Watchdog v2.x

Manuel du développeur

Table des matières

1 Introduction

Watchdog est un outil communiquant, écrit en langage C. Il s'appuie sur les outils standards de la chaine de compilation GNU GCC, et s'articule via 4 outils spécifiques :

- Le client, qui représente l'interface déportée d'utilisation du logiciel.
- Le serveur, qui représente le nœud central et l'intelligence du logiciel.
- Une librairie commune au client et au serveur
- Un outil d'administration local au serveur.

Les répertoires sont les suivants hébergeant les codes sources sont les suivants:

- Client
  - Code source du composant « Client », et ses librairies graphique et de configuration
- Watchdogd
  - Code source du composant « Serveur » de Watchdog, et ses sous-composants (accès à la base de données, module de communication, d'archivage, ...)
- Commun
  - Code source de la librairie partagée entre le « Client » et le « Serveur »
- Include
  - Valeurs, références et déclarations des fonctions partagées entre le « Client » et le « Serveur »
- WatchdogdAdmin
  - Code source du composant d'administration locale du composant « Serveur »

L'objectif de ce document est de présenter les architectures techniques et logicielles correspondantes, après un focus sur les spécifications et les contraintes associées.

1.1 Spécifications

- Langage de programmation automatisme propre « D.L.S »
- Interfaçage RS485 et MODBUS sur IP
- Interfaçage avec les Onduleurs
- Émission de messages textuels, audio, et SMS
- Supervision vidéo passive
- Architecture logicielle modulaire
- Reconnaissance vocale (Soon)
- Administration locale et diagnostique système bas niveau
- Gestion fine des Utilisateurs et des Habilitations
- Supervision graphique des sites surveillés
- Historique des évènements
- Reprise sur erreur/reboot
- Authentification des clients par certificats
- Flux réseaux clients chiffrés

1.2 Contraintes

Les contraintes imposées sont les suivantes :

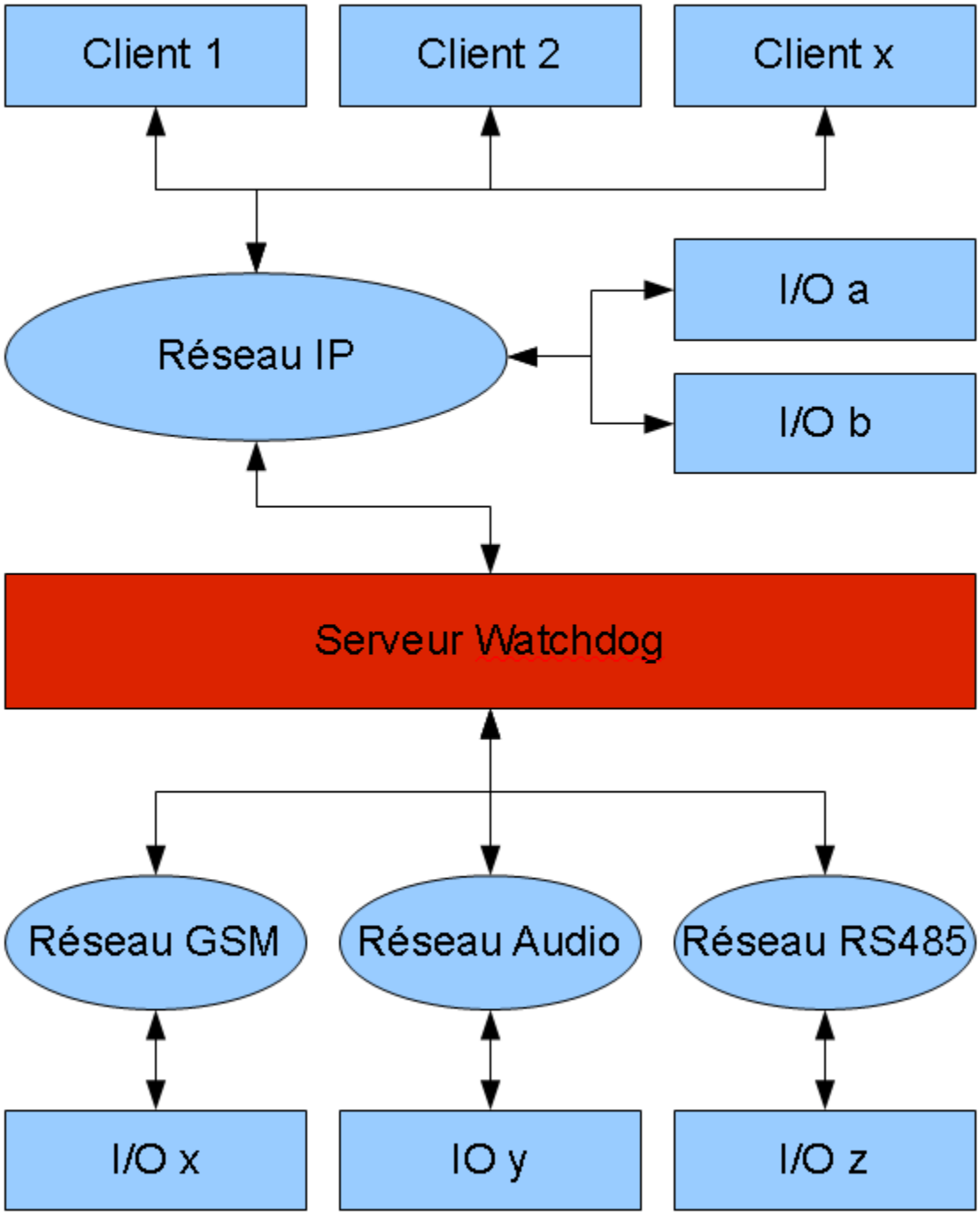
- Des communications inter-outils basées sur les réseaux IP ou sur des tubes nommés
- Utilisation des outils et librairies standards livrés avec les distributions Linux « Mandriva »
- Diffusion sous licence « GNU GPL v2 or later » pour le code source, sous licence CC-BY-NC-SA pour les manuels et documentations associés.

