation, QoS flow creation, and PFCP session setup.

After the initial setup, the program simulates the access network procedure where gNB resource setup is completed and tunnel information is sent back to the SMF. The SMF then updates the session using PFCP modification to activate the user plane path. Finally, the program prints the PDU session summary and completes the simulation.

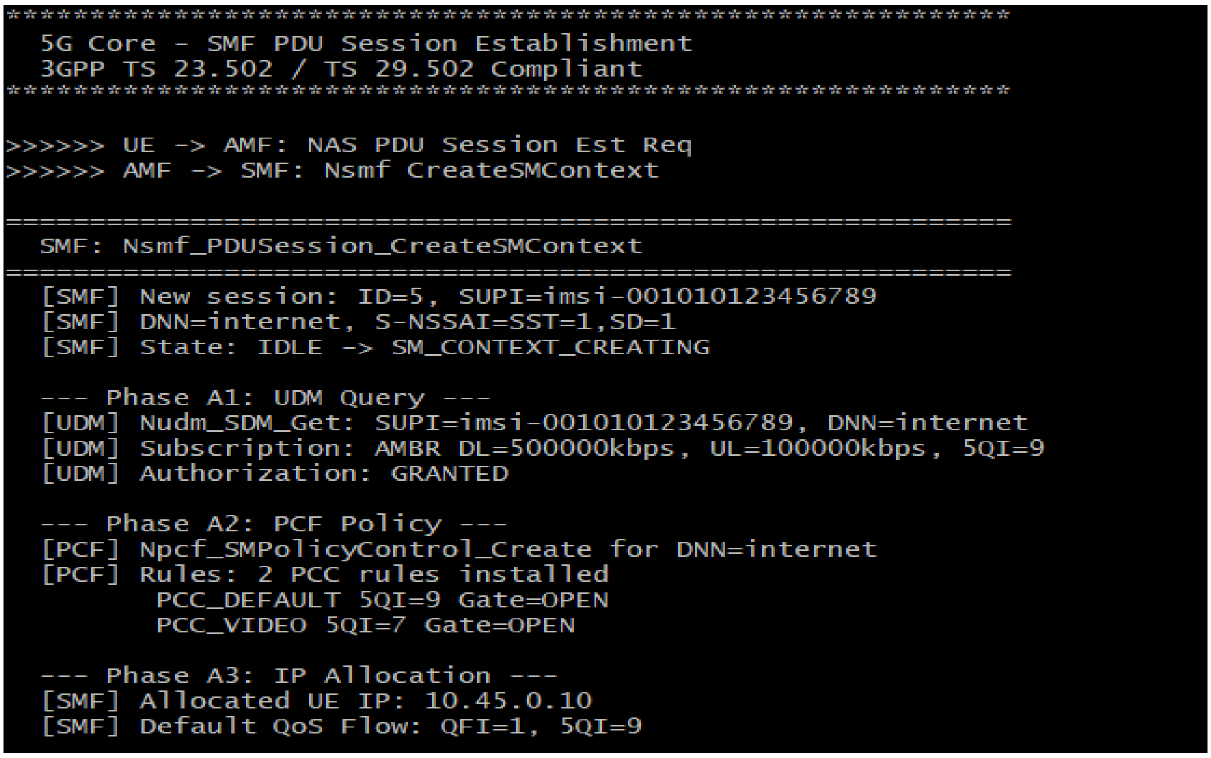


```
// ============================================================
// STEP 5: Main - Complete PDU Session Establishment
// ============================================================
int main() {

    cout << string(60, '*') << endl;
    cout << "  5G Core - SMF PDU Session Establishment" << endl;
    cout << "  3GPP TS 23.502 / TS 29.502 Compliant" << endl;
    cout << string(60, '*') << endl;

    SMF smf;

    cout << endl;
    cout << ">>>>>> UE -> AMF: NAS PDU Session Est Req" << endl;
    cout << ">>>>>> AMF -> SMF: Nsmf CreateSMContext" << endl;

    SNSSAI slice = {1, 1};

    bool created = smf.create_sm_context(
        "imsi-001010123456789",
        5,
        "internet",
        slice,
        PDU_IPv4v6,
        INITIAL_REQUEST
    );

    if (!created) {
        cout << "PDU Session Creation FAILED!" << endl;
        return 1;
    }

    cout << endl;
    cout << ">>>>>> AMF -> gNB: PDU Session Resource Setup Request" << endl;
    cout << ">>>>>> gNB -> UE: RRC Reconfiguration (DRB setup + NAS Accept)" << endl;
    cout << ">>>>>> UE -> gNB: RRC Reconfig Complete" << endl;
    cout << ">>>>>> gNB -> AMF: Resource Setup Resp" << endl;
    cout << ">>>>>> AMF -> SMF: UpdateSMContext (gNB TEID)" << endl;

    FTEID gnb_teid = {0x2001, "192.168.1.50"};
    smf.update_sm_context(5, gnb_teid);

    smf.print_session(5);

    cout << endl << "Simulation Complete." << endl;
    return 0;
}
```

