POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

REALIZZAZIONE DI UN SISTEMA DI RICEZIONE ALLARMI E TELEGESTIONE UNIFICATO PER CENTRALI ANTI-INTRUSIONE

Relatore: Prof. William Fornaciari
Correlatore:

Tesi di Laurea di:
Matteo Gianello, matricola 771166

Anno Accademico 2014-2015

A Ilaria

Sommario

Ringraziamenti

Ringrazio .....

Indice

Capitolo 1

Introduzione

La diffusione di connessioni a banda larga, il progressivo abbandono di reti telefoniche convenzionali e il passaggio su linee telefoniche VoIP hanno costretto le vigilanze private a trovare nuovi meccanismi di comunicazione verso gli apparati remoti di sicurezza da loro gestiti. Inoltre la richiesta di tempi di intervento più brevi e contatto con il cliente solo quando è indispensabile richiedono strumenti di controllo e verifica immediati e di facile utilizzo.

1.1 Inquadramento generale

Questa tesi è stata sviluppata in collaborazione con *LIS s.r.l.*, vigilanza privata che si distingue per i tempi di intervento ridotti e la possibilità di gestione degli impianti da remoto. Queste caratteristiche distinguono *LIS* già dai primi anni di attività quando ancora la ricezione degli allarmi e la tele-gestione avveniva tramite linee telefoniche tradizionali.
Negli ultimi anni tuttavia ci siamo trovati in difficoltà in quanto molte delle linee telefoniche tradizionali stanno scomparendo sostituite da fibre ottiche e linee VoIP questo impedisce la normale gestione. Si è deciso perciò di effettuare un aggiornamento del sistema in modo da permettere a *LIS* di gestire gli impianti tramite le connessioni a banda larga o tramite linee telefoniche mobili. Oltre alla gestione degli impianti un altro punto sul quale ci focalizzeremo è quello della ricezione degli allarmi in modo tale da permettere una ricezione quasi istantanea della segnalazione di allarme e quindi una gestione immediata dell'eventuale situazione di emergenza.

1.2 Breve descrizione del lavoro

Questa tesi si sviluppa nell'ambito della sicurezza privata. Prima di addentrarci nello specifico dobbiamo capire come lavora una vigilanza privata.

Possiamo distinguere due operazioni principali che una vigilanza privata svolge, la prima è il lavoro di ricezione e verifica delle segnalazioni d'allarme provenienti dalle varie centrali di sicurezza e gli operatori, tramite l'ausilio di immagini provenienti da eventuali sistemi di videosorveglianza, valutano e gestiscono i vari eventi.
La seconda funzione è quella di gestire gli impianti come ad esempio l'inserimento delle centrali antintrusione o l'esclusione di sensori guasti.
In LIS era già presente un sistema di che permetteva all'operatore di gestire

1.3 Struttura della tesi

La terza parte contiene la descrizione della struttura della tesi ed è organizzata nel modo seguente. "La tesi è strutturata nel modo seguente.

Nella sezione due si mostra ...

Nella sez. tre si illustra ...

Nella sez. quattro si descrive ...

Nelle conclusioni si riassumono gli scopi, le valutazioni di questi e le prospettive future ...

Nell'appendice A si riporta ... (Dopo ogni sezione o appendice ci vuole un punto)."

I titoli delle sezioni da 2 a M-1 sono indicativi, ma bisogna cercare di mantenere un significato equipollente nel caso si vogliano cambiare. Queste sezioni possono contenere eventuali sottosezioni.

Capitolo 2

La sicurezza privata

> *"Giuro di osservare lealmente le leggi e le altre disposizioni vigenti nel territorio della Repubblica e di adempiere le funzioni affidatemi con coscienza e diligenza, nel rispetto dei diritti dei cittadini."*
>
> Giuramento di una guardia particolare giurata

La sicurezza (dal latino sine cura: senza preoccupazione) può essere definita come la conoscenza che l'evoluzione di un sistema non produrrà stati indesiderati. In termini più semplici è: sapere che quello che faremo non provocherà dei danni.[5]

2.1 Le vigilanze private

La vigilanza privata è l'attività, posta in essere da persone o da enti di coloro che operano nel campo della sicurezza privata, a tutela di persone, beni e/o enti pubblici o privati [6].
Le vigilanze private sono aziende che si occupano della protezione di persone e di beni mobili ed immobili, esse derivano direttamente dalle milizie cittadine del medioevo che, in tempo di pace svolgevano il compito di controllare e garantire la sicurezza dei cittadini durante la notte, nelle fiere e nei mercati. Oggi le vigilanze private si occupano di diversi aspetti della sicurezza tramite l'utilizzo di tecnologie all'avanguardia. Tra queste attività troviamo:

Piantonamento: questo tipo di attività consiste nel presidio fisso da parte di una o più guardie particolari giurate (GPG) dotate di protezione anti proiettile e solitamente armate, esse sono collegate in modo costante con una centrale operativa. Solitamente tale attività viene svolta presso istituti di credito e enti pubblici. Possiamo distinguere tra piantonamenti diurni, piantonamenti notturni o piantonamenti per brevi periodi. Tale attività viene svolta in quei luoghi nei quali esiste un pericolo costante.