2 Architecture technique

« Watchdog » s'appuie des technologies éprouvées dans les domaines industriels, et notamment du domaine des réseaux et de l'automatisme :

- Les liaisons entre les différents composants « Clients » et le composant « Serveur » sont assurées par un réseau IP (filaire ou wifi)
- Les liaisons entre le composant « Serveur » et les modules de contrôle/commande (« I/O ») sont assurées soit par ces mêmes réseaux IP, soit par des bus dédiés type RS232/485
- Les modules de communication, intégrés au composant « Serveur » du logiciel, utilisent des liaisons propres à chacun des modules en question : le GSM pour le module de gestion des SMS, des liaisons filaires audio pour celui gérant la diffusion de messages vocaux, etc …

Les composants « Clients » sont installés sur les stations (PC de bureau) des utilisateurs du logiciel.



Les composants « Outil d'administration locale » et « Serveur » sont installés sur une machine en salle blanche tournant 24h/24, 7j/7

3 Architecture logicielle

Les différents composants du logiciel s'appuient sur plusieurs briques logicielles nécessaires à leurs bons fonctionnements :

- Socle Linux « Mandriva »
- Serveur de base de données « MySQL »
- Journalisation via « Syslog »
- Interface graphique « GTK »
- Librairie de gestion Multi-threadé « pthread »
- Interface GSM « gnokii »
- Chiffrement par certificat « OpenSSL »
- Synthèse vocale basée sur « Espeak » et « MbrOla »
- Envoi de SMS par SMSBOX, basé sur la librairie « libcURL »

L'objectif de ce chapitre est de préciser, pour chacun des composants du logiciel, leurs fonctionnements internes.

3.1 La librairie commune à « Watchdog-client » et « Watchdogd »

La gestion de la journalisation des messages d'erreur et d'information, ainsi que la gestion des communications réseaux étant communes aux composants « Clients » et « Serveur » du logiciel, elles ont toutes deux été positionnées dans une librairie, liée dynamiquement aux binaires des deux composants.

- Les sources de cette librairie sont dans le répertoires « Commun/ »
- Les headers sont dans le répertoire « Include/ »

3.1.1 Journalisation et gestion d'erreur

La gestion d'erreur est prise en charge par le fichier « erreur.c », qui met a disposition des fonctions s'occupant principalement de diriger les messages vers « Syslog », et ajoutant des informations liées au processus en cours de fonctionnement.

Au démarrage du Client, ou du Serveur, celui-ci :

- Initialise les paramètres via la fonction « Info_init »
- Passe ses messages via les fonctions « Info_n » et « Info_c » selon la nature du paramètre (caractère ou numérique)
- Termine la communication via la fonction « Info_stop »

3.1.2 Gestion des communications réseau

La gestion des communications réseaux s'appuie sur des fonctions positionnées dans les fichiers « Réseau.c », « Reseau_*.c » et sur des entêtes « Reseau.h » et « Reseau.*.h ».

