April 2022

IOT SMART CONTRACT ENABLEMENT WITH TEMPLATE BASED APPROACH

Dennis Lanov

IOT SMART CONTRACT ENABLEMENT WITH TEMPLATE BASED APPROACH

AUTHORS:
Dennis Lanov

ABSTRACT

Techniques are described herein for "natively" deploying smart contracts into 5G networks. This may enable Internet of Things (IoT) devices to sign-up the smart contracts with 5G to deliver services and features provided by the 5G network to IoT devices on a per-contract basis.

DETAILED DESCRIPTION

By 2025, there will be more than 27 billion IoT connections in the world. The mass deployment of 5G connections will bring 5G to IoT devices and drive new IoT use cases.

Service Providers (SP) want to launch smart contract enabled IoT devices natively. The SPs want to enable native IoT User Equipment (UE) to connect securely to their 5G networks, using a distributed ledger for buying a service from the 5G network. The 5G network should be able to offer smart contracts to the IoT devices. Mission Critical IoT devices, for example, require specific QoS provided by the 5G network.

The techniques described herein are built on how to enable smart contract capabilities in a 5G network. In one example, a smart contract technology may be deployed "natively" into the 5G network. This may also enable smartphone devices and IoT devices to participate in buying smart contracts from the 5G network and drive new use cases. Due to the specific requirements of mission-critical IoT devices, penalties are introduced for non-performing 5G networks and the QoS monitoring procedure may be exposed with smart contracts.

As described herein, a template approach is used in smart contracts between IoT devices and the 5G network. The template includes programming functions for the charging, policy, and subscription elements of the 5G network. The programming functions (code) are inserted into corresponding Network Functions (nodes) in the 5G network for execution.

The smart contract includes a penalty clause for when the 5G network does not fulfill the terms of the smart contract specifically for the QoS requirements. The 5G network functions directly report the results of the distributed ledger providers (e.g., Blockchain, Ethereum, IOTA, etc.) as corresponding programming functions are being processed.

The contract is immutable because it is registered on the distributed ledger. Approval by the IoT owner may be introduced for the purpose of billing and controlling how the payment comes from the IoT owner account. All parties of the contract (e.g., IoT device, 5G Network Functions, and any third party such as IoT owner, clearing houses, etc.) may have real-time visibility of the fulfillment of the contract. The template approach may enable programming on the IoT side as well as the 5G network side.

A pseudocode example of the smart contract is provided below. As shown, the UpgradeQoS contract has subscription, charging, and policy information parameters. Initially (start() function), the contract calls the 5G network elements to update the current subscription, policy, and charging information based on the values in the contract. Later, after the duration of the contract is expired, the end() function checks whether the subscription, policy, and charging requirements were met at the requested values, and if not, discounts the charge with a penalty fee. For simplification purposes, the charge and penalty are defined as constant values, but in real-life scenarios they can be implemented as variables, too.

Compiler version
Libraries
Interfaces
Sub-contracts

```
contract UpgradeQoS{
   bool public started;
   unit public endedAt;
   new_5G_QoS_parameters public
```

```
new_5G_subscription public
new_5G_policy public
duration public
total_traffic public
IOT_active public
address device_owner
timestamp
unit public agreed_charge
unit public penalty

event Start(unit32 startAt, unit32 endedAt)

event End(unit amount)


constructor{
   unit256 charge
   unit penalty
   address account
   unit charges
}{
   payment = payable(msg.account)
}


function update_policy() public view returns(boolean){
   // Update policy function executed on PCF
   // Executed by policy side to set QoS values for the flow or/and bearer

}
function query_policy() public view returns(unit256){
```

```
        // Update policy function readable on PCF
        // Executed on policy side to query current QoS values

    }
    function update_charging() public view returns(boolean){
        // Charging function executed on CHF
        // Executed by charging function current traffic usage

    }
    function query_chargin() public view returns(unit256){
        // Charging function executed on CHF
        // Executed on charging function to retrieve current traffic usage
        _total_traffic

    }
    function update_subscription() public view returns(boolean){
        // Subscription function executed on UDR side and propagated to AMF, NSSF
and SMF
        // Update Slice information, PDU information

    }
    function query_subscription() public view returns(unit256){
        // Subscription function executed on SMF side to get current slice information,
PDU information

    }
    function start() external{
        update_subscription() ;
        update_charging();
        update_policy();
        started = True
```

```
    endedAt = timestamp + duration;
    emit Start(_timestamp, _endedAt);
  }

  function end() external{
    require started True;
    require timestamp >= endedAt or traffic >= total_traffic
    if (query_subscription() == 5G_QoS_parameters){
      charge = agreed_charge
    } else {
      charge = agreed_charge - penalty
    }
    if (query_subscription() == 5G_subscription){
      charge = charge
    } else {
      charge = agreed_charge - penalty
    }
    if (query_policy() == new_5G_policy){
      charge = charge
    } else {
      charge = agreed_charge - penalty
    }
    emit End(charge)
  }
}
```