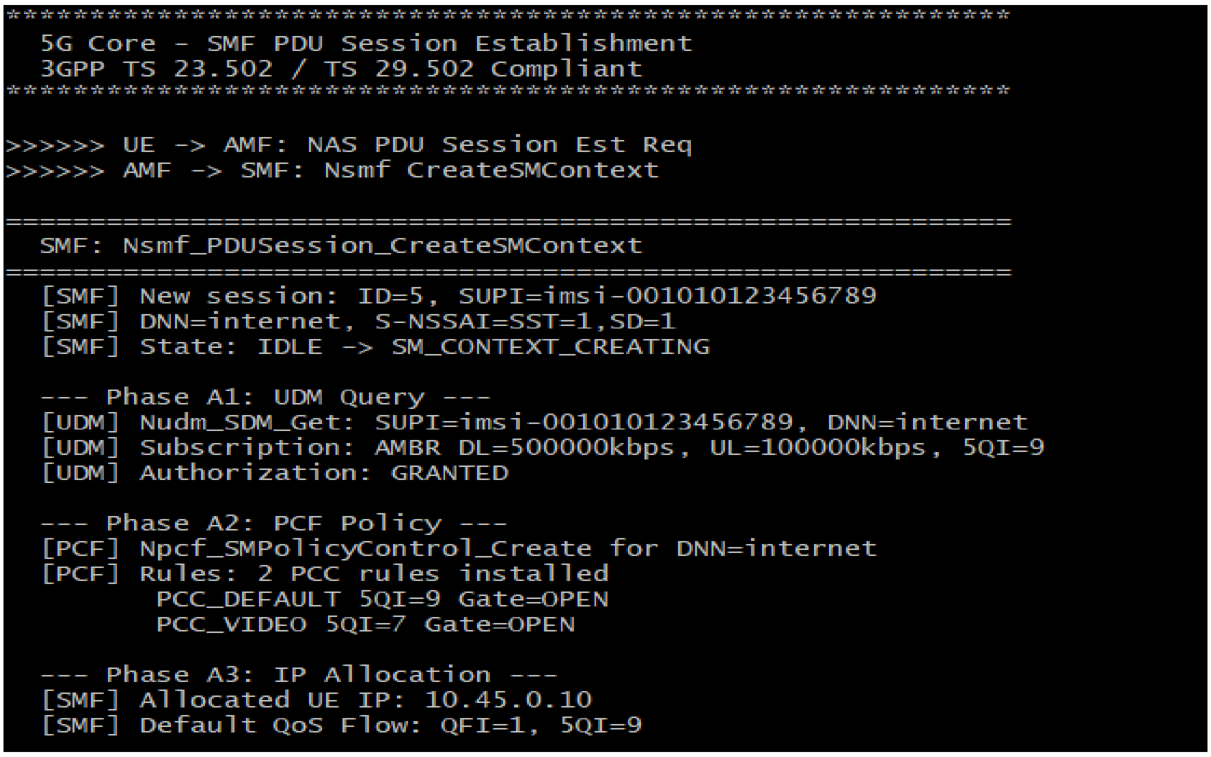
6. SAMPLE OUTPUT

The sample output shows the complete execution flow of the SMF PDU Session Establishment simulation. It displays the sequence of interactions between UE, AMF, SMF, UDM, PCF, and UPF, including subscription verification, policy creation, IP allocation, QoS flow setup, and PFCP session establishment. The output also shows SMF state transitions as the session moves from IDLE to ACTIVE, indicating successful control plane processing.

