Université Abdelmalek Essaâdi
Ecole Nationale des Sciences Appliquées - Al-Hoceima
Département Mathématiques et Informatique

Filière: Data Engineering
Module: Advanced Big Data

Project

Telecom Data Pipeline for Mediation, Rating and Billing Process

Proposed by: Mohamed El Marouani

April 2025

I. Introduction

The objective of this project is the hands-on application of the knowledge acquired in the Big Data course in a practical and industrial use case. This project consists to the implementation of a **simplified process of mediation, rating and billing in a telecom operator**.

In the context of an academic project, we will describe in this document the context, the functional requirements, the technical environment, and the expected deliverables.

II. Context

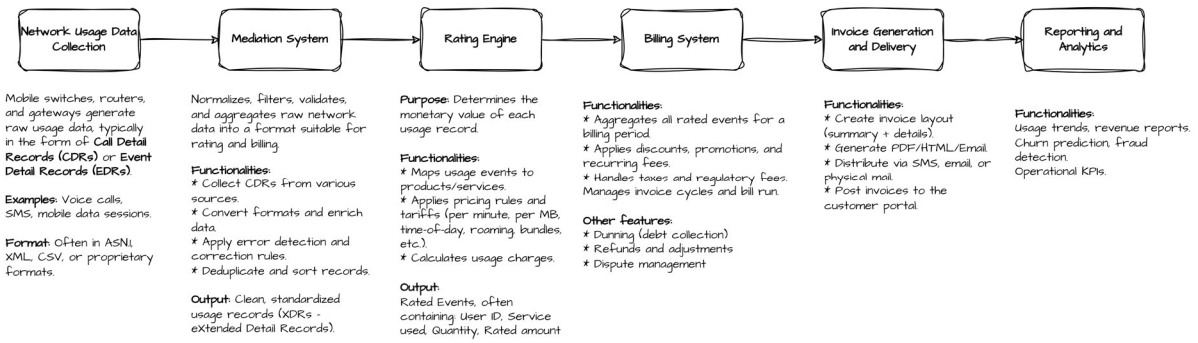
In the telecommunications industry, the processes of mediation, rating, and billing are critical for ensuring accurate and timely revenue collection. These processes enable the operator to transform raw usage data collected from network systems into billable records, calculate charges based on complex pricing models, and generate customer invoices.

Typically, this involves a series of intricate steps:

- Collecting and normalizing call and data usage records (**collecting & mediation**);
- Applying pricing rules and tariffs to compute charges (**rating**);
- Aggregating these charges to produce and dispatch bills (**billing & invoice delivery**);
- In addition of a transverse layer where we visualize reports and insights about the whole process (**reporting & analytics**).

Due to the high volume of data and real-time requirements, telecom systems employ robust and scalable Big Data architectures.

The real process can be simplified and schematized as follows:



In order to frame the project in the academic context, we will proceed to a set of simplifications:

- The collection component will be replaced by **synthetic data generation engine** where we will simulate the results of usage tickets provided by the several network nodes;
- We will consider the rating/billing process only in the context of post-billing scenario (not online charging scenario);
- We suppose that the telco operator have only a mobile network infrastructure (not fixed or optical fiber networks)
- We consider that the collection and mediation steps will be implemented in a streaming mode, and the rating and billing steps will be implemented a batch mode.

III. Functional requirements

The system to be implemented must reproduce the core logic of a postpaid telecom billing process in a simplified and organized manner. The functional requirements of the project are structured around the key components of the mediation, rating, and billing chain, as well as the reporting layer:

1. **Synthetic Data Generation**:

The system must include a component to generate synthetic usage records (e.g., voice calls, SMS, mobile data sessions). These records should reflect realistic usage patterns across different customers and services, and be formatted similarly to real network usage tickets.

In a real telecom environment, usage data is captured in the form of Call Detail Records (CDRs) or Event Detail Records (EDRs). These records are automatically generated by network nodes (e.g., switches, gateways, charging systems) to log user activity such as phone calls, SMS, data sessions, etc. Each record typically includes technical and business attributes describing the event: timestamp, duration, caller/callee IDs, service type, data volume, cell ID, charging context, and more.

