CALL SCREENING

Defne Call Screening provide operators a solution for fraudulent calls by detecting and preventing fake calls in a mobile terminating way by utilizing Intelligent Networks.

Almost everyone who has a phone is affected by the rise in unsolicited calls. Even more disturbing are the calls in which scammers attempt to target or mislead people while disguising their identities by **spoofing** or changing the caller ID. To provide a better customer experience by analyzing and blocking the fraud calls, Mobile Network Operators need to utilize Fraud detection & blocking services. Defne Call Screening can distinguish between legitimate and unauthorised traffic, block scam calls and prevent Spoofing and Robocalling.

To determine whether to prevent a call or not, examination is made within the operator first, by considering Network, Provisioning and Fraud integration points by Defne Call Screening . If the calling number is a subscriber of another operator within the country, examination is continued by querying the Central Call Screening solution to get the service subscription information of the subscriber to determine whether to allow or prevent the call and prevention is provided by Defne Call Screening if necessary.

Feature set:

- The analysis of calls that overlap and ring repeatedly from the same numbers
- Identification of fake caller ID
- Blacklist supervision
- Validation of number ranges
- Detecting & blocking fraud calls

Defne Call Screening is Kubernetes orchestrated and has Cloud-native architecture. So it comes along with reduced development and release timeframes, optimized costs, increased software scalability and availability, flexibility in multi-cloud environments, and migration from an on-premises environment to hybrid cloud is possible.

USE CASES

Blocking calls if there is no such subscriber in the served operator

All calls will be canceled if there is no such subscriber in the served operator. Defne Call Screening checks following to determine:

- Existence of number portability status information.
- Porting status of the numbers within the served operator's number range (ie. 9054X).

Blocking calls that exceed the predetermined number

The maximum number of calls which can be made from the same number towards a subscriber per day can be determined by regulators and exceeded calls are prevented.

Blocking fraud calls coming from abroad (International interconnects incoming calls)

When a call is coming from abroad, Network examination is required for roaming status check with the purpose of detecting calls made by subscribers who pretend to be abroad but are actually inland.

Blocking calls with invalid Caller IDs.

If there is no caller ID or the caller ID is alphanumeric, incoming calls from domestic and international numbers will be disconnected.

Blocking calls of subscribers whose number is not aligned with the numbering plan

If the subscriber number does not match the numbering plan, all calls will be terminated.

Blocking calls with missing Caller ID.

According to the different caller ID lengths, contents, etc., calls can be cleared. For MO calls, MT calls, and interconnect calls, this feature shall be applicable. While determining whether the caller ID is present or absent, the number of digits for the number types specified in the Numbering Regulation are taken into consideration.

Blocking frequent calls & Fraudulent Call Detection capabilities

Defne CLI finds out calling numbers that are making frequent calls. For instance, a list of the phone numbers that place more than "x" calls in an hour can be retrieved. These kinds of quantities create statistics and alarms. Regulators in the nation can then review these numbers and, if necessary, add them to the list of fraudulent numbers.

Blocking calls of subscribers who have barring services

If the customer has blocking service (e.g. barring of all outgoing calls), all calls will be terminated.

Blocking calls originated from blacklisted numbers

For domestic calls, determination of 'whether the subscriber is blacklisted across the nation or not' is made through Fraud identification. If the subscriber number is in FRAUD lists, the call will be terminated. Fraud number list can be updated manually with GUI interface and by script via standard interfaces like Rest API. Database also contains lists of SERVICE numbers (e.g. white lists).

Blocking calls of personally blacklisted numbers for each subscriber

Blocking outgoing calls of subscribers who have no voice subscription

If the customer has no voice (telephony service, T11) subscriptions, all calls will be terminated.

Blocking fraud calls coming from inland

Network examination is required for roaming status check with the purpose of detecting calls made by the subscribers who pretend to be inland but are actually abroad. By Provisioning examination, a certain set of subscriber registrations are checked such as early call forwarding settings and IMS/TAS registrations. Calls that are initiated in the served operator's country where the subscriber is actually in roaming will be canceled except those conditions.

