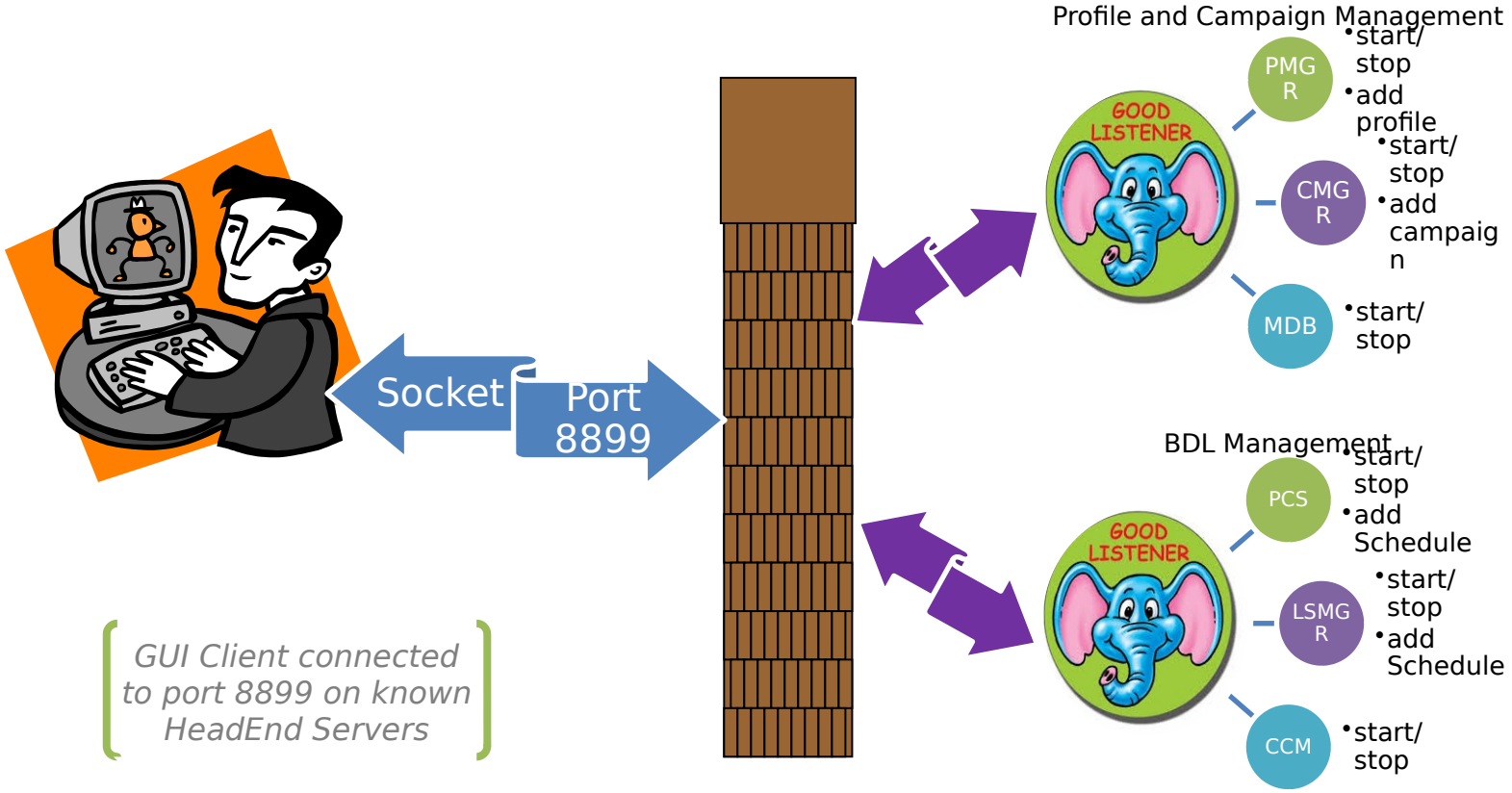
Remote Execution Framework

Kumaran Devaneson

Block Diagram



GUI Client connected to port 8899 on known HeadEnd Servers

Firewall changes have been made to allow connections on port 8899

One listener process per server handling headend operation and one executor process per headend process running on that server.

Listener

- Listener is a double ended application, on one side it creates a socket on port 8899 and listens. On the other side, it connects to one or more Executor process over a local socket.
- The names of Executors to connect to is defined in the configuration file “listener.config”.
- Listener.config format.

```
[EXECUTORS]
CMGR
PMGR
```

- The local (named) sockets are named using the Executor names. For example, socket to CMGR will be “/tmp/CMGR” etc.
- Clients will connect to Listener on port 8899 and send command strings targeted for Executors.
- Listener is responsible to dispatch commands to Executor(s) and return results obtained from them.
- It is a multi-threaded application, which spawns dedicated threads to each client connection.
- A background task to update the status of Executors runs as a continuous thread. If an Executor has restarted or shutdown, its status is automatically updated. This will enable Listener and Executor to start/shutdown independently.
- Listener process will need a restart if new Executors are added because the configuration file is loaded only during start-up.