For the purpose of this project, a synthetic data generation module will be developed to simulate these network usage records. The engine should create records for various telecom services, for example:

Voice call CDR

```
{
  "record_type": "voice",
  "timestamp": "2025-04-18T14:32:15Z",
  "caller_id": "212612345678",
  "callee_id": "212698765432",
  "duration_sec": 180,
  "cell_id": "ALHOCEIMA_23",
  "technology": "3G"
}
```

SMS CDR

```
{
  "record_type": "sms",
  "timestamp": "2025-04-18T14:35:02Z",
  "sender_id": "212612345678",
  "receiver_id": "212678901234",
  "cell_id": "ALHOCEIMA_23",
  "technology": "4G"
}
```

Data session EDR

```
{
  "record_type": "data",
  "timestamp": "2025-04-18T14:40:00Z",
  "user_id": "212612345678",
  "data_volume_mb": 150.75,
  "session_duration_sec": 240,
```

```
  "cell_id": "IMZOUREN_10",
  "technology": "LTE"
}
```

Simulated records should reflect different mobile technologies:

- 2G, 3G, 4G, 5G
- Differing behavior in data volume, duration, or availability of fields depending on the technology (e.g., 2G may lack high data volume events)

The generated records can be enriched with other attributes in order to get closer to the real data generated by the telecom networks.

To reflect real-world scenarios and test system robustness, the data generator should include:

- Missing fields (e.g., `callee_id` missing in some records)
- Corrupted data (e.g., negative duration, non-numeric phone numbers)
- Duplicate records
- Out-of-order timestamps
- Unrecognized or unsupported service types

Example corrupted record:

The data generation module should allow users to control:

- Volume of data (e.g., 10,000 events/hour)
- Ratio of errors or anomalies (e.g., 5% corrupted records)
- Distribution of service types (e.g., 60% voice, 30% data, 10% SMS)

Generated records can be output as:

- JSON or CSV files for batch ingestion
- Streamed records using tools like Kafka for real-time ingestion

This synthetic data engine is crucial for mimicking realistic telecom operations while giving users flexibility to test their pipelines and error-handling logic in both streaming (mediation) and batch (rating/billing) components.

2. **Streaming-Based Mediation**:

A streaming pipeline must be implemented to ingest, parse, and normalize usage records in real time. This component should handle basic data validation, filtering, and standardization of input records into a common format for downstream processing.

The rules to implement in this component can cover the following:

- **Data normalization:** normalize all types of records (voice, SMS, data) into a common schema, rename fields to common names, standardize units (e.g., MB, seconds), add missing fields as `null, …`
- **Filtering Rules:** drop records where `msisdn` is missing or invalid, drop test/dummy numbers (e.g., starting with `999`), …

- **Duplication Detection:** Implement a deduplication mechanism where you keep a cache of processed `record_ids` and if a record with the same ID appears, you discard or log it.
- **Error Handling:** If a record is malformed or fails validation: tag it with status = error and send it to a dead letter topic or log for analysis.
- **Output Transformation:** Send clean, enriched records to the next system (rating engine) in a streaming manner.

3. **Rating Engine (Batch)**:

A batch process must be developed to apply pricing logic to normalized usage records. The rating engine should compute the cost of each event based on predefined tariffs that vary by service type (e.g., per-minute call charge, per-MB data cost).

We expose in the following a detailed set of rules and mechanisms that can be implemented in the Rating Component to realistically simulate a professional telecom billing system. These rules concern a number of aspects:

Customer Database

The system should maintain a customer table with attributes such as:

- `customer_id` (unique)
- `customer_name`
- `subscription_type`: `prepaid` or `postpaid` (limit to `postpaid` for this project)
- `rate_plan_id`
- `activation_date`
- `status`: `active`, `suspended`, `terminated`
- `region` or `zone`

This database determines which rate plan applies and whether the customer is eligible for rating (only active, postpaid customers).

Product Catalog

Define a list of telecom services offered:

- Voice calls (national / international)
- SMS
- Data usage (MB or GB)
- Optional: Roaming, value-added services

Each product includes:

- `product_code`
- `service_type` (e.g., voice, sms, data)
- `unit` (minute, SMS, MB)
- `rate_type` (flat, tiered, time-based)

This catalog connects each usage record to a known service and its rating logic.

Rate Plans

Each customer subscribes to a rate plan that defines how services are charged. A plan includes:

- `rate_plan_id`
- `plan_name`
- List of `product_rates`, for each service:
  - `service_type`: voice/sms/data
  - `unit_price`: e.g., 0.01 MAD per second, 0.05 MAD per SMS
  - Optional: free units included per cycle (e.g., 100 minutes)
  - Optional: tiered pricing (first 100 MB at 0.05 MAD/MB, next at 0.1 MAD/MB)

The rate plans define how the cost is calculated per usage event.

