Big Red button API

1. Requirements

Summary

The purpose of The Big Red button is to shut down immediately the RIC in the case of messes up the network.

Particularly, with the “delete all” request the e2 manager is requested to delete all the RANs and update node states in the RNIB DB.

Once the e2 manager received the “delete all” request,

he should update states of all the nodes and send a “clean all” message to the e2 terminator,

which, his turn, should close all SCTP connections to RANs and send unsolicited disconnection message to the e2 manager per each RAN.

The e2 manager’s concurrently listening RMR service should handle these unsolicited disconnection messages.

Once 5-second timeout expired, the e2 manager has to retrieve all nodes from the RNIB DB, process all the Shutting-Down RANs,

log errors and change the state to Shut-Down and save them in the RNIB DB.

The “delete all” request handling: success flow

The “delete all“ HTTP Request header	
URL	http://<host>:<port>/v1/nodeb/ shutdown
Method	POST
accept	application/json
No Body	

In case if "Delete All" request has been received, the e2 manager has to retrieve all nodes from the RNIB DB,

change the states of all the nodes according to the table below and save updated nodes in the RNIB DB.

The node state change logic in case of shutting down request received	
Current state	**Next state**
Connecting	Shutting down
Connected	
Connected Failure	
Disconnected	Shut down
Shut down	Ignore
Shutting down	

Once the updated nodes successfully stored e2 manager sends to e2 terminator “Clean All” message,

which eventually closes all the SCTP Connections to all RANs:

RIC_SCTP_CLEAR_ALL == 1090

Once 5-second timeout expired (BigRedButtonTimeout should be defined in the configuration.yaml),

the e2 manager has to retrieve all nodes from the RNIB DB, process all the Shutting-Down RANs,

log errors and change the state to Shut-Down and save them in the RNIB DB.

Once “delete all” request successfully handled e2 manager returns an HTTP response with code: 204 No Content.

The “delete all” request handling: unhappy flow

Once e2 manager failed to handle “delete all” request successfully he should log the failure and return specified error HTTP response:

Module/ cause	HTTP response code
RMR service internal error	500
Resource temporary not available, please try later	500
RNIB DB internal error	500
E2 manager internal error	500
E2 manager HTTP server inaccessible	404

The unsolicited disconnection message handling: success flow

During the execution of the delete of the SCTP Connections by the e2 terminator,

the e2 manager will receive unsolicited disconnection message for each connection.

The e2 manager has to handle each unsolicited disconnection message in the following way: retrieve node from the database,

change the state of the node according to the table below and save the updated node in the database.

In case of absence of the node in RNIB DB the e2 manager log it with the error mode.

The node state change logic in case of the unsolicited disconnection message received	
Current state	**Next state**
Connecting	Disconnected
Connected	
Connected Failure	
Disconnected	Ignore with log warning
Shut down	Shut down
Shutting down	

2. Solution

Summary

The current section describes The Big Red button flow and implementation of the “delete all” RESTful API of the e2 Manager.

The document describes single thread logic per each of 2 flows:

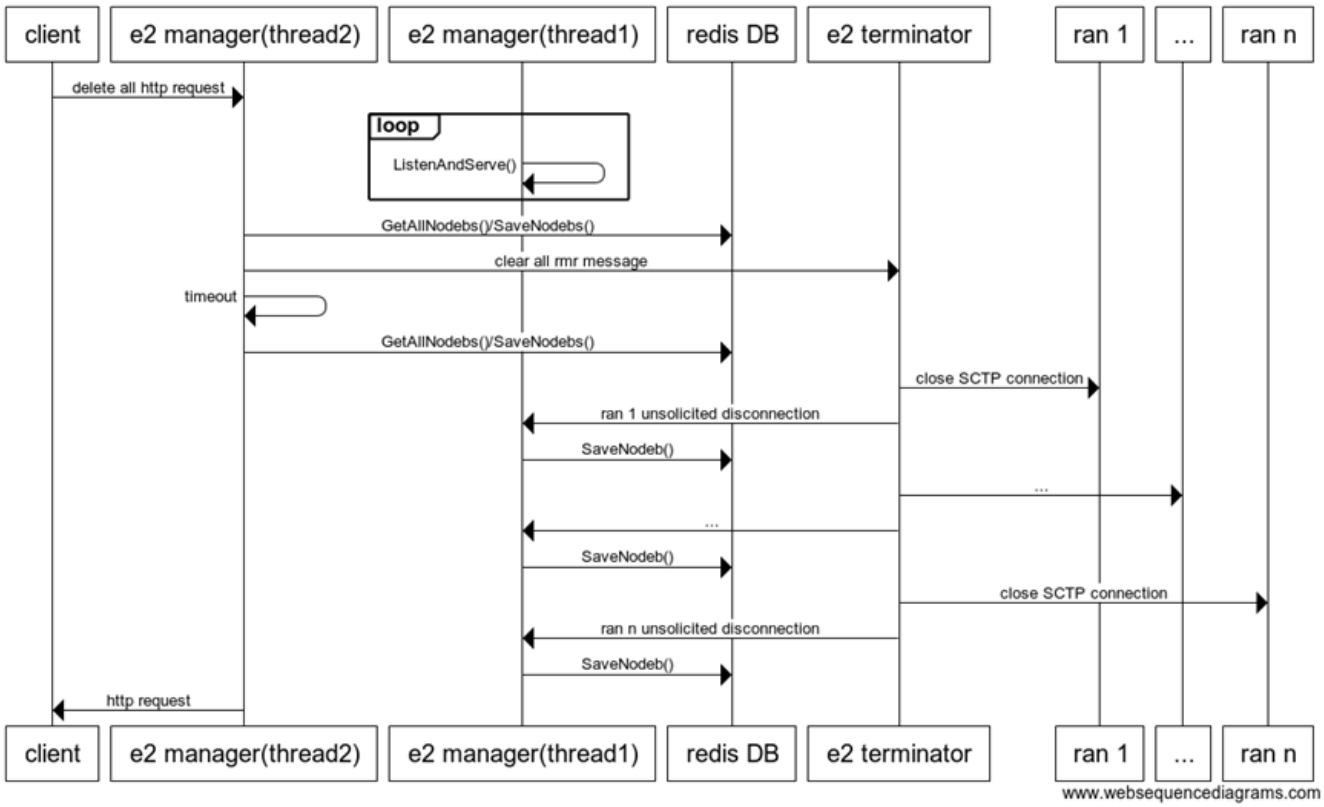
"delete all" request handling and "unsolicited disconnection" message handling,