Defne Call Screening's unique criterias to block calls are listed below:

- Fraud calls coming from abroad (International interconnects incoming calls)
- Fraud calls coming from inland
- Frequent calls
- Calls coming from personally blacklisted numbers of each subscriber
- Calls of subscribers who have barring services
- Outgoing calls of subscribers who have no voice subscription
- Calls of subscribers whose number is not aligned with the numbering plan
- Calls if there is no such subscriber in the served operator
- Calls with invalid caller ID.

The perks of utilizing Call Screening Service

Prevents Voice Fraud

- Monitoring incoming and outgoing calls in accordance with carrier policies can be used as validation procedures for call fraud schemes.

Blocks Connections in Real-time

- To find potential fraud, our technology checks pre-made watch lists against each call. The solution can block the caller and notify the fraud team if a fraudster is found.

Lowers Company Expenses

- Voice fraud is a serious and expanding issue for the telecom sector. A business might quickly lose thousands to millions of dollars due to a single fraud incident.

Minimizes Conflicts with Service Providers

- One of the most prominent inter-carrier fraud cases is voice fraud, which must be exposed with the help of timely information. Monitoring of incoming calls, particularly the path they took to enter the network is essential.

Enhances the Customer Experience

- People are being attacked by criminals from all sides, even their phones. The ultimate impact on voice customer experience is fraud and spam.

Cloud Native Design of Calling Line Identification

Introduction

The main purpose of Calling Line Identification (CLI) is to accomplish secure communication by preventing calls such as Fraud and Spam, as well as incoming calls from numbers that are not registered with a trusted operator. When a call starts to establish between the calling and called party, the Supervisory Authority determines whether both are registered in the database of the home network operator or other operators, using the Counselee & Counselor control mechanism. Counselee and Counselor are a control mechanism located in the Supervisory Authority data center and they have the authority to continue or not the call setup. Each operator has an intelligent network (IN). When an operator receives a call, IN forwards it to Counselee to check whether the called number exists in the operator database. If the number cannot be found in the operator's own database, Counselee requests the Supervisory Authority and the Supervisory Authority checks the databases of other operators with the help of the Counselor. If the Counselor finds the number, it returns a successful response and the Supervisory Authority accepts the call. If the Counselor cannot find the number, the Supervisory Authority rejects the call.

One of the most critical parts of the project is fast response. CLI application must give a response to requests coming from the counselee and counselor within maximum 5 milliseconds. Only Open Source tools have been used in CLI. The used technologies are Redis, Redis Sentinel, Docker, Containerd, Kubernetes, Apache Kafka and Zookeeper.

- Redis Database which is called In-Memory Database preferred due to fulfill required performance. Redis has a very high read/ write speed as it keeps the data on memory.
- Docker and Containerd technologies are used with Kubernetes to create containerized services. The containerized structure provides fast deployment and maintenance.
- Kubernetes are used for the purpose of easily controlling and orchestrating the containers. It has a key role to ensure the system is highly available, reliable, scalable, monitorable and easily manageable.
- Redis sentinel is created to prevent data loss when the Redis master node fails. Redis in cluster structure have data recovery capability and failover mechanisms.
- Apache Kafka, an event streaming application, queues requests from Notification and Provisioning Services and forwards them to other geo-redundant sites, as all sites must have synchronized.
- Zookeeper is used to help high availability of Kafka and selection of Kafka leader.
- Home subscriber's data is stored in a database which is located in the Outsourced Data Center. The Redis database in the CLI and the database in the Outsource Data Center must be synchronized continuously. When a change occurs on one of the database instances both must communicate with each other to update itself. SOAP protocol is used to transport notification packets and Ldap is used to establish a connection to the Outsource database from CLI.

Cloud-Native Network Function (CNF)

In CNFs, Physical Network Functions (PNF) and Virtual Network Functions (VNF) features are included in containers. Unlike VNF, VM software is no longer needed. Because containers can run without the need for a guest OS or hypervisor. In addition, long boot times in VMs with VNF structure are history due to lightweight container technology. CLI services are packaged as Docker Images and then deployed as containerized services or microservices. CLI includes container lifecycle management, agility, resilience, and observability which are cloud-based architectural and operational requirements of the CNF. Services can be managed using only Master Nodes. Being able to manage the whole system from a single source provides great convenience.
itoring Center, and it triggers an alarm.