Servizio ispettivo: questa attività consiste nell'ispezione saltuaria di alcune zone come piccole imprese, locali e aree circoscritte. Solto principalmente durante le ore notturne consiste in una visita della zona e nell'esame degli ingressi, degli infissi e del perimetro. Se la GPG durante l'ispezione nota delle anomalie provvede a contattare la centrale operativa che effettuerà gli opportuni controlli ed ad avvisare eventualmente le forze dell'ordine.

Trasporto valori: in questo caso si tratta di un servizio di scorta effettuato da personale armato e dotato di protezioni antiproiettile ed effettuato tramite l'ausilio di mezzi blindati.

Sala conta: questa attività è destinata soprattutto agli istituti di credito e ai centri commerciali. Il denaro viene prelevato dalla sede del cliente e prima di essere custodito nel *caveau* dell'istituto di vigilanza vine ricontato trattato e confezionato secondo precise istruzioni.

Localizzazione satellitare: tramite il sistema GPS è possibile localizzare a distanza un mezzo, inoltre è possibile effettuare alcune operazioni per gestire il mezzo in tempo reale. Tale servizio è rivolto soprattutto ai possessori di auto di valore, ad aziende di trasporto, ai mezzi blindati, e a chiunque abbia necessità di tenere sotto controllo la propria flotta di veicoli. Tale servizio è possibile grazie ad un apparecchio dotato di ricevitore GPS e di un interfaccia GSM o UMTS per la comunicazione dei dati.

Teleallarme: questo servizio consiste nell'installazione di un sistema antintrusione in abbinata ad un sistema di teleallarme dove è necessario, collegati alla centrale operativa in modo da ricevere le eventuali segnalazioni di allarme e gestirle di conseguenza.

Telesoccorso: molto simile al teleallarme ma questa volta la periferica invia le segnalazioni di allarme alla centrale solo nel caso in cui la persona prema un pulsante di allarme e non in modo automatico.

Videosorveglianza: sistema complementare a quello di teleallarme o di telesoccorso avviene tramite l'utilizzo di telecamere collegate con la centrale operativa dell'istituto di vigilanza. Tale meccanismo permette di valutare la reale presenza di eventuali pericoli e di guidare i controlli.

Nella nostra trattazione ci occuperemo solo alcuni di questi servizi in particolare del teleallarme e della videosorveglianza oltre ad alcune funzionalità complementari e di supporto a queste attività.

2.1.1 La vigilanza di LIS

Fondata nel 1982, LIS si contraddistingue nel panorama italiano per l'eccellente qualità dei servizi e prodotti offerti, per l'elevato standard della

Figura 2.1: Foto di una Webu All-In-One

tecnologia usata e per la completezza dell'offerta proposta.[4].
LIS, non si occupa di tutti gli aspetti di vigilanza visti in precedenza, si concentra invece su quegli aspetti che permettono agli operatori LIS di individuare in modo rapido le minacce ed i pericoli ed agire di conseguenza inviando sul posto delle guardie giurate o contattando direttamente le forze dell'ordine nei casi più gravi cercando tuttavia di non contattare inutilmente il cliente.
I servizi di LIS si concentrano perciò principalmente sul teleallarme e telesoccorso affiancati nella maggior parte dei casi da meccanismi di videosorveglianza. Inoltre, LIS non sfrutta i normali ponti radio per la comunicazione con le centrali di sicurezza, ma sfrutta i mezzi di comunicazioni preesistenti nelle sedi da controllare in modo da avere sempre a disposizione più di un canale di comunicazione come ad esempio, linee PSTN e GSM utilizzate una come backup dell'altra; in alcuni casi più critici vengono, inoltre, installate anche periferiche di comunicazione supplementari per trasmettere eventuali guasti della centrale principale.

2.2 Le tecnologie di LIS

LIS sfrutta una serie di tecnologie innovative in tutti i settori del controllo di sicurezza, si parte dall'installazione dell'hardware dal cliente che solitamente comprende una centrale antintrusione di ultima generazione tra cui:

- Tecnoalarm
- UTC Fire & Security
- Bentel

Tutte centrali che permettono la ricezione degli allarmi tramite connessioni ADSL e GPRS, ed inoltre forniscono dei software di telegestione tramite gli stessi canali di comunicazione.
Affiancato alla centrale antintrusione solitamente LIS installa una centrale di backup come ad esempio la *Webu All-In-One* della *Urmet* questa centrale viene utilizzata come backup della centrale di sicurezza avendo alcuni ingressi programmabili, inoltre viene utilizzata come ponte per il vettore PSTN. Infine, dove possibile, viene installato anche un sistema di videosorveglianza che permette un analisi più approfondita della situazione. In particolare, dove il budget lo permette, viene installato un sistema *Dallmeier* che offre meccanismi di compressione delle registrazioni quando queste vengono consultate da remoto[1].