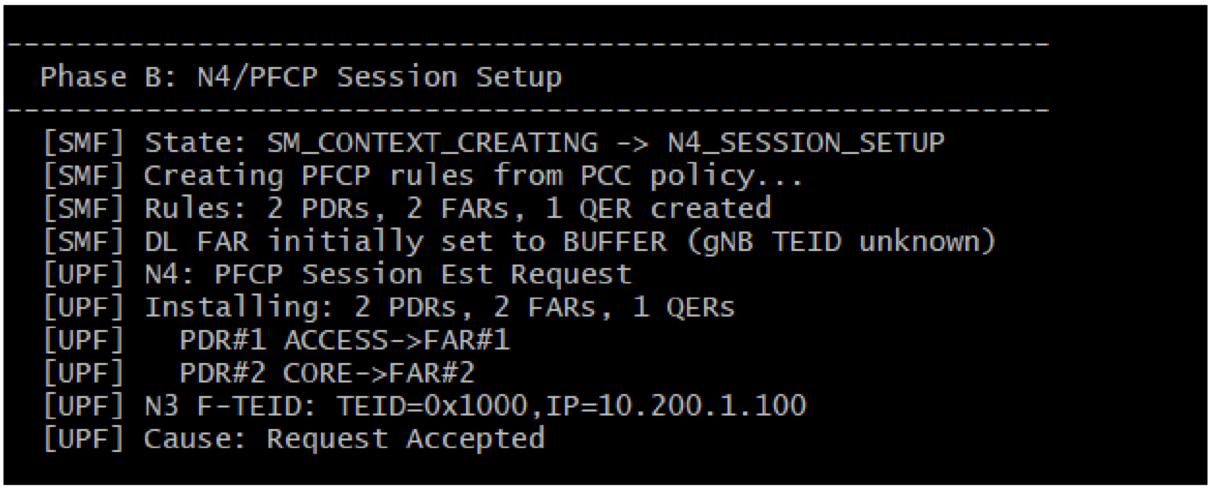
During the later stages, the output shows N1N2 message transfer to AMF, reception of gNB tunnel information, and PFCP session modification where the downlink FAR is updated from BUFFER to FORWARD, activating the user plane path. Finally, a PDU session summary is printed showing session details, tunnel information, QoS flows, and installed PFCP and PCC rules, confirming successful session establishment.