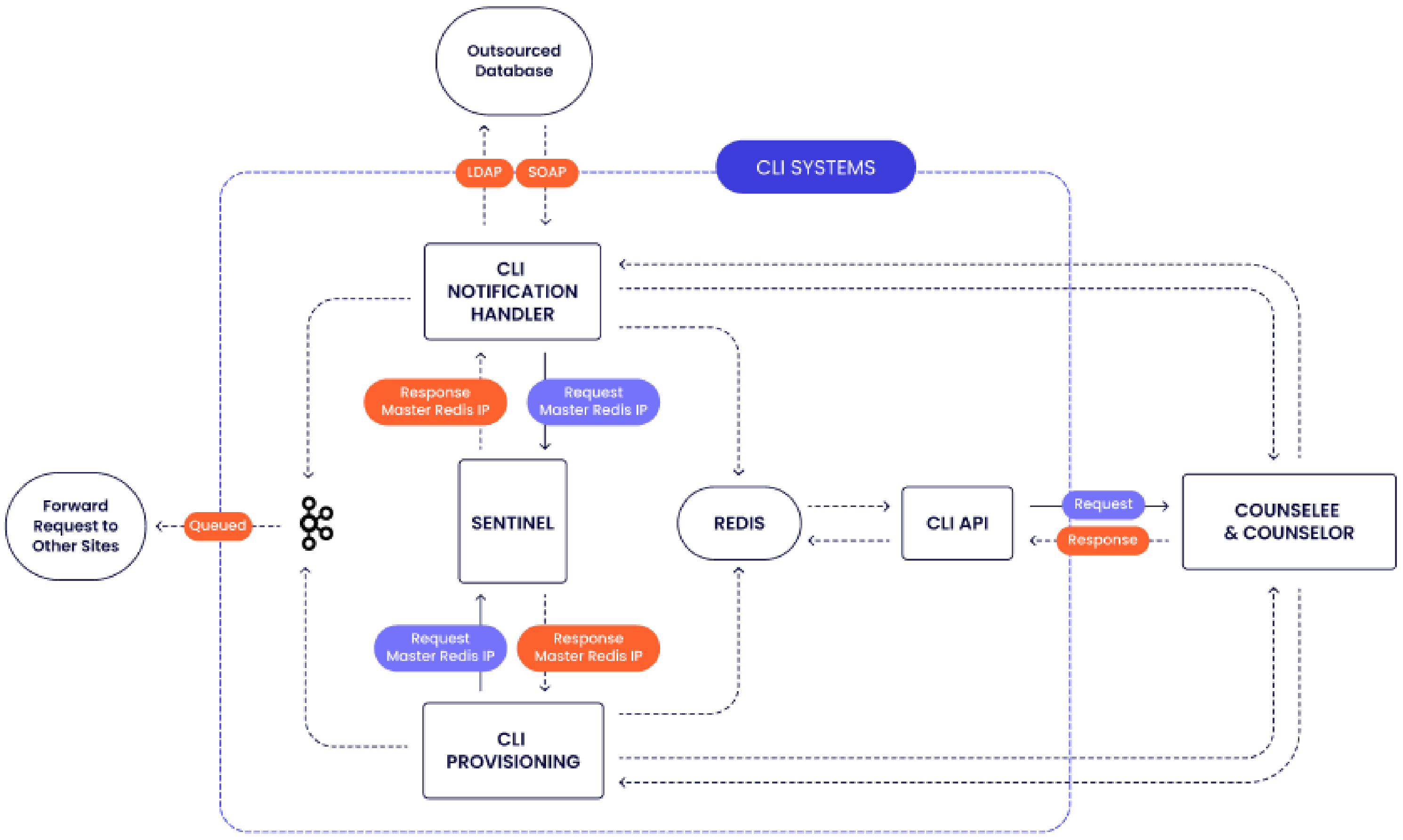
CLI Architecture

In the CLI structure there are 3 core services: CLI API, CLI Provisioning and CLI Notification Handler. These services are working as containerized packages and independently from each other. Therefore CLI services can be called microservices.