Une communication réseau est établie via la fonction xxx, et se termine via la fonction xxx. Le processus demandeur doit alors appeler régulièrement la fonction xxx pour s'assurer qu'un packet réseau est à sa disposition ou non.

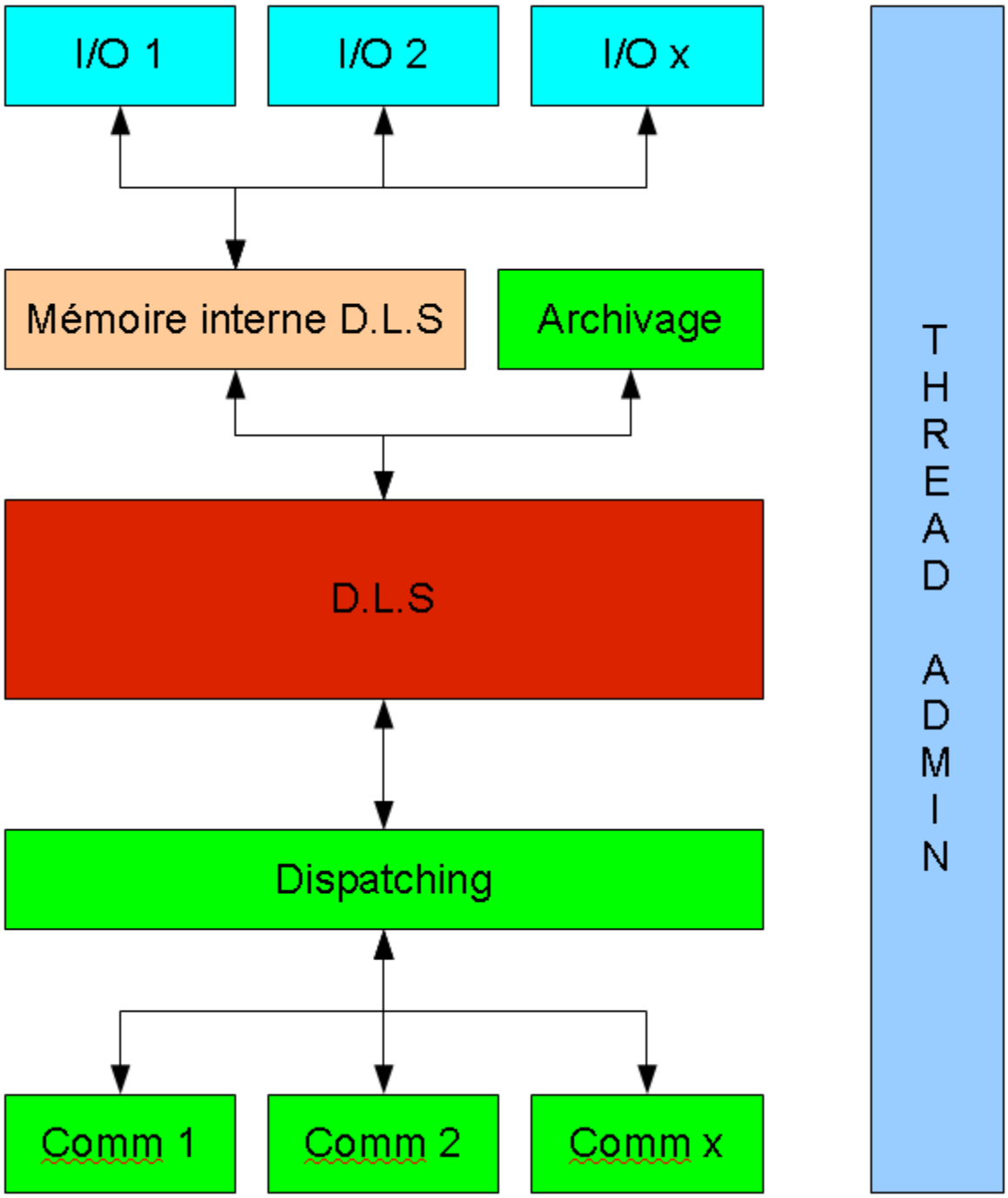
Pour envoyer un paquet, le demandeur doit utiliser la fonction Envoyer_reseau.