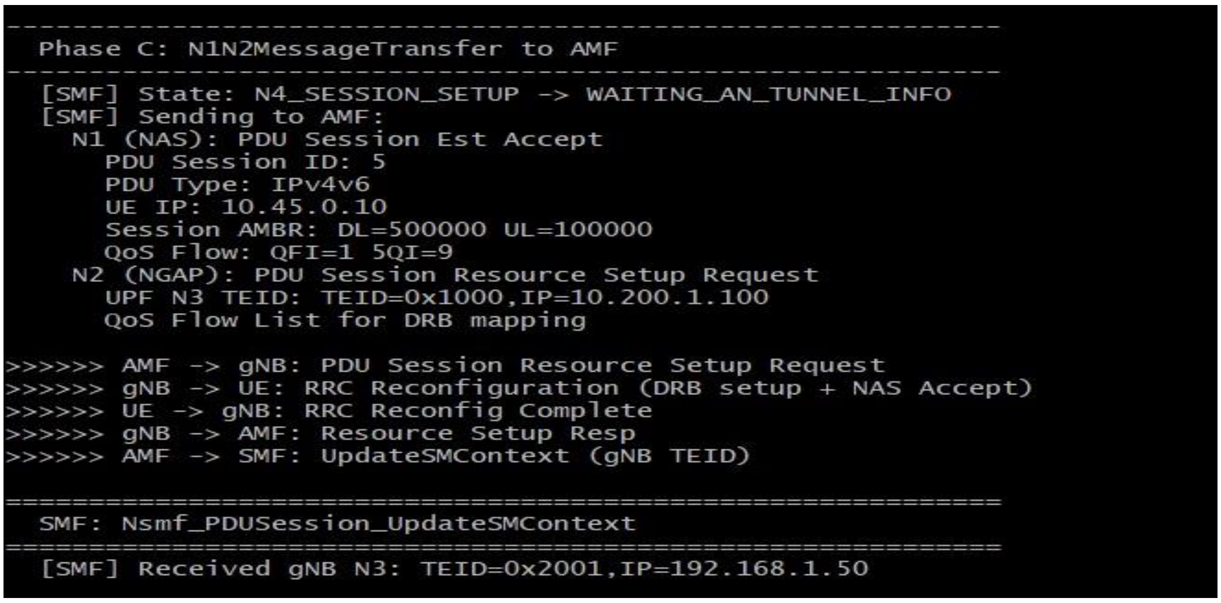
```
************************************************************
  5G Core - SMF PDU Session Establishment
  3GPP TS 23.502 / TS 29.502 Compliant
************************************************************

>>>>>> UE -> AMF: NAS PDU Session Est Req
>>>>>> AMF -> SMF: Nsmf CreateSMContext

============================================================
  SMF: Nsmf_PDUSession_CreateSMContext
============================================================
  [SMF] New session: ID=5, SUPI=imsi-001010123456789
  [SMF] DNN=internet, S-NSSAI=SST=1,SD=1
  [SMF] State: IDLE -> SM_CONTEXT_CREATING

  --- Phase A1: UDM Query ---
  [UDM] Nudm_SDM_Get: SUPI=imsi-001010123456789, DNN=internet
  [UDM] Subscription: AMBR DL=500000kbps, UL=100000kbps, 5QI=9
  [UDM] Authorization: GRANTED

  --- Phase A2: PCF Policy ---
  [PCF] Npcf_SMPolicyControl_Create for DNN=internet
  [PCF] Rules: 2 PCC rules installed
         PCC_DEFAULT 5QI=9 Gate=OPEN
         PCC_VIDEO 5QI=7 Gate=OPEN

  --- Phase A3: IP Allocation ---
  [SMF] Allocated UE IP: 10.45.0.10
  [SMF] Default QoS Flow: QFI=1, 5QI=9
```