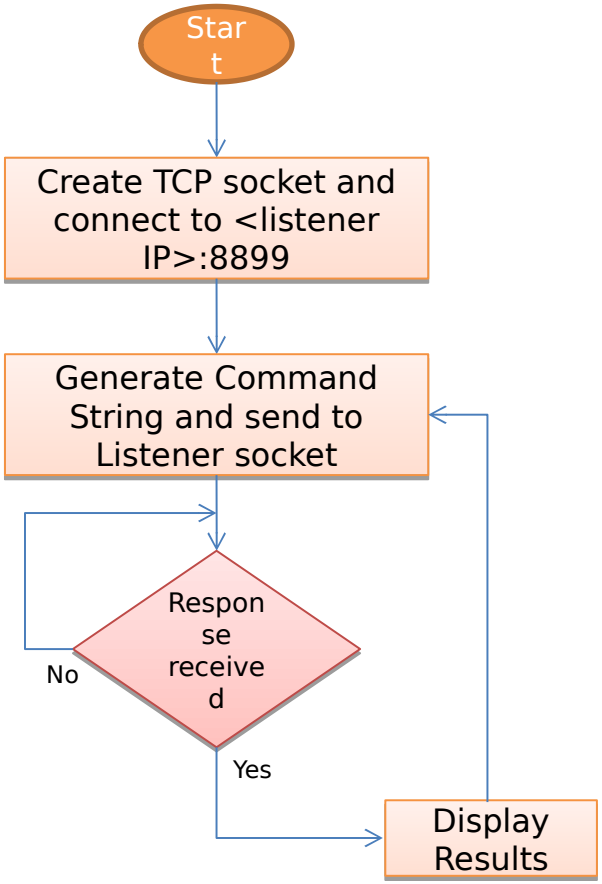
Executor

- Executor is the work horse application of the framework. The commands are executed and results are sent back by this application.
- It is a single application which takes up different avatars of Executor based on the entries in the configuration file “executor.config”.
- Executor.config format.

```
[NAME]
PMGR
[COMMANDS]
start.sh
stop.sh
pam_send.py <SubId> <SubId> (<AttributeNames> <AttributeValues>)
```

- When the Executor is started off the above configuration file, it will assume the role of PMGR Executor and be capable of running those listed commands.
- Executors are bound to user profiles. Typically, Executors will be started after logging in as different users as the HeadEnd application they represent.
- Executor is a multi-threaded application, where each command request is run using a separate thread. This will enable Executor to run multiple command simultaneously.
- The commands are local executable scripts that are to be invoked. The results returned by Executor is the output generated by these scripts.
- “list” is a special command which will return the list from the [COMMANDS] section of the configuration file.
- Clients should not talk to Executors directly.

Simple Client Design



SimpleClientTest.7z

1. This is a simple Client which connects to Listener IP over port 8899 using TCP sockets.
2. Command string is generated using the generic format : “<Executor Identifier> <Command String with Parameters>”
3. Example: PMGR ./pam.py 54321 54322 TSA1 Max
4. The logic as to how the command is generated is left to client implementation as long as the format is as expected.
5. The above command will be routed to Executor running as PMGR and the script pam.py is executed with given parameters.
6. The client then waits till a response is received, and continues to form the command again.
7. The client written using C is attached to this page.

Framework Design

- The framework is capable of the following:
  - Remote execution of commands/scripts which would typically require the user to assume different user profile.
  - Text based interface. The commands dispatched for execution and results obtained are simple null terminated text.
  - The command sent and the received are 1024 bytes in size. It is expected that the client is always writing and reading 1024 bytes from the socket, preferably null padded if lesser sized data is carried across.
  - A client can talk to more than one Listeners on different sockets.
  - There are no such limitations on the number of connected clients to each Listener and number of commands that can be run simultaneously.
  - The list of commands in executor.config is currently used for cosmetic purposes. Listener is not verifying the received commands based on this list.
  - A new script can be added on the Executor side and invoked from the client with out requiring a restart of the framework.

- Some of the limitations:
  - 1024 size can be limited in some situations. This value can be increased or decreased at will. But any increase or decrease will require a re-compilation of Framework and client code.
  - The framework itself doesn’t support any authentication. The clients would have handle any such requirements.
  - The interface is text based and not suitable to transfer binary data. The interface can still be used to transmit binary data but may be result in unexpected behaviour.
  - The number of clients connected to a Listener and concurrent commands are only limited by the operating system or the hardware on the server. It should also be noted that the Listener and Executor are run on the same server as HeadEnd modules, so, discretion is expected.