Figura 2.2: Schermata del software E-Pro

Quanto visto fino ad ora è quello che LIS installa dal cliente; vediamo adesso, invece, come LIS riceve e gestisce gli allarmi. In primo luogo, prima del nostro intervento, LIS utilizzava solo i vettori PSTN, GSM e solo per alcuni modelli della *UTC*, il vettore GPRS nonché per le periferiche di backup interne. Per ricevere gli allarmi su questi vettori utilizzavano due sistemi, il primo chiamato *System III* è un sistema di ricezione allarmi su PSTN che permette di ricevere gli allarmi tramite il protocollo Contact-ID o SIA su PSTN e quindi su linea telefonica tradizionale tramite i toni. Il secondo sistema è chiamato Osborne Hoffman LAN che permette, tramite un protocollo proprietario, di ricevere gli allarmi sul canale GPRS per alcuni modelli della *UTC Fire & Security*.
Questi due ricevitori permettono la ricezione degli allarmi, tuttavia, per permettere la gestione di tali allarmi LIS sfrutta un programma proprietario denominato *E-Pro* che permette la gestione degli allarmi presentando l'anagrafica del cliente, il posizionamento del sensore che è andato in allarme, una segnalazione esaustiva della tipologia di allarme ed altre informazioni che possono essere utili all'operatore, il quale una volta preso in carico una segnalazione la gestisce e compila un rapporto di gestione.

2.3 Cosa offre il mercato

Al nostro arrivo in LIS la situazione presente era quella specificata in precedenza e quindi le centrali supportate erano molto poco ed inoltre il vettore di comunicazione principale era il PSTN. Si è quindi deciso di analizzare che cosa offrisse il mercato per valutare se sostituire il software E-Pro o aggiornarlo per renderlo più competitivo rispetto alla concorrenza.
Adesso analizzeremo quali sono i principali prodotti che avrebbero potuto sostituire il software di Lis analizzando in dettaglio i punti a favore e quelli a sfavore della scelta. I software che analizzeremo sono:

- AteArgo di Urmet
- WebSat Enterprise di AMA Software
- Advisor Managment di UTC Fire & Security
- Mastermind e x-View di Enai

2.3.1 AteArgo

AteArgo è un prodotto che ha alle spalle vent'anni di sviluppo da parte di *Urmet*, esso si basa su un sistema Unix/Linux ed è stato il primo software per vigilanze private creato in Italia. Il continuo confronto con il mercato e l'aggiornamento tecnico costante anno permesso di perfezionare sempre più AteArgo e adattarlo alle reali esigenze delle centrali operative degli istituti di vigilanza.
La centrale AteArgo è composta da due server uno principale e uno di backup con allineamento automatico dei dati in maniera trasparente all'operatore. Il software è multi-operatore e multi protocollo e prevede di base la gestione dei teleallarmi radio, PSTN e GSM, inoltre per le centrali Urmet permette anche la ricezione degli allarmi tramite vettori GPRS e TCP/IP. Il software permette di gestire delle periferiche mobili GPS per applicazioni di teleallarme come l'invio della pattuglia più vicina in caso di allarme o la verifica del servizio di ronda. Nell'ultima versione del software è possibile, tramite l'ausilio di periferiche aggiuntive, l'acquisizione video automatica dalle centrale AteArgo a seguito di un evento di allarme. Infine è possibile la gestione di sistemi video navigabili via browser.
Oltre a queste funzionalità il sistema è in grado di eseguire alcune funzioni automatizzate come effettuare chiamate automatiche per particolari segnalazioni effettuare il backup a caldo sia della centrale principale che quella di riserva, invio di segnalazioni direttamente al cliente tramite SMS, gestione automatica di un call-center e di un agenda cliente.
AteArgo fornisce anche la possibilità di rilevazioni statistiche sia in formato elettronico sia via Web. possibilità di creare report personalizzati o l'interazione con altri sistemi di centralizzazione.
Per quanto riguarda la gestione multi operatore ad ogni operatore viene assegnato un profilo specifico in base al suo livello di specializzazione. Per accedere al sistema ogni operatore si identifica tramite login e password con le quali attiva le funzioni a lui destinate.
L'interfaccia utente permette di intervenire all'arrivo di un allarme con il minor numero di azioni. Inoltre vi è un costante monitoraggio del sistema e dei collegamenti periferici.
Oltre a queste funzionalità di base il sistema può essere espanso tramite moduli software aggiuntivi tra cui:

- comunication server: il sistema permette la comunicazione con altri sistemi informativi in modo bidirezionale assumendo la funzionalità di front-end.

- Utente ricaricabile: consente alla vigilanza di gestire clienti di tipo - ricaricabile ovvero con la possibilità da parte del cliente di attivare o disattivare il servizio di vigilanza solamente tramite l'invio di un SMS.

- Verbali: garantisce la rintracciabilità di tutte le azioni della centrale operativa e legate ad un allarme e supporta l'operatore nella gestione degli eventi.
- ArgoSound: permette la gestione di determinate segnalazioni tramite una chiamata vocale automatica.
- Fast call: compone automaticamente un numero dalla lista dei contatti del cliente in allarme evitando errori o perdite di tempo.
- Messaging: permette alla centrale di inviare automaticamente SMS, FAX o E-Mail verso recapiti preimpostati nel caso di particolari eventi.
- Modulo database: relaziona e rende disponibili tutti i dati archiviati e permette di eseguire qualsiasi tipo di ricerca, inoltre permette di interfacciarsi con altri sistemi. Permette la creazione di indici per la stima e la valutazione degli interventi.
- Modulo installatore: questo modulo permette di sollevare l'operatore dal supervisionare una nuova installazione dando la possibilità al tecnico di collegarsi alla centrale mediante portatile o smartphone e verificare la corretta ricezione degli allarmi dell'impianto sul quale sta intervenendo.
- Modulo GPS: permette di ricevere in centrale la posizione in tempo reale del parco pattuglie e di inviare sul sito in allarme la pattuglia più vicina.
- Modulo video: permette alla postazioni operatore di collegarsi a video server e telecamere IP in tempo reale.