1- CLI-API Service: CLI API service is a connection between CLI and Counselee and Counselor in Supervisory Authority. Counselee and Counselor can communicate with Redis via CLI API. Counselee and Counselor sends GET requests to CLI API due to check IMSI or MSISDN of the calling number, whether these numbers are registered in Redis Database. If numbers are registered, CLI API sends a response including subscription data or if not registered then returns an unknown subscriber message.

2- CLI-Provisioning Service: CLI provisioning service is used to execute CRUD operations of subscriber information.

3- CLI-Notification-Handler: CLI Notification Handler Service is an information exchanging service between Outsource Database which stores information related with home subscribers, and CLI. The main functionality of Notification Handler service is to synchronize data among Outsource Database and CLI Redis Database. Notification Service obtains required info from SOAP triggers. But, some of the triggers cannot be read by Notification handler service therefore service connects to the Outsource Database via LDAP and obtains required information. Therefore, continuous network traffic flows on Notification Handler Service. The current configuration uses LDAP, but if messages need to be encrypted LDAPs option is also available. The notification service has a health check mechanism. It periodically checks the Kafka and Redis Pods and the status of the LDAP connection. If there is a problem with any of them, SNMP traps are generated and sent to the Network Monitoring Center, and it triggers an alarm.



General Architecture

Containerized Systems & Kubernetes Orchestration

Microservices vs Monolithic Architecture

A monolithic structure is a traditional architecture and is a single, large system with a single codebase, bringing together all business-related. As such systems grow, they become harder to scale and components cannot be scaled individually. At the same time, a small change to the application affects the entire system and causes increased development and deployment time.

On the contrary, the microservice architecture is structurally a service-oriented architecture, and services are loosely coupled with each other. Loosely coupled means that changes in one of the services minimally affect the performance and operation of other services. Adding a new service or changing an existing service does not affect the whole system, as there is no significant dependency between the services. This gives flexibility to the system, so the Development Team can perform new developments in a shorter time, and thus the Devops Team can deploy new versions more easily and frequently.

The three fundamental services included in the CLI structure are called the CLI microservices. Any development or update on the Provisioning service takes place independently of the other services. A new service can be added in the future without disrupting the general structure of the system, and it does not affect the operation of existing services.