3.2 Architecture logicielle du Serveur « Watchdogd »

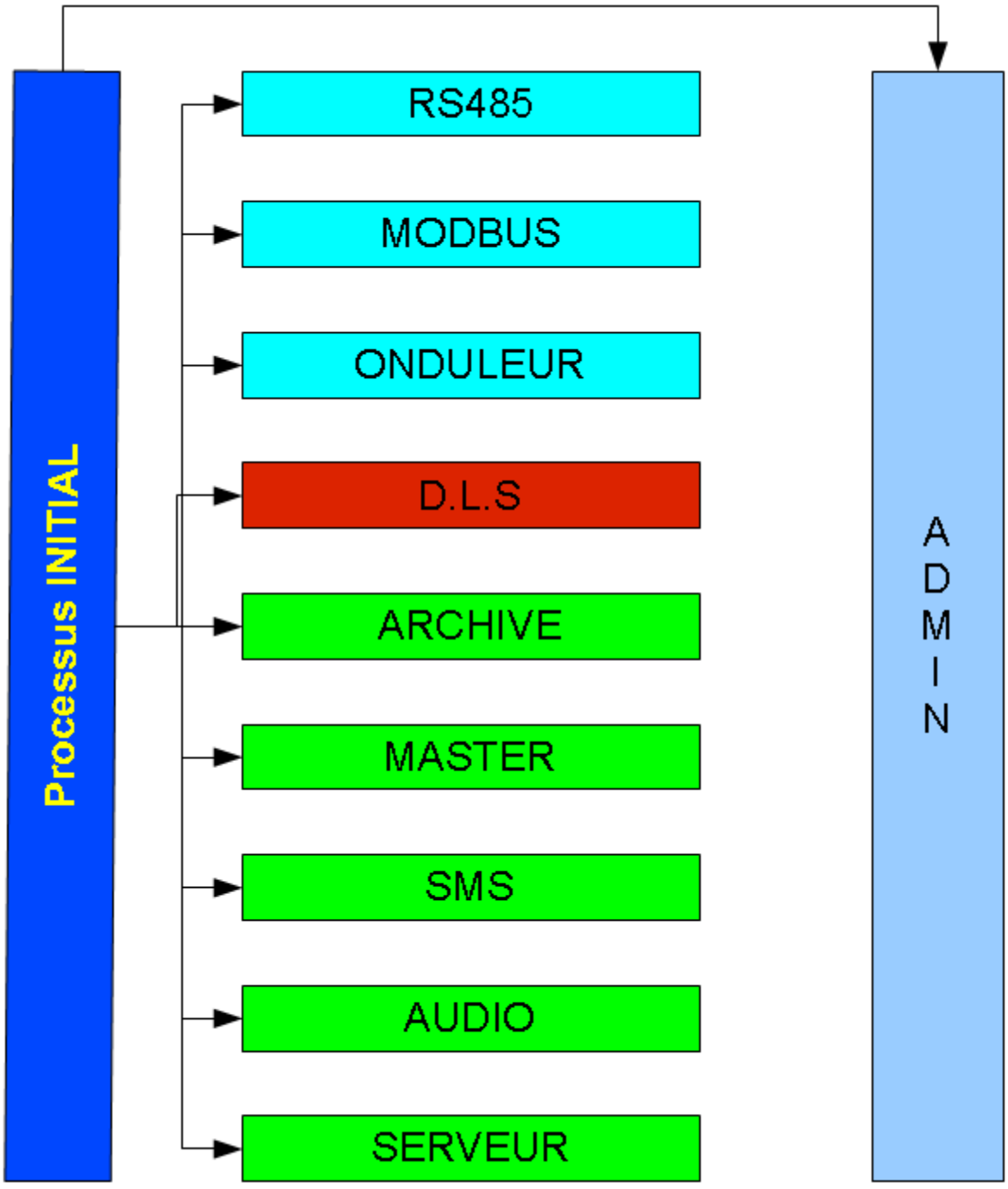
Plusieurs processus (thread) tournent en parallèle, chacun pouvant être catégoriser selon son activité :

- **Thread Initial**
  - C'est le processus initial, créé au lancement de l'application Watchdogd
- **Thread I/O**
  - Ce sont les processus en charge de la communication entre les modules physiques de contrôle/commande et la mémoire interne de travail
- **Thread D.L.S**
  - Ce processus a en charge le traitement des informations présentes dans la mémoire interne par des sous-processus représentant l'intelligence du système
- **Thread Dispatching**
  - Ce processus a en charge la mise en forme des informations issu du D.L.S avant exportation vers les processus de Communication (Thread Comm)
- **Thread Comm**
  - Ce sont les processus chargés d'exporter une information mise en forme par le processus de dispatching vers des modules de communication spécifiques
- **Thread Archivage**
  - C'est le processus en charge de l'archivage des bits internes D.L.S
- **Thread Admin**
  - C'est le processus en charge de l'administration locale de l'outil. Il assure la communication entre le technicien responsable de la plateforme et le logiciel

Un schéma de synthèse est présenté ci dessous.