Rating Logic

The rating component processes normalized usage records and applies pricing rules:

Example rules:

- Voice call: cost = duration_sec × price_per_second
- SMS: flat rate per event
- Data session: cost = data_volume_mb × price_per_mb
- Apply rounding rules (e.g., round call duration to next minute)

Time-Based Modifiers

Some rate plans vary based on time of day or day of week:

- Peak/off-peak pricing
- Weekend offers

Location-Based Pricing

Optional: simulate different zones (urban/rural or local/international):

- Domestic call → standard rate
- International call → premium rate
- Roaming scenario → extra charges (e.g., `base_rate × 2.0`)

Discounts & Promotions

Simulate some special cases:

- First 50 SMS are free per month
- 10% discount for students
- Loyalty discounts (e.g., if `subscription_age > 12 months`, apply 5% discount)

Error & Exception Handling

Implement checks before rating:

- If customer ID is unknown → reject
- If record has invalid/missing values → flag as "`unratable`"
- If customer is not active → mark event as "`ignored`"
- Maintain a `rating_status` flag for each event: `rated`, `rejected`, `error`, `unmatched`.

4. **Billing Engine (Batch)**:

Another batch component must aggregate rated usage records per customer, summarize charges by billing cycle, and generate a billing summary (invoice) per user. The billing logic must also handle basic account information such as customer ID and billing profile.

This component can have as inputs a set of elements such as:

- **Rated Usage Events** (from the rating engine): enriched with cost, timestamps, customer ID, service type, etc.
- **Customer Information**: subscription details, billing preferences, contact data.
- **Rate Plans and Rules**: to retrieve free unit balances or billing thresholds.
- **External Parameters**: tax rates, billing cycle definitions, penalties, etc.

Billing Cycle Grouping

- Group all rated events by `customer_id` and `billing_period` (e.g., monthly).
- Only include events with `rating_status = 'rated'`.

Free Unit Allowance / Quotas

If the rate plan includes free usage (e.g., 100 minutes/month), the billing system must:

- Deduct rated volume from available quota.
- Apply charges only for excess usage.

Discounts and Promotions

Apply discounts after computing the total:

- Percentage discount (e.g., -10% for loyalty program)
- Flat reductions (e.g., -20 MAD off for a promotion)

Apply them conditionally:

- Customer segment
- Service usage thresholds
- Billing month (e.g., seasonal offer)

Taxes and Regulatory Charges

After discounts, apply standard telecom taxes:

- VAT rate (e.g., 20%)
- Optional: regulatory fees (e.g., 1 MAD per customer/month)

Invoice Generation

Create a structured billing record for each customer in an XML or JSON format where you store:

- The invoice header: customer data, bill period, …
- The invoice lines that reflect the consumption details.

Then, this billing record can be formatted in a PDF or HTML format to distribute via email or other channels to the customers.

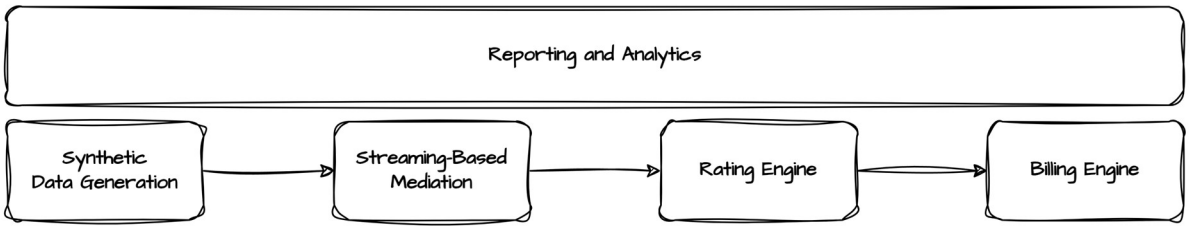
5. **Reporting & Analytics**:

The system must include a dashboard or reporting module that provides insights into the usage data, revenue metrics, customer consumption patterns, and the performance of the mediation and billing processes.

Each of these components must be designed with modularity and scalability in mind, reflecting Big Data principles such as distributed processing, separation of concerns, and fault tolerance, within the constraints of an academic project.

IV. Architecture

The proposed architecture of the implemented data pipeline can be viewed as follows:



This pipeline must be implemented to be deployed in a distributed (cluster) environment.

It is recommended that the solution to implement use this technical stack: Python, Spark, Kafka, and PostgreSQL. You can extend this stack to other tools (frameworks, packages, orchestrators, …), but these choices must be justified.

V. Deliverables

The deliverables of the project will be:

- Code: scripts, notebooks, database, …
- Report or presentation that summarize the different parts of the project.