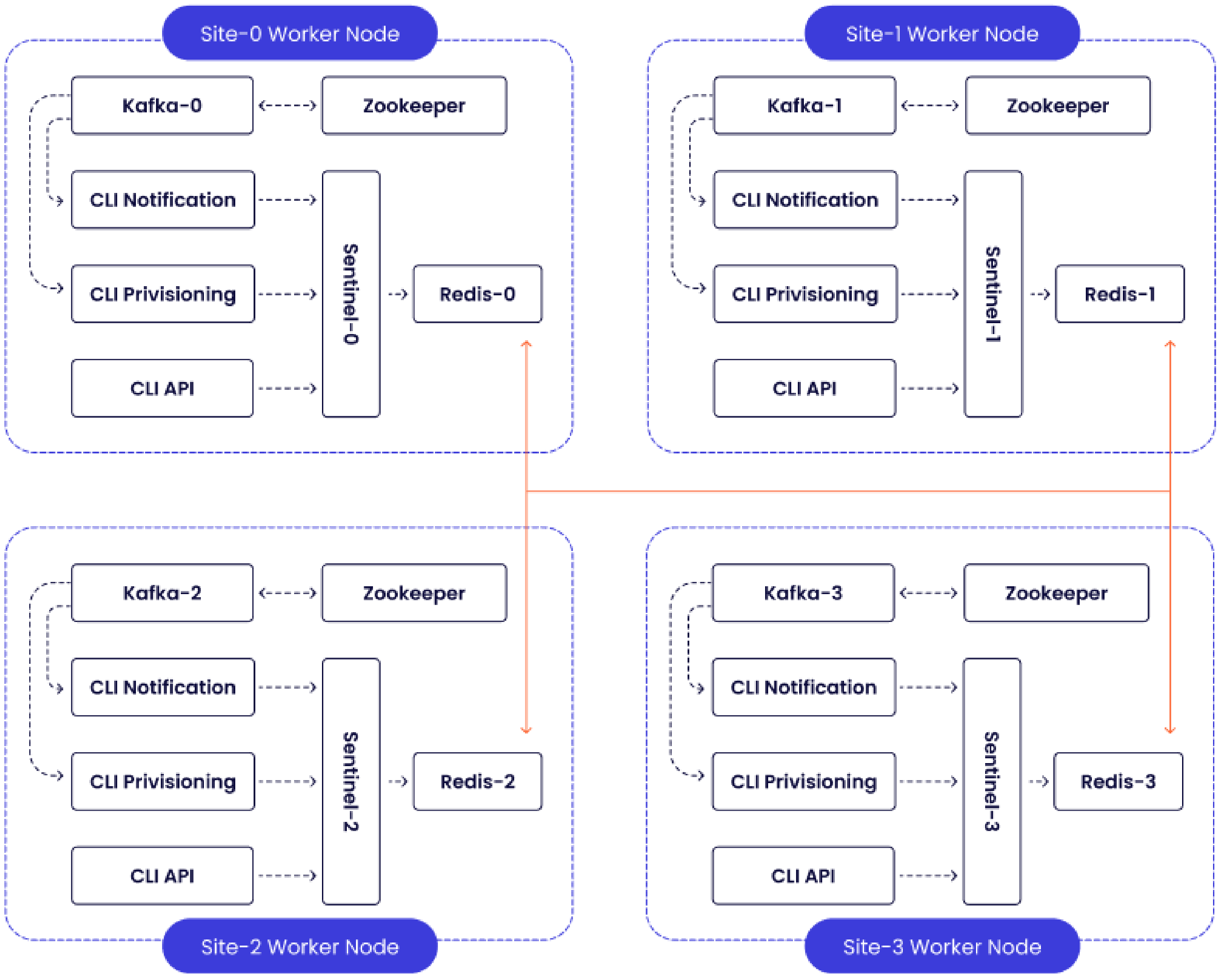
Kubernetes Components

Entire container lifecycle management is provided via Kubernetes orchestration. Each microservices runs as a container inside the smallest unit of Kubernetes, the Pod. Pods usually contain one Container, but can contain multiple containers as needed. Each Pod has its own virtual network and virtual IP addresses, and can communicate with each other through the Kube-Proxy and Kubernetes ClusterIP service. Kubernetes needs a Container Network Interface to create and use a virtual network. Calico CNI is used in CLI to provide better performance and network isolation. Pods run on components known as Nodes. Actually, a node is a virtual machine. Kubernetes has a structure consisting of Master and Worker Nodes. Services are working on Worker Nodes, management and monitoring operations are done using Master Node. The collection of nodes, pods, and all other Kubernetes objects is called a cluster.

CLI Kubernetes Architecture

Kubernetes architecture of CLI includes 3 Master Nodes and 3 Worker Nodes in 4 geo-redundant sites. CLI services work with a total 3 replicas in each Worker Node and different Pods. If one of the pods is down, the copy of the pod starts and runs with the same state because it is deployed with 3 replica sets. If a node fails for any reason, the system continues to work on other standing nodes since each site has 3 worker nodes. In case of a complete geo-redundant site failure, the system continues to work through other geo-redundant sites. In this way, services are able to run with Zero Downtime. If the master node fails then it needs a failover mechanism. When a master node is down, extra master nodes are needed to be able to restore itself with the old state. That's why Multi-Master Clusters are the best practice.

Each Kubernetes Cluster includes CLI API, Notification Handler, Provisioning Service, Kafka, Redis, Zookeeper, and Sentinel pods. For the application to be accessible externally, a Kubernetes service must be created for the deployment. In the CLI Kubernetes Structure, a Kubernetes Load Balancer service is created for each of the CLI API, Notification Handler, Provisioning API and Redis Deployments. The Kubernetes Load Balancer service takes an external IP and allows the user to access the pod via this IP. But the critical point here is that each service communicates with the external network functions over different network interfaces. Thanks to the MetalLB open source tool, each microservice receives different IP blocks from different network interfaces.
Incoming requests are equally distributed to the 4 geo-redundant sites by an external load balancer. Counselee or Counselor provides access to CLI microservices using this load balancer.