2.3.2 WebSat Enterprise

WebSat Enterprase è un software sviluppato da AMA per la gestione di problemi di sicurezza ed emergenza degli ambienti, delle persone e dei veicoli. Esso consente la gestione di diverse tipologie di dispositivi di teleallarme come combinatori telefonici digitali (Contact ID, SIA, Fast Format), combinatori telefonici a sintesi vocale, teleallarmi radio. Inoltre tramite l'ausilio di periferiche AMA è possibile la gestione di teleallarmi via GSM/GPRS o IP, video allarmi, video sorveglianza con telecamere IP ed anche localizzazione satellitari per il controllo delle pattuglie, dei furgoni per il trasporto valori e qualsiasi tipo di veicolo.
Oltre a queste caratteristiche il software *WebSat Enterprise* permette un'interazione automatizzata verso il cliente tramite SMS o chiamate pre-registrate. Gestisce, inoltre, i servizi di pattuglia e di ronda tramite il sistema di localizzazione satellitare. Infine permette la gestione della videosorveglianza.

Per quanto riguarda l'aspetto amministrativo il software permette l'integrazione con i software amministrativi e gestionali già presenti in azienda ed è possibile predisporre un interfacciamento per la ricezione di allarmi provenienti da altri software.
Il software WebSat Enterprise è:

Multifunzionale: in quanto permette all'operatore di gestire tramite un unica interfaccia sia gli allarmi provenienti da periferiche mobili sia da impianti di allarme fissi.

Geo-referenziato: per ogni tipo di allarme sia esso fisso o mobile, vengono messe a disposizione dell'operatore il maggior numero di informazioni possibili.

Gestione delle pattuglie: Tramite la periferica WebSat Patrol la centrale permette di gestire in modo automatizzato le pattuglie in modo da ottimizzare le pattuglie sul territorio.

Report di comunicazione verso i clienti: la centrale è in grado di inviare avvisi di stato di allarme in modo automatico via SMS , invio automatico di report mensili, possibilità da parte del cliente di richiedere report parziali tramite l'invio di SMS.

Interfacciamento automatico con centralino telefonico: WebSat Enterprasie consente l'automazione delle chiamate verso i contatti telefonici del cliente in caso di allarme.

Interfacciamento con altri ricevitori: oltre ai ricevitori di AMA la Websat enterprise è in grado di interfacciarsi con tutti i ricevitori telefonici che implementano i protocolli ADEMCO 685, standard SurGuard, per ricevere gli allarmi in codice Contact ID, SIA level 2 e Ademco high Speed.

Ricezione allarmi da combinatori con sintesi vocale: la centrale di AMA è in grado di ricevere allarmi tramite chiamate effettuate da combinatori telefonici con sintesi vocale, solamente assicurandosi che il numero del chiamante sia in chiaro.

Gestione telecamere Axis: ad ogni sistema di teleallarme è possibile combinare una o più telecamere Axis con connessione diretta via Internet in modo da mettere a disposizione dell'operatore un sistema di videosorveglianza ad un costo contenuto.

Polling GPRS: per quelle periferiche che sono connesse tramite canale GPRS o TCP/IP la centrale è in grado di effettuare un polling ad intervalli molto brevi per verificare l'esistenza in vita della periferica e rilevare in modo tempestivo eventuali problemi di connessione.

2.3.3 Advisor Managment

Advisor Managment è una soluzione di UTC Fire & Security pensata per centralizzare la gestione della sicurezza di uno o più siti facenti riferimento ad un unica centrale di sicurezza. Tuttavia questo prodotto non è pensato specificatamente per le vigilanze bensì per piccoli complessi residenziali o commerciali aventi un unica centrale di monitoraggio.
Questo sistema si concentra sulle caratteristiche necessarie alla gestione della sicurezza in ambienti particolari nel quale accedono dipendenti e lavoratori, nel quale si hanno orari di lavoro flessibili. Solo la gestione integrata di controllo accessi, sicurezza antincendio e videosorveglianza permette ai security manager di avere una visione chiara e completa di tutto il sistema.
L'advisor managment offre un interfaccia intuitiva per la gestione di ambienti differenti, un'unica interfaccia per funzioni multiple, consente la creazione di report e rivolto alla gestione efficiente degli allarmi qualunque essi siano. Per quanto riguarda la videosorveglianza l'advisor managment è in grado di supportare tutti i videoregistratori di rete TrueVison, questa integrazione permette agli operatori di avere accesso sia alle telecamere in tempo reale sia agli eventi registrati consentendo così una verifica immediata degli eventi di allarme.
L'advisor management supporta le centrali antintrusione della linea advisor management e advisor advance questo permette una gestione combinata di sistemi antintrusione e di videosorveglianza, inoltre è possibile configurare aree ad accesso limitato. Nel caso di rilevazione di un allarme, questo viene evidenziato nell'area di competenza e le immagini vengono mostrate in modo automatico all'operatore. Advisor management permette inoltre l'integrazione con le centrali di rilevazioni incendi della serie FP sia per il monitoraggio degli eventi sia per tutto quello che riguarda la gestione dell'impianto. Tutto questo integrato in un unico software permette di intervenire in modo tempestivo in caso di incendio; infatti, in caso di allarme la posizione del sensore viene mostrata dinamicamente e viene attivato il flusso video in tempo reale per verificare la presenza dell'incendio. Inoltre, grazie al sistema di controllo accessi è possibile sbloccare tutte le porte in modo automatico, e gli ascensori portati a terra.