and doesn’t cover shared database resource access management (such as distributed lock management, etc.)

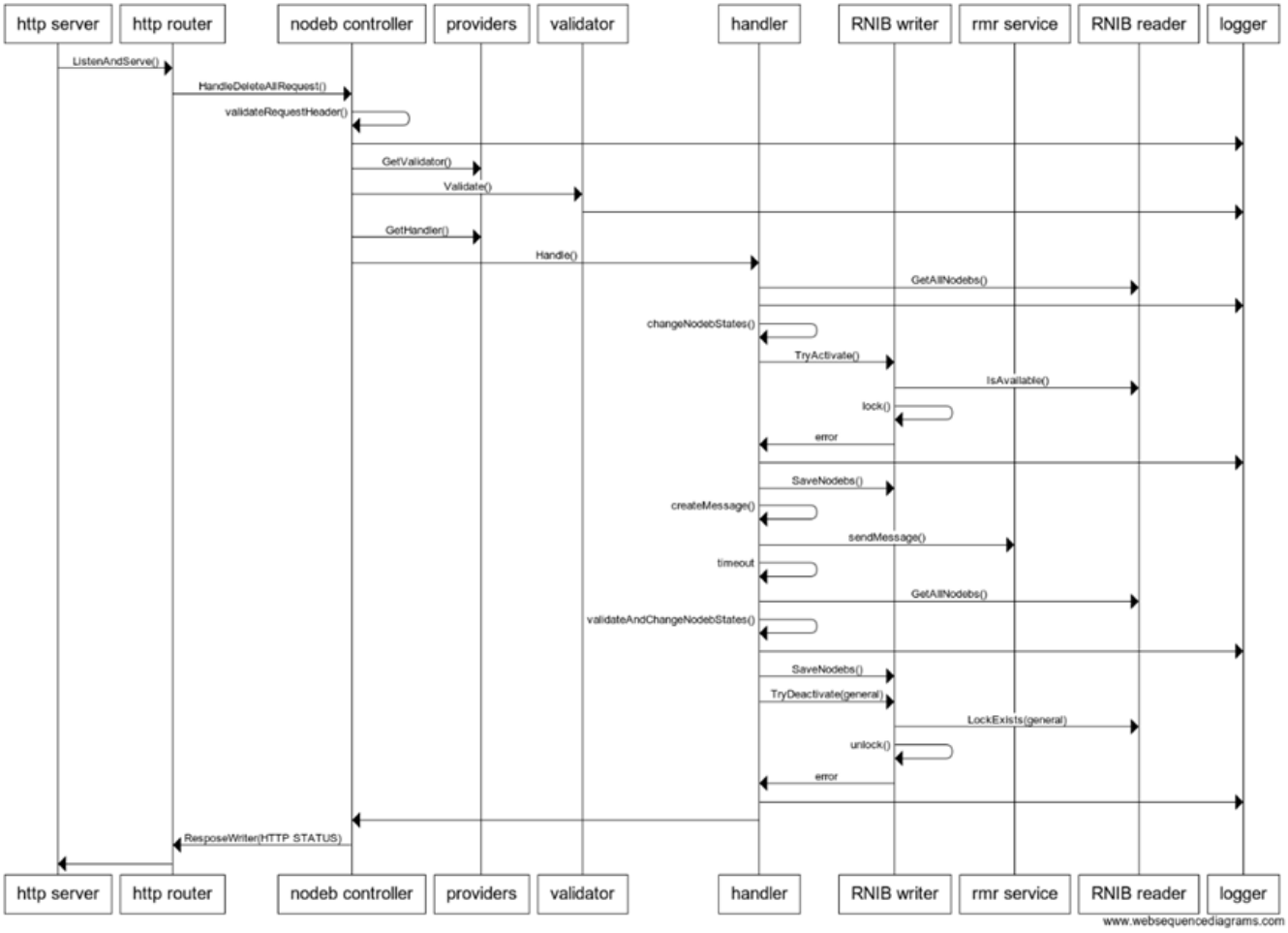
The implementation will extend/change HTTP request handling logic that already exists in E2 Manager.