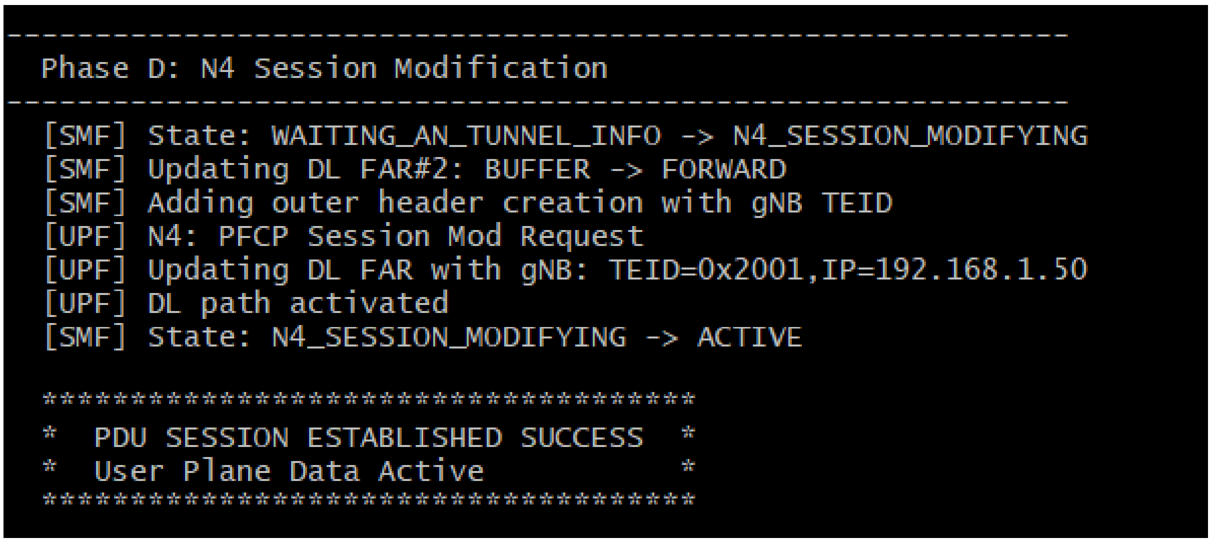
```
************************************************************
  5G Core - SMF PDU Session Establishment
  3GPP TS 23.502 / TS 29.502 Compliant
************************************************************

>>>>>> UE -> AMF: NAS PDU Session Est Req
>>>>>> AMF -> SMF: Nsmf CreateSMContext

============================================================
  SMF: Nsmf_PDUSession_CreateSMContext
============================================================
  [SMF] New session: ID=5, SUPI=imsi-001010123456789
  [SMF] DNN=internet, S-NSSAI=SST=1,SD=1
  [SMF] State: IDLE -> SM_CONTEXT_CREATING

  --- Phase A1: UDM Query ---
  [UDM] Nudm_SDM_Get: SUPI=imsi-001010123456789, DNN=internet
  [UDM] Subscription: AMBR DL=500000kbps, UL=100000kbps, 5QI=9
  [UDM] Authorization: GRANTED

  --- Phase A2: PCF Policy ---
  [PCF] Npcf_SMPolicyControl_Create for DNN=internet
  [PCF] Rules: 2 PCC rules installed
         PCC_DEFAULT 5QI=9 Gate=OPEN
         PCC_VIDEO 5QI=7 Gate=OPEN

  --- Phase A3: IP Allocation ---
  [SMF] Allocated UE IP: 10.45.0.10
  [SMF] Default QoS Flow: QFI=1, 5QI=9
```