Figure 1 below illustrates a smart contract processing flow. As shown, the IoT requests a list of available contracts from the Application Function (AF) and selects one. The contract is registered with a Distributed Ledger and signed by the AF (5G network) and the IoT device. The AF propagates the updated information to the 5G network elements (e.g., Charging Function (CHF), Policy Control Function (PCF), and Session Management

Function (SMF)). The 5G network elements receive the corresponding programming part of the smart contract for execution. The CHF, PCF, and SMF directly report the corresponding parameters back to the Distributed Ledger.

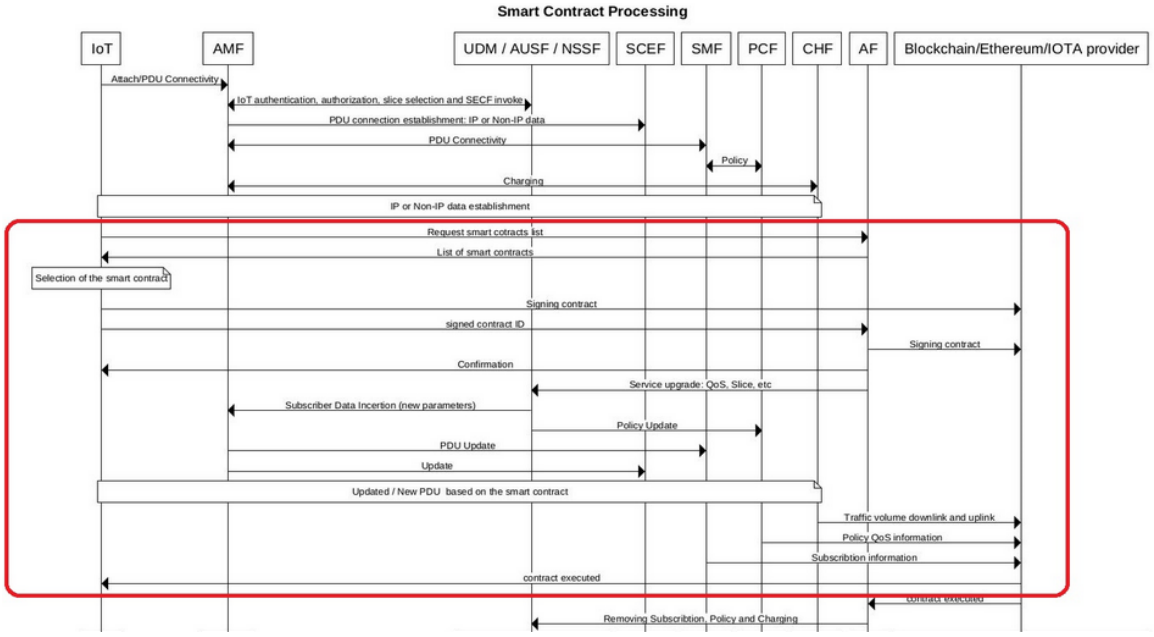

Figure 1

Using this solution, IoT devices may sign their agreements directly with the 5G network without requesting new network slicing. Network slicing can be an option for the smart contract, but need not necessarily be mandatory. In many cases, only a new QoS policy is needed from the IoT device side. Additionally, penalties may be enabled for non-providing requests in the smart contract QoS. This may provide final charges to the IoT owner when the contract is completed. For example, if an IoT device for earthquakes needs to send a live camera feed for a short time, it may buy a smart contract to enable the live camera feed to a server.

As described herein, smart contracts may be enabled "natively" in a 5G network without introducing additional elements in the network. Network slicing may not be required to change the QoS of the flow. The QoS of the device communication flow may be updated without changing its slice via a policy update flow available for 4G and 5G devices. Smart contracts may be provided to existing customers in the 5G network to provide a particular QoS on demand.

In summary, techniques are described herein for "natively" deploying smart contracts into 5G networks. This may enable Internet of Things (IoT) devices to sign-up the smart contracts with 5G to deliver services and features provided by the 5G network to IoT devices on a per-contract basis.