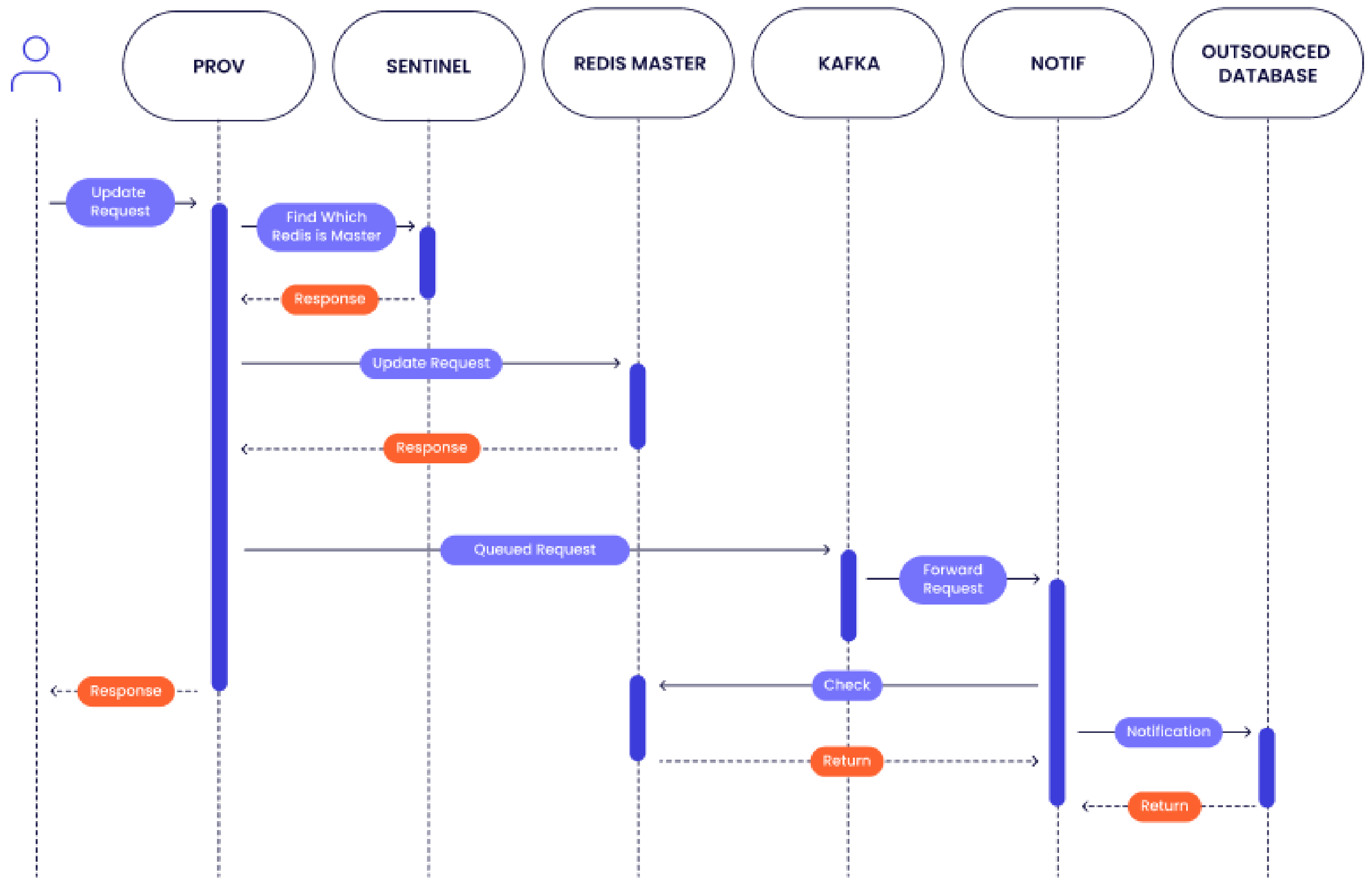


CLI Kubernetes Structure

Redis Database & Apache Kafka

Redis is an open source and In-Memory database. Redis can store many kinds of data structures such as strings, hashes, lists etc. Since Redis is a memory-based database, the amount of memory used increases in direct proportion to the size of the dataset in the database. As the data grows, the memory requirement also increases. Therefore, high performance VMs should be used. Although Redis uses memory to read, write and store data. It periodically writes all data to a file on the disk for data persistence. When the database falls and starts again, it can get the data from this file and return to its old state. Unlike the relational databases, Redis stores its data as key/value pairs.