2.3.4 Mastermind e X-View

Mastermind è uno dei prodotti più utilizzati dalle maggiori industrie di sicurezza principalmente in US ma ultimamente anche nel resto del mondo sia nel settore privato che in quello pubblico. Questa diffusione è dovuta alla grande sicurezza e affidabilità del sistema unita alla possibilità di gestire sistemi su larga scala. I punti di forza di mastermind sono la possibilità di lavorare sia con ambienti di piccola scala sia con ambienti più grandi. Possibilità di configurazione multi sito con ridondanza a caldo. Integrazione

di molti sistemi in un unica interfaccia di sistema.
Mastermind è un sistema ad architettura aperta per fare in modo che i compratori possano adattare le loro tecnologie al sistema; esso permette la comunicazione tramite rete locale LAN e mette a disposizione degli SDKs per supportare al meglio lo sviluppo da parte di venditori di terze parti. come vendor di videosorveglianza.
Il sistema si suddivide in due parti slegate, una parte tratta il monitoring degli allarmi, l'altra parte, denominata *business suite* si occupa del contatto e della gestione del cliente. Tutte queste funzioni sono supportate da una serie di applicazioni che permettono una gestione ottimale del sistema come il MASVideo che permette di registrare sui server le immagini associate ad un allarme o il MASWeb utilizzato per consentire un accesso remoto alla reportistica a clienti con particolari esigenze. Oppure X-View un software affiancato a MasterMind che permette all'operatore di costruire la propria interfaccia personale e mette a disposizione maggiori informazioni come la localizzazione GPS o la piantina dell'edificio.
A differenza degli altri prodotti analizzati mastermind risponde alle esigenze impostate dal cliente tramite l'utilizzo di workflows che gestiscono tutti gli eventi in modo tale da guidare l'operatore lungo una sequenza predefinita di operazioni. Inoltre questo meccanismo permette di gestire in modo automatico le operazioni più semplici e che non richiedono la necessità dell'interazione con l'operatore.
La parte di *business suite* comprende tre sotto sezioni, la prima per definire le promozioni ed i prezzi da applicare ai clienti e organizza gli appuntamenti per le vendite. La seconda parte si occupa della relazione col cliente contenendo i dati del contratto la fatturazione dei servizi e il controllo dei pagamenti. l'ultima parte invece si occupa dell'organizzazione dell'installazione e dell'attivazione del servizio, si può occupare della gestione del magazzino mantenendo un inventario aggiornato. Le funzionalità di Mastermind sono:

VRT/IVR ovvero la possibilità di sfruttare un centralino automatico che risponde alle richieste più comuni dei clienti senza tenere occupato inutilmente un operatore di centrale, inoltre questa funzionalità può essere sfruttata per creare chiamate automatizzate verso il cliente nel caso di allarmi specifici.

Voice Recorder Integration: tutte le chiamate in entrata ed in uscita possono essere registrate e immagazzinate per un successivo controllo sia da parte del cliente che da parte della vigilanza nel caso in cui sia necessaria una verifica.

Report Server: questa funzionalità permette l'invio di email o messaggi periodici con il riassunto degli allarmi e degli interventi.

MASWeb: questa funzionalità permette ai clienti e ai tecnici di accedere alle loro informazioni per generare report o modificare i dati. Le pagine

web sono strutturate per includere i servizi di monitor e reportistica, l'elenco dei servizi e le informazioni di fatturazione. I tecnici possono modificare uno stato dell'impianto e inserirlo in modalità test per ricevere sul proprio browser o sull'app l'esito dei test effettuati durante una manutenzione o installazione.

MASmobile: questa app permette di controllare lo stato del sistema effettuare controllare lo storico eventi, agli installatori permette di mettere il sistema in modalità test per ricevere gli esiti.

Controllo accessi: è possibile integrare la ricezione degli allarmi con sistemi di controllo accessi

Capitolo 3

Obiettivi della tesi

L'obiettivo di questa tesi è quello di illustrare le modifiche, le aggiunte ed i cambiamenti effettuati al software di LIS per permettere un integrazione dei ricevitori di allarme in maniera più veloce e standard, una seconda parte della tesi complementare alla prima si focalizza invece sulla telegestione del delle centrali di allarme. Marginalmente tratteremo la videosorveglianza ed altri aspetti più pratici che riguardano la gestione di un allarme.