Le schéma suivant présente l'exhaustivité des threads du logiciel.



Les threads « RS485 », « MODBUS », « ONDULEUR » sont des threads I/O.

Le thread « D.L.S » est le thread associé à l'activité DLS.

Le thread « MASTER » est le thread de dispatching.

Le thread « ARCHIVE » est en charge du stockage des informations dans la base de données au fil de l'eau.

Les threads « SMS », « AUDIO », et « Serveur », sont des threads de Communication.

Le thread « Admin » est aussi représenté, de manière transversale.

Les sources du serveur « Watchdogd » sont présentes dans les répertoires suivants:

Thread	Répertoires
INITIAL et MASTER	Watchdogd/
D.L.S	Watchdogd/Dls Watchdogd/Traduction_DLS
Type I/O	Watchdogd/RS485 Watchdogd/Modbus Watchdogd/Onduleur
Type Communication	Watchdogd/Audio Watchdog/Sms Watchdog/Archive
Communication avec les clients	Watchdogd/Serveu
Administration locale	Watchdogd/Admin
Références les valeurs et définitions de fonctions partagées à l'ensemble des sources de l'outil	Watchdogd/Include
Héberge les fonctions de gestion des paramètres de configurations ainsi que la grammaire associée	Watchdogd/Config
Les autres répertoires présentent les codes sources des fonctions ayant une activité de mise à jour de la base de données globale « Watchdog »	Watchdogd/Message, Watchdogd/Mnemonique, Watchdogd/Synoptique, ...

3.2.1 Le thread « RS485 »

Ce thread a la charge les communications des différents modules « RS485 » Pour cela, à l'initialisation, il charge en mémoire une liste de module issue du serveur de base de données. Cette liste est la liste de travail du thread, et est composé de structures

permettant d'avoir une visibilité sur l'état de chacun des modules.

Le corps du thread est une boucle sans fin, parcourant chacun des items de la liste, et reagissant à l'état actuel du module :

- Si l'etat actuel n'est pas l'etat souhaité, le système connecte (ou deconnecte) le module
- Si le système n'est pas en attente de réponse du module, une trame de commande lui est envoyé, en précisant quelles sont les sorties à positionner
- Si le système est en attente de réponse du module, une zone mémoire est remplie avec les octets recus par le système. Quand cette zone vérifie les contrainte d'une trame RS485, alors elle est processée, et les bits internes de D.L.S sont mis à jour.

3.2.2 Le thread « MODBUS »

Le thread « MODBUS » a exactement le meme squelette que le thread précédent, si ce n'est le type de communication, qui devient MODBUS sur TCP, au lieu d'une communication série RS232.

Le principe est le meme

- Chargement d'une liste de configuration des modules MODBUS issue de la Base de Données
- Boucle sans fin gérant un par un les différents modules
  - Demande l'etat des entrees et mets à jour les bits internes D.L.S sur réception d'une trame correctement formatée de la part du module.
  - Positionne les sorties en fonction des bits internes D.L.S
  - Gère l'insertion et le retrait des modules MODBUS.

3.2.3 Le thread « ONDULEUR »

Le thread « ONDULEUR » a exactement le même squelette que le thread précédent. Les communication entre le système et les onduleurs se basent sur TCP/IP, et plus particulièrement sur l'application NUT (Network UPS Tool).

Le principe est le même

- Chargement d'une liste de configuration des modules ONDULEUR issue de la Base de Données

- Boucle sans fin gérant un par un les différents modules
  - Demande l'état des entrées et mets à jour les bits internes D.L.S sur réception d'une trame correctement formatée de la part du module.
  - Gère l'insertion et le retrait des modules ONDULEUR.

La différence importante réside dans le fait que les onduleurs sont en Entrée Seulement. Il n'y a pas de possibilité de commande (Sortie) de ce genre d'équipement.