CLI Redis has 2 different type KEY/VALUE pairs; these are IMSI/MSISDN and MSISDN/SUBSCRIPTION. If a GET KEY request is received by Redis, then it sends VALUE in response. In this way, it can be quickly and easily queried whether an IMSI or MSISDN belongs to a registered subscriber in the database. The most critical part in the CLI is the response time of the Redis. Counselee and Counselor send GET KEY request to CLI API. CLI API queries the request on the Redis Database and responds the VALUE to Counselee or Counselor. All of the processes must take 5 milliseconds max. A request-response pair on CLI API is a transaction. The total number of transactions that occur in 1 second are called TPS. As a result of the load tests performed with the CLI API service, it has been proven that even in high network traffic like 30000 TPS (transaction per second) in a site, the response time varies between 0-1 milliseconds. In the regular conditions expected average network traffic of CLI API is 7500 TPS in a site. Redis is a right choice for performance requirements.



Provisioning Request Flow

Redis Cluster contains Master and Slave Nodes. If Redis Master Node fails with some reasons, one of the Slave Nodes becomes a new Master. However, Redis connection will fail with CLI API, CLI Provisioning and CLI Notification Handler services due to IP address or other information about the new master is not defined in configuration files. Redis Sentinel joins the Redis Cluster to solve this problem. When the master node falls, the Sentinels vote among themselves to assign a new master node. One of the Redis slave nodes becomes the new master and the microservices take this information from the own sentinel. In this way, services will not lose connection with Redis.

In the event of a disaster, if one of the sites fails completely, the synchronization between the Redis Cluster Nodes will be broken. In this case, a manual intervention is required. If a site containing one of the Redis Slave pod fails, the Redis Slave can re-sync itself using the Redis Master once the site is up and running. If 2 sites fail at the same time, synchronization will be broken. In this case, all redis pods are stopped by manually. The Redis Pod that is initially run becomes Master Pod. After that, other Redis pods are run one by one. These pods become slaves and start synchronizing themselves from the master.

Apache Kafka queues the incoming requests so that the same data can be found on each node of Redis Cluster and distributes them to the Provisioning and Notification Handler services.

Zookeeper tracks status of Kafka cluster nodes, Kafka topics, Kafka partitions etc. Zookeeper ensures Kafka Cluster to be highly available.The relationship between Redis and Sentinel is also visible in Zookeeper and Kafka. Zookeeper also plays an active role in determining Kafka Leader.

Conclusion

The CLI, which is developed using open-source tools, achieved the purpose of initiating a reliable call between subscribers registered to different operators. With the load tests, it has been observed that CLI gives response to the requests from the Supervisory Authority between 0 and 1 milliseconds. The response time has 0 and 1 ms average also with different network loads. Even when the traffic load is tested with a load much higher than the normal traffic for test purposes, the response time did not exceed 5 milliseconds on average. It has been proven to respond quickly and accurately to HTTP requests sent by the CLI API. With Kubernetes, the system has become highly available and reliable.

Defne Telekomünikasyon AŞ,
headquartered in Istanbul, Turkey, established in 1996, is a leading global provider of telecom solutions, software products, and services for communications networks. Defne's solutions enable network operators and service providers to monetize every potential connection beyond limits while enhancing the subscriber experience.

Backed up with professional and managed services, Defne offers solutions in call completion, messaging, value added services and roaming business lines of telco. Expertise in IN, IVR, and messaging combined with a wealth of skilled resources, allows Defne to provide reliable and scalable solutions that seamlessly integrate with existing customer infrastructure.

+90 212 909 85 85

sales@defne.com.tr

Maslak Mahallesi, Maslak Meydan Sokak, Spring Giz Plaza,
No:5, Kat:9 Sariyer, ISTANBUL, 34485 Turkey