3.1 La situazione iniziale

Al nostro arrivo in LIS la situazione che si presentava era poco chiara e non ben definita. Gli operatori di centrale che gestivano gli allarmi utilizzavano un software denominato *E-Pro*. Questo programma è un adattamento di un programma utilizzato da Cobra SPA per la gestione degli allarmi provenienti da veicoli ed è stato adattato negli anni alla gestione degli impianti fissi.
Questo software è un formato da un insieme di moduli alcuni scritti in C ed altri in Java, la parte che riguarda la ricezione degli allarmi il loro immagazzinamento nel database e la logica che gestisce il comportamento da intraprendere per la loro gestione è implementato in una serie di moduli scritti in C; questi moduli sono chiamati:

- cp220_3
- cp220_4
- MTSfe_fissi

Questi tre moduli si occupano della ricezione degli allarmi dai vettori PSTN e GPRS/GSM per fare questo utilizzano dei ricevitori fisici chiamati

System III: si occupa della ricezione degli allarmi sul vettore PSTN tramite protocollo Contact ID.

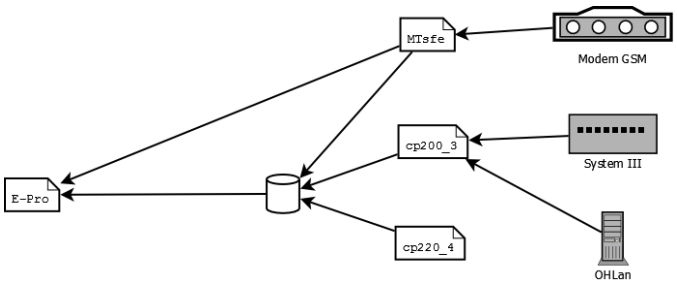


Figura 3.1: Schema dei moduli applicativi di LIS

OHLan: questo ricevitore è un PC fisico collegato in LAN sul quale è installato un software della UTC Fire & Security che si occupa della ricezione tramite protocollo proprietario degli allarmi provenienti dalle centrali UTC.

Modem GSM: più di un modem GSM è collegato ad una porta multiseriale connessa nella rete locale questi modem permettono la ricezione degli allarmi tramite vettore GSM e GPRS.

Per quanto riguarda la parte Java essa si occupa dell'interazione con l'operatore e quindi il compito di questi moduli è quello di prelevare gli allarmi dal database e mostrarli all'operatore, inoltre, questi moduli si occupano di tenere traccia di tutte le azioni eseguite dall'operatore. Infine, questa parte è strutturata in modo da garantire la multiutenza, infatti, questi moduli sono scritti in Java EE e sono eseguiti su di un server JBoss in modo da permettere l'esecuzione di più sessioni contemporaneamente.

3.1.1 cp220_3

Il *cp220_3* ha la funzione di leggere e trasformare gli allarmi provenienti dai sistemi di ricezione come il *System III* o lo *OHLan* e di immagazzinarli in un database non prima di averli trasformati in un formato interpretabile dall'E-Pro. questo formato è derivato direttamente dal Contact ID il prevedere diversi campi

```
ACCT MT QXYZ GG CCC S
```

dove i diversi campi indicano:

ACCT: è un identificativo assegnato al cliente

MT: è un numero che indica il tipo di messaggio, se esso è nuovo oppure una ritrasmissione.

XYZ: è un codice che indica il tipo di evento che è avvenuto

Q: un numero che può essere 1 o 3 ed indica se l'evento trasmesso è rispettivamente iniziato o finito.

GG: è il numero che identifica la partizione nella quale è stato generato l'allarme

CCC: è un numero che identifica il sensore o zona che ha generato l'allarme.

S: valore di checksum per il controllo degli errori

Il `cp220_3` utilizza i campi ACCT, XYZ, Q, GG, CCC insieme al timestamp nel quale arriva il messagio e li inserisce in una tabella del database dal quale poi saranno prelevati ed inviati all'operatore.
Oltre a questa funzione il modulo all'arrivo di ogni messaggio aggiorna un campo in una tabella chiamata *Ricevitori* per controllare periodicamente lo stato in vita dei ricevitori e la loro connessione con il modulo in questione.
Questo modulo è il più interessante per noi in quanto è quello che permette la ricezione degli allarmi e quindi sarà oggetto di una più approfondita analisi.

3.1.2 cp220_4

Questo modulo si occupa della logica della gestione degli allarmi per capire meglio il funzionamento di questo modulo facciamo un esempio e consideriamo un locale commerciale con prefissati orari di apertura e di chiusura. Il sistema di allarme viene disinserito poco prima dell'apertura, in questo caso alla centrale arriva l'evento di disinserimento dell'impianto ma il cp220_4 controlla l'orario di arrivo di tale evento e calcola che questa segnalazione è conforme all'orario di apertura e quindi lo registra ma non lo inoltra agli operatori di centrale. Nel caso in cui, invece, tale evento di disinserimento si verifica durante le ore notturne, e quindi fuori dal normale orario di apertura allora il software verifica l'anomalia e invia la segnalazione all'operatore che provvederà a gestirla.
Il `cp220_4` si occupa di capire quali segnalazioni inviare all'operatore. Oltre al controllo orario visto in precedenza, si occupa di filtrare le segnalazioni multiple provenienti ad esempio da ritrasmissioni, di filtrare gli impianti in test o disattivati che comunque inviano segnalazioni alla centrale.