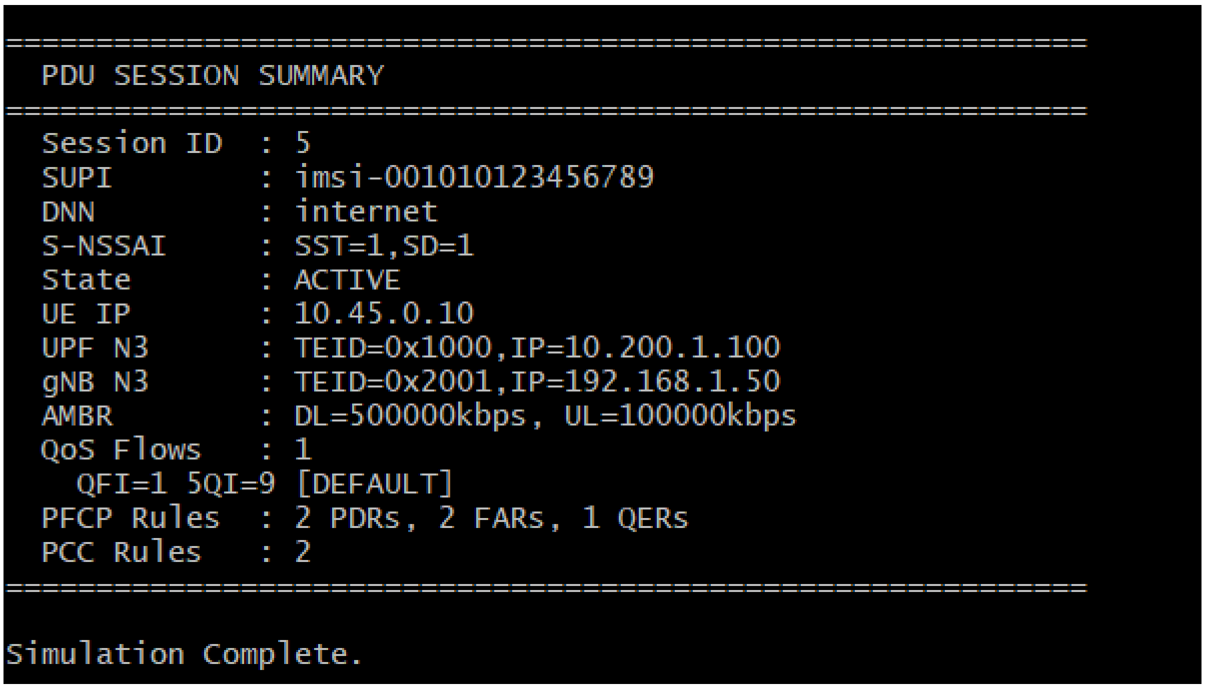
```
------------------------------------------------------------
  Phase B: N4/PFCP Session Setup
------------------------------------------------------------
  [SMF] State: SM_CONTEXT_CREATING -> N4_SESSION_SETUP
  [SMF] Creating PFCP rules from PCC policy...
  [SMF] Rules: 2 PDRs, 2 FARs, 1 QER created
  [SMF] DL FAR initially set to BUFFER (gNB TEID unknown)
  [UPF] N4: PFCP Session Est Request
  [UPF] Installing: 2 PDRs, 2 FARs, 1 QERs
  [UPF]   PDR#1 ACCESS->FAR#1
  [UPF]   PDR#2 CORE->FAR#2
  [UPF] N3 F-TEID: TEID=0x1000,IP=10.200.1.100
  [UPF] Cause: Request Accepted
```



```
------------------------------------------------------------
  Phase C: N1N2MessageTransfer to AMF
------------------------------------------------------------
  [SMF] State: N4_SESSION_SETUP -> WAITING_AN_TUNNEL_INFO
  [SMF] Sending to AMF:
    N1 (NAS): PDU Session Est Accept
      PDU Session ID: 5
      PDU Type: IPv4v6
      UE IP: 10.45.0.10
      Session AMBR: DL=500000 UL=100000
      QoS Flow: QFI=1 5QI=9
    N2 (NGAP): PDU Session Resource Setup Request
      UPF N3 TEID: TEID=0x1000,IP=10.200.1.100
      QoS Flow List for DRB mapping

>>>>>> AMF -> gNB: PDU Session Resource Setup Request
>>>>>> gNB -> UE: RRC Reconfiguration (DRB setup + NAS Accept)
>>>>>> UE -> gNB: RRC Reconfig Complete
>>>>>> gNB -> AMF: Resource Setup Resp
>>>>>> AMF -> SMF: UpdateSMContext (gNB TEID)

============================================================
  SMF: Nsmf_PDUSession_UpdateSMContext
============================================================
  [SMF] Received gNB N3: TEID=0x2001,IP=192.168.1.50
```



```
------------------------------------------------------------
  Phase D: N4 Session Modification
------------------------------------------------------------
  [SMF] State: WAITING_AN_TUNNEL_INFO -> N4_SESSION_MODIFYING
  [SMF] Updating DL FAR#2: BUFFER -> FORWARD
  [SMF] Adding outer header creation with gNB TEID
  [UPF] N4: PFCP Session Mod Request
  [UPF] Updating DL FAR with gNB: TEID=0x2001,IP=192.168.1.50
  [UPF] DL path activated
  [SMF] State: N4_SESSION_MODIFYING -> ACTIVE

  *************************************
  *  PDU SESSION ESTABLISHED SUCCESS  *
  *  User Plane Data Active           *
  *************************************
```



```
============================================================
  PDU SESSION SUMMARY
============================================================
  Session ID  : 5
  SUPI        : imsi-001010123456789
  DNN         : internet
  S-NSSAI     : SST=1,SD=1
  State       : ACTIVE
  UE IP       : 10.45.0.10
  UPF N3      : TEID=0x1000,IP=10.200.1.100
  gNB N3      : TEID=0x2001,IP=192.168.1.50
  AMBR        : DL=500000kbps, UL=100000kbps
  QoS Flows   : 1
    QFI=1 5QI=9 [DEFAULT]
  PFCP Rules  : 2 PDRs, 2 FARs, 1 QERs
  PCC Rules   : 2
============================================================

Simulation Complete.
```