3.2.4 Le thread « D.L.S «

3.2.5 Le thread « MASTER »

3.2.6 Le thread « ARCHIVE »

Le thread « ARCHIVE » se base sur une liste chainée, remplie par D.L.S, présentant chaque bit a archiver dans la base de données. Ce thread existe principalement pour décharger D.L.S, et faire tampon entre le thread D.L.S qui doit avoir une certaine réactivité, et le serveur de base de données qui lui est plus lent en terme d'accès.

Il est composé d'une boucle sans fin, attendant des enregistrements dans la liste de travail. Dès qu'un enregistrement est trouvé, il formate une requête et l'envoi à la base de données. Il libère enfin l'enregistrement

3.2.7 Le thread « SMS »

Le thread « SMS » se base sur une liste chainée, remplie par D.L.S, présentant chaque message a envoyer sur le mobile GSM.

Il est composé d'une boucle sans fin, attendant des enregistrements dans la liste de travail. Dès qu'un enregistrement est trouvé, il formate une requête et l'envoi au GSM, ou a SMSBOX via l'utilisation de la bibliothèque libre « Gnokii », ou la librairie « libcURL ». Il libère enfin l'enregistrement

Périodiquement, ce thread fait une demande de lecture sur le répertoire SMS du mobile GSM et, si un SMS correctement formaté est reçu, il positionne le bit interne monostable de D.L.S dont le numéro correspond au SMS reçu.

3.2.8 Le thread « AUDIO »

3.2.9 Le thread « Serveur »

3.2.10 Le thread « Admin »

3.3 L'outil d'administration locale « WatchdogdAdmin »

L'outil d'administration locale est une petite application limitée à la connexion sur un tube nommé de la machine hébergeant le composant « Serveur ».

Les sources sont disponibles dans le répertoire « WatchdogdAdmin ».

Chaque commande tapée par le technicien sur le prompt sera envoyé dans ce tube, lié au processus « Admin » du Serveur et ce dernier renverra la réponse au demandeur.

Cette réponse est alors affichée sur l'interface textuelle de l'outil d'administration local.

L'outil consiste donc à :

- Ouvrir en lecture/écriture des tubes nommés
- Attendre une commande de la part du technicien
- Envoyer la commande dans le tube nommé
- Afficher la réponse sur la console
- Revenir en attente d'une nouvelle commande.

4 Protocole réseau

Les connexions au serveur « Watchdog » sont soumises au chiffrement par certificat. Avant tout échange de données applicatif, la couche d'abstraction « OpenSSL » à donc en charge d'établir le tunnel de chiffrement en accord avec les paramètres de sécurité et de l'autorité de certification.

*

Dès que le tunnel est monté, les communications applicatives peuvent commencer.

En période d'initialisation, si l'un ou l'autre des composants ne suit pas le protocole, la session est terminée sur le champs.

En période stabilisée, si l'un ou l'autre des composants ne suit pas le protocole, le paquet associé à l'erreur est éliminé

La connexion s'articule autour d'un « mode » : le serveur dispose, pour chacun de ses clients, d'une variable permettant d'estimer la progression de la connexion réseau. Cela permet d'adapter la réponse en fonction de cette variable. Typiquement, le serveur n'enverra pas de données liées à l'interface graphique si le client n'a pas encore envoyé ses login et mot de passe.

La procédure de connexion d'un client au serveur est la suivante:

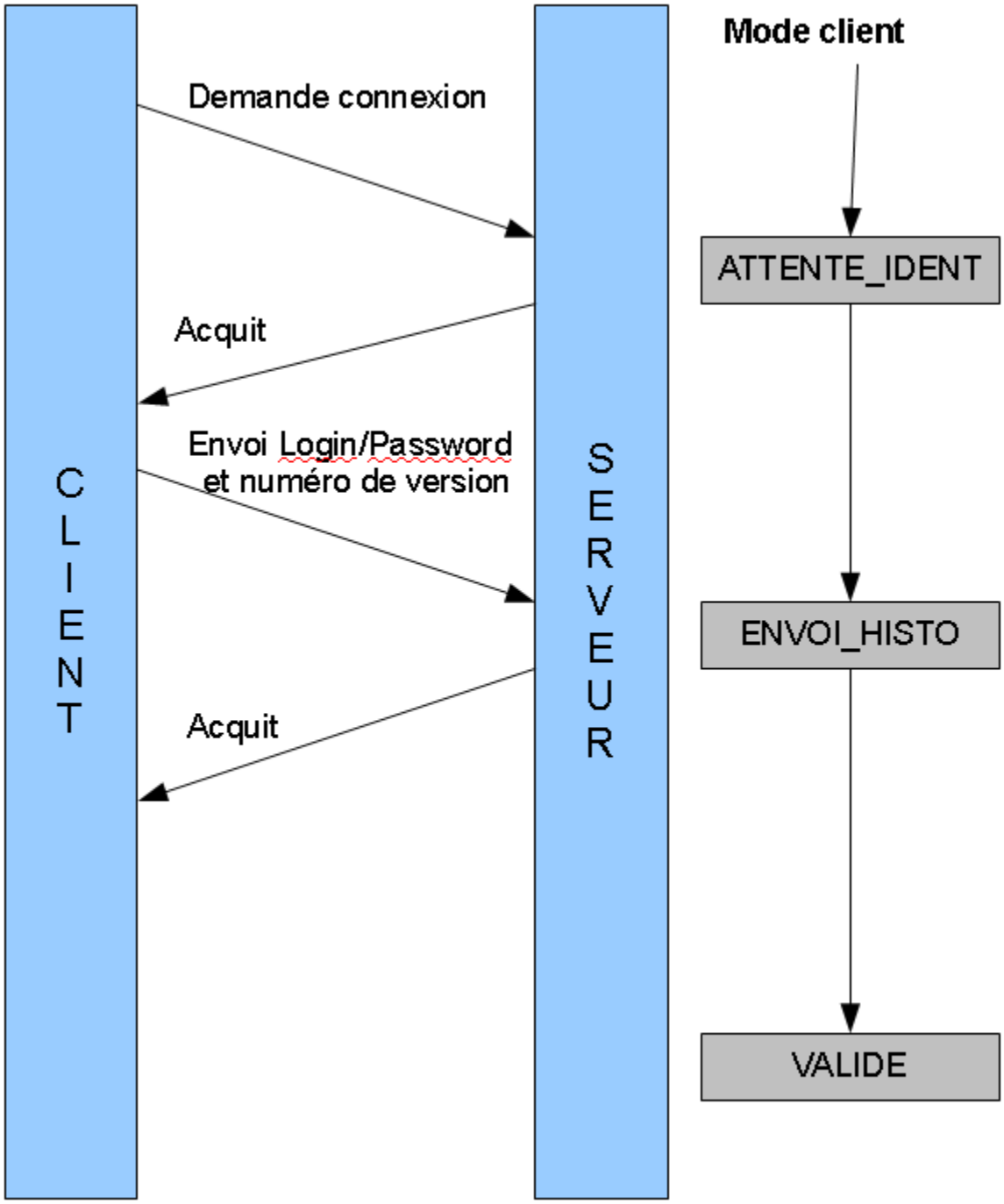
- Le client envoie une demande de connexion au serveur
- Le serveur demande un login et mot de passe au client
- Le client envoie ces informations, ainsi que la version des données locales.
- Si l'authentification échoue, le serveur coupe la connexion.
- Sinon, et si le client doit changer son mot de passe, le serveur le lui demande.
- Le serveur envoie ensuite les données de référence si la version des données locales clientes n'est pas identique
- Le serveur envoie ensuite l'historique des messages en cours de validité
- Le client passe alors, du point de vue du serveur, en mode « VALIDE »

Toutes les communications du logiciel sont catégorisées selon leurs périmètres dans des TAGS, référencés dans les fichiers headers « Reseau.h ». Chacun des paquets échangés entre le « Client » et le « Serveur » sont donc taggués.