3.1.3 MTSfe

Lo `MTSfe` è un software che deriva da *Cobra Italia* e si occupa principalmente della gestione delle periferiche di backup di derivazione automotive ovvero delle periferiche *PowerSat*. Queste periferiche erano state pensate per il controllo di autoveicoli e sono state adattate all'utilizzo negli impianti fissi come periferiche di backup in quanto sono dotate di una decina di ingressi programmabili e di alcune contatti di uscita.
Questo modulo pur essendo rimasto parte integrante del software di LIS è

ancora di proprietà di Cobra Italia è perciò stato impossibile per noi modificarlo, tuttavia uno dei nostri obiettivi a lungo termine era quello di sostituire le vecchie periferiche di backup PowerSat con altre più preformanti e aperte così da poter essere gestite direttamente dal nuovo software.

3.1.4 E-Pro

E-Pro è il software centrale che gestisce l'interazione tra operatore e sistema. Mentre i moduli in C si occupano di ricevere e selezionare gli allarmi il software E-Pro si preoccupa di prelevarli dal database e mostrarli all'operatore per poi aiutarlo nella gestione e controllo della segnalazione.
Questo software è composto da una serie di moduli JAVA EE che vengono eseguiti in un ambiente JBoss. Questi moduli hanno compiti diversi e svolgono le seguenti funzionalità:

- Chiedere in lis quando vai

3.2 Obiettivi della tesi

Gli obiettivi di questa tesi sono diversi e intervengono su diversi aspetti del software preesistente ma tutti questi obiettivi si prefiggono lo scopo principale di strutturare il software in maniera adeguata per permettere l'estensione delle funzionalità in modo rapido e strutturato permettendo così in futuro di non dover ripensare e riscrivere il software per integrare nuovi protocolli o aggiungere nuove funzionalità al software.

3.2.1 Integrazione di nuovi protocolli in minor tempo

Il primo degli obiettivi che vogliamo analizzare è quello dell'integrazione dei nuovi protocolli. Per prima cosa dobbiamo fare una piccola distinzione in due casi il primo caso si ha quando le segnalazioni di allarme arrivano da un ricevitore esterno come avveniva in precedenza con il System III in questo caso il protocollo da integrare è quello del ricevitore che si interpone tra la centrale di allarme ed il software di ricezione. Il secondo caso si ha quando la centrale d'allarme comunica direttamente con il software di ricezione. I due casi se pur distinti sono simili in quanto si può trattare il ricevitore esterno come una centrale d'allarme che comunica gli eventi con identificativi diversi.
Per fare ciò si è pensato di sostituire o comunque affiancare al cp200_3 un nuovo modulo che memorizza gli allarmi sulla base di dati nello stesso modo del cp200_3 per mantenere la retrocompatibilità del software. Questa soluzione è stata una scelta obbligata per non dover riscrivere interamente il software tuttavia, come vedremo nella sezione successiva, non è stata una scelta ottimale.

3.2.2 Strutturazione del software

Il secondo obiettivo che ci siamo prefissati è stato quello di dare al software una struttura più solida e modulare in modo da non dover riprogettare il software in futuro con la richiesta di nuove funzionalità come è stato necessario fare per l'integrazione che abbiamo dovuto fare noi. Per fare ciò si è deciso di strutturare anche la parte C in un software ad oggetti con conseguente passaggio obbligato al linguaggio C++. La scelta del linguaggio C++ rispetto al Java è stata una scelta puramente pratica in quanto le nostre conoscenze erano più sbilanciate verso questo linguaggio inoltre, la gestione delle periferiche seriali è più semplice utilizzando tale linguaggio.

3.2.3 Telegestione

Una funzione completamente nuova richiesta dalla centrale operativa era quella di poter telegestire le diverse centrali direttamente dal software E-Pro. Per fare ciò è stato necessario innanzitutto capire quali erano le funzioni necessarie per una corretta telegestione da parte dell'operatore. In secondo luogo è stato necessario capire quali centrali fossero in grado di permettere tali funzionalità e su quale vettore di comunicazione erano disponibili. Infine a livello pratico è stato necessario capire in quale modo implementare tali funzioni.
Obiettivi secondari alla telegestione sono stati la minimizzazione dei tempi di reazione del software e lo studio di una nuova interfaccia grafica per esprimere in modo immediato le nuove informazioni messe a disposizione dell'operatore.

3.2.4 Utilizzo di nuovi vettori di comunicazione

Gli obiettivi di rinnovamento del software non potevano limitarsi solamente ad un nuovo software ma, soprattutto, erano legati in modo indissolubile dall'utilizzo di nuovi vettori di comunicazione in particolare TCP/IP, GPRS ed SMS per minimizzare i costi di comunicazione e diminuire i tempi di dialogo tra la centrale di allarme e la centrale operativa.
L'utilizzo di questi vettori ha comportato lo studio di protocolli specifici come il Contact-ID o il SIA over IP e l'utilizzo e quindi l'interfacciamento del software con nuovo hardware come i modem GPRS.