7. KEY CONCEPTS

1. PDU Session

A PDU Session is a logical tunnel between the UE and a specific Data Network (DN). It provides the UE with an IP address and enables user data to flow with defined QoS. Each session is identified by a PDU Session ID, DNN, and S-NSSAI.

2. Session Management Function (SMF)

The SMF is the control-plane function responsible for creating, modifying, and releasing PDU Sessions. It selects the UPF, allocates IP addresses, enforces QoS and policy rules, and controls the user plane using PFCP.

3. User Plane Function (UPF)

The UPF is the data-plane function that forwards user traffic between the RAN and the Data Network. It applies QoS enforcement, packet routing, and charging rules configured by the SMF.

4. DNN (Data Network Name)

The DNN identifies the external network the UE wants to connect to, such as the Internet, IMS, or a private enterprise network.

5. S-NSSAI (Network Slice Identifier)

S-NSSAI identifies the network slice requested by the UE. It allows different services (eMBB, URLLC, mMTC) to run with different performance characteristics.

6. QoS Flow

A QoS Flow is the finest granularity of QoS control in 5G. The SMF creates QoS flows based on PCF policies and maps them to Data Radio Bearers in the RAN.

7. PFCP (N4 Interface)

PFCP is the protocol used between SMF and UPF to install and manage user-plane rules such as PDR, FAR, QER, and URR.

8. Session AMBR

Session AMBR defines the maximum aggregated bit rate allowed for a PDU Session.

9. NAS SM Signaling

NAS Session Management signaling carries PDU Session control messages between the UE and AMF.

10. Service-Based Architecture (SBA)

The 5G Core uses SBA, where network functions expose services via APIs. The SMF consumes and provides services to AMF, PCF, UDM, and UPF.

8. COMPILATION & EXECUTION GUIDE

8.1 Prerequisites :

This simulation requires a C++11 (or later) compatible compiler. Recommended compilers include GCC 7+, Clang 5+, or MSVC 2017+. The program uses only the C++ Standard Library, so no external libraries are required.

8.2 Compilation Commands

- Windows (MinGW / Git Bash):

```
g++ code_final.cpp -o Test2_SMF
./Test2_SMF.exe
```

- Linux / macOS:

```
g++ code_final.cpp -o Test2_SMF
./Test2_SMF
```

- Windows (MSVC):

```
cl /EHsc code_final.cpp
/Fe:Test2_SMF.exe
```

9. CONCLUSION :

The Session Management Function (SMF) plays a central role in the 5G Core by controlling the entire lifecycle of PDU Sessions, which are the foundation of user data connectivity in the 5G system. From allocating IP addresses and selecting the appropriate UPF to enforcing QoS and policy rules, the SMF ensures that each UE receives reliable, secure, and optimized access to the required Data Networks. Through standardized procedures defined in 3GPP specifications, the SMF coordinates seamlessly with AMF, PCF, UDM, and UPF to establish, maintain, and release PDU Sessions efficiently.

By enabling flexible session control, network slicing, and service-based interactions, the SMF makes 5G networks scalable and adaptable to diverse use cases such as enhanced mobile broadband, ultra-reliable low-latency communications, and massive IoT. In essence, the SMF acts as the control-plane “brain” of 5G data services, ensuring that user traffic flows smoothly from the UE to the Data Network and back with the required performance and quality of service.