The suggested changes will be minor (mini-invasive) and as much as possible.

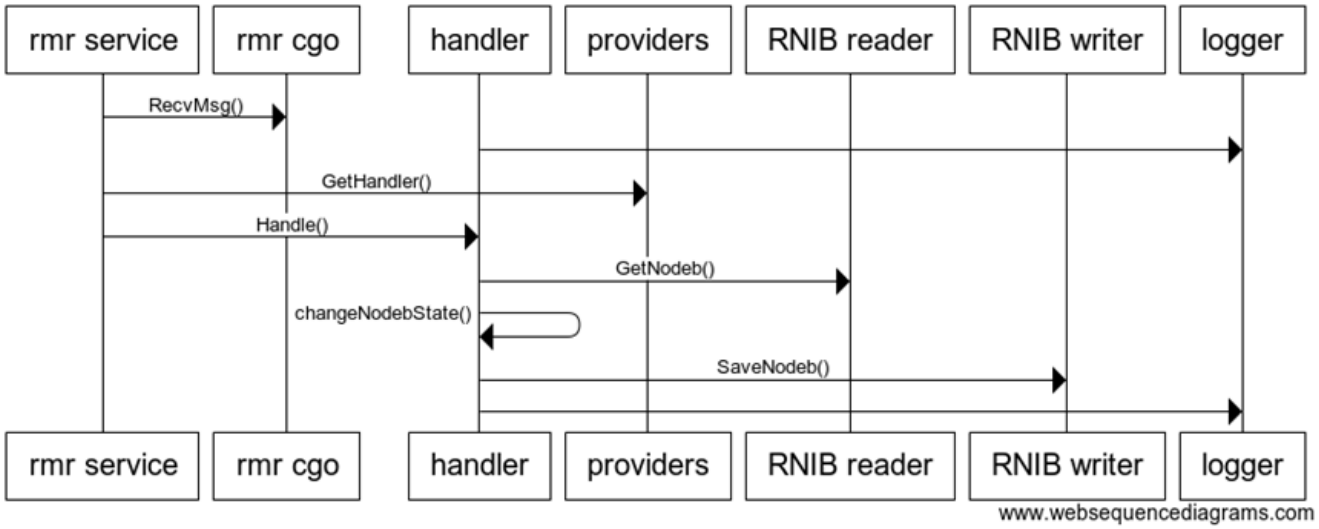
The Big Red Button: high-level flow



The Big Red Button flow: handle "delete all" request flow (e2 manager level)



Unsolicited disconnection message handling: e2 manager level



The implementation: e2 manager changes

Package	new method	description
controllers: nodebController	HandleDeleteAllRequest() validateRequestHeader(http.Request)	Handles received “delete all” request Validates received HTTP request header

RNIB: Reader	GetAllNodebs() IsAvailable() LockExists(general)	Retrieves all the nbIdentity entities from RNIB DB (both ENB and GNB) Checks that the shared resource is not locked Checks that the received general entity exists
RNIB: Writer	SaveNodebs(nodesList) TryActivate() TryDeactivate(general)	Saves all the received node entities in RNIB DB Locks shared resources in case they are not locked yet. Removes received general entity
handlers: DeleteAllRequestHandler	changeNodebStates (nodesList) getAllNodebs() validateAndChngeNodebStates (nodesList)	Changes the states of the nodes according to the logic described above Retrieves all the node entities from RNIB DB (both ENB and GNB) Passes on all Shutting-Down RANs, log errors and change the state to Shut-Down
handlers: SctpErrorNotificationHandler	changeNodebState(node)	Changes the state of the node according to the logic described above
providers: RequestValidatorProvider	GetValidator()	Returns a validator object specifically for an HTTP request

The new Protobuf entity “General” will be implemented:

message General{
uint32 big_red_button_activated = 1;
uint64 timestamp = 2;

}

The new constants (5, 6) will be added to NodebInfo Protobuf:

enum ConnectionStatus{

UNKNOWN_CONNECTION_STATUS = 0;

CONNECTED = 1;

NOT_CONNECTED = 2;

CONNECTED_SETUP_FAILED = 3;

CONNECTING = 4;

SHUTTING_DOWN = 5;

SHUT_DOWN = 6;

}

Open issues

1. Consider using Redis TTL to lock / unlock
2. Update swagger (PUT method instead of DELETE)