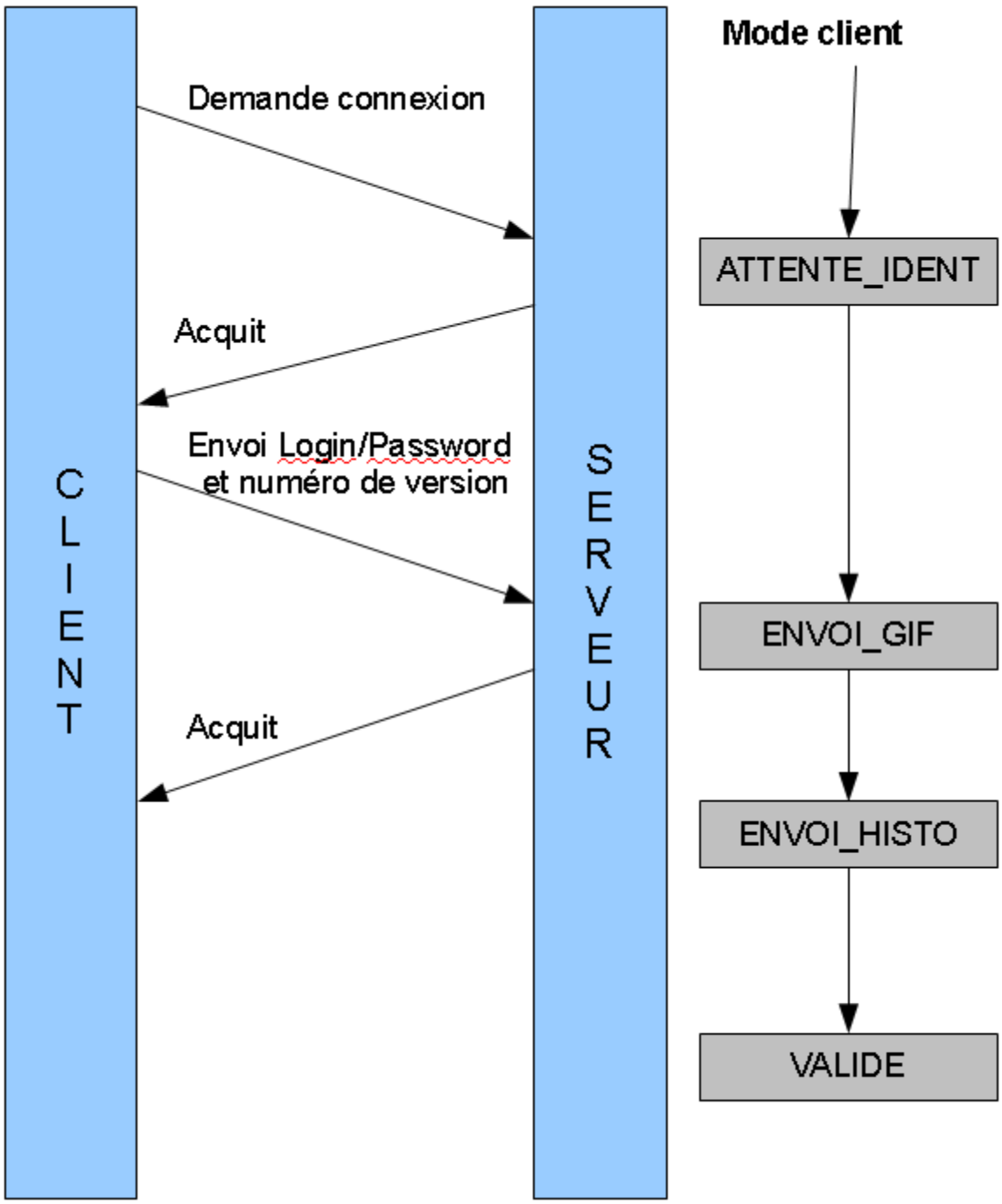
A l'intérieur d'un TAG particulier, un sous tag SSTAG permet d'affiner la demande pour une meilleure compréhension et donc un meilleur traitement. Chacun des SSTAG est référencé dans les fichiers spécifiques « Reseau_*.h », Il existe un fichier de ce type par périmètre de communication (et donc, par TAG).

Chacun des composants communiquant doit donc en premier lieu détecter quel est le TAG associé au paquet réseau reçu, et ensuite seulement aiguiller le paquet selon le SSTAG associé.

Le protocole de connexion le plus simple est le suivant :


CLIENT
SERVEUR
Demande connexion
Acquit
Envoi Login/Password
et numéro de version
Acquit
Mode client
ATTENTE_IDENT
ENVOI_HISTO
VALIDE

Le protocole de connexion avec mise à jour des données locales au client est le suivant :



5 Liens/Contacts

Les différentes personnes ayant contribué au projet sont :

- Bruno LEFEVRE <bruno.lefevre1953@gmail.com>
  - Architecture Technique, Module de controle/commande, Logistique, Programmation D.L.S, Beta-testeur Watchdog.
- Sébastien LEFEVRE <lefevre.seb@gmail.com>
  - Développeur, Programmation D.L.S, Bêta-testeur Watchdog

Les sources du logiciel peuvent être téléchargées sur les sites internets suivants :

- Format web : http://seblef.dyndns.org/
- Format SVN : svn://seblef.dyndns.org/branches-2.x/