3.3 Vincoli di progetto

Con il nostro arrivo sono state introdotte numerose modifiche al software preesistente tuttavia per mantenere l'operatività e l'usabilità durante tutto lo sviluppo è stato necessario effettuare alcune scelte che permettessero la retrocompatibilità e la normale esecuzione del software preesistente. In parti-

colare è stato necessario mantenere il meccanismo nel quale il cp220_3 memorizzava gli allarmi nel database per diverse ragioni. Questo memorizzazione consisteva nell'inserimento in tabella di un nuovo record così composto:

Ora ricezione:

Ora allarme:

Codice centrale:

Codice allarme:

Numero partizione:

Numero zona:

Gestito:

Il primo problema che si presenta anche se di facile soluzione si ha quando la segnalazione di allarme non porta con se l'ora della segnalazione, questo problema si risolve impostando l'ora della ricezione come ora in cui è avvenuta la segnalazione, tuttavia questo tipo di informazione è superflua in quanto non viene utilizzata in nessuna altra parte del software.
Il secondo problema più complesso consiste nel codice di allarme, esso è simile al Contact ID tuttavia mentre il Contact ID ha una struttura `QXYZ` dove Q è uno tra i numeri 1, 3 o 5 che indicano rispettivamente il nuovo evento, il termine di un evento e la continuazione di una segnalazione mentre XYZ è un codice identificativo del tipo di evento. I problemi che si presentano sono di tre tipi il primo puramente di forma consiste nel fatto che il codice di allarme veniva memorizzato nella forma `XYZQ` e questo comporta delle elaborazioni aggiuntive prima di memorizzare il dato. Inoltre il codice Contact ID non è univoco e quindi per centrali differenti avremmo significati differenti questo, in precedenza era stato risolto con una tabella nel database che associava ad un determinato tipo di centrale le corrispondenti etichette con il significato del codice. Questo meccanismo è rimasto invariato tuttavia questo comporta che ad ogni integrazione è necessario aggiungere centinaia di record a questa tabella per poter permettere la conversione. Un altro problema riscontrato è stato il fatto che non tutte le tabelle comunicano tramite protocollo Contact Id e questo ha comportato l'introduzione di una tabella di traduzione per convertire i codici provenienti da altri protocolli in codici Contact Id.
Un problema che abbiamo riscontrato sempre riguardante i codici Contact Id riguardava il valore del campo `Q` in quanto questo campo è utilizzato dal cp220_4 per effettuare il controllo orario di un impianto i codici 1401 e 3401 indicano rispettivamente l'inserimento ed il disinserimento da parte dell'utente dell'impianto, il cp220_4 sfrutta questi due codici per verificare

che tali eventi avvengano nelle fascie orarie prestabilite. Questo meccanismo funzionava correttamente per le centrali già integrate in quanto esse rispettavano questi codici per indicare gli inserimenti e i disinserimenti, tuttavia, per alcune centrali integrate durante il nostro percorso ci è capitato di trovare centrali che invertivano i due codici e questo ci ha obbligati ad utilizzare la tabella di traduzione per mantenere un comportamento corretto del software.
Il principale problema riscontrato tuttavia nell'utilizzo del codice precedente è stato proprio la comprensione del suddetto codice sorgente mal commentato, con refusi di vecchie funzioni provenienti da versioni precedenti e soprattutto la mancanza di una struttura logica del programma, basti pensare che il cp220_3 e il cp220_4 sono due programmi che svolgono due funzioni completamente distinte ma che in realtà provengono dallo stesso codice sorgente compilato con parametri differenti. Per ovviare a questo problema abbiamo innanzitutto introdotto il codice in un sistema di controllo versione, eliminato oltre diecimila righe di codice inutilizzato e commentato ogni singola funzione tramite un sistema di generazione della documentazione automatica. Questo ci ha permesso di comprendere le funzionalità del software anche se in alcuni casi non è servito a capirne il reale funzionamento o la logica con la quale è stato programmato; questo non ha causato grosse difficoltà ma potrebbe crearne quando si cercherà di sostituire il cp220_4 con un nuovo modulo meglio strutturato.

Capitolo 4

Ricevitori multi protocollo

In questo capitolo tratteremo la progettazzione e la realizzazione del nuovo ricevitore per la ricezione degli allarmi tramite vettori TCP/IP sfruttando sia le normali linee ADSL e GPRS.

4.1 Nuovi vettori di comunicazione: dal PSTN al TCP/IP

Con la diffusione della fibra ottica e delle comunicazione VoIP sono sempre meno le tradizionali linee PSTN che permettono la comunicazione di messaggi tramite i toni. Si è deciso perciò di passare alla comunicazione TCP/IP per comunicare gli eventi di allarme. Questo tipo di comunicazione permette di utilizzare sia connessioni ADSL che connessioni GPRS senza dover modificare nulla per quanto riguarda i protocolli di trasmissione.
Per poter utilizzare questa tipo di vettori di comunicazione sono stati adattati tre protocolli,uno standard e due proprietari per la trasmissione dei dati su TCP/IP. Il primo permette la ricezione direttamente dalle centrali di allarme, gli altri due pemettono la comunicazione con i ricevitori proprietari corrispondenti. Questi protocolli sono:

- SIA over IP
- Urmet AteArgo
- Bentel Visonic

4.1.1 SIA over IP

Per l'analisi e lo studio del protocollo si è fatto riferimento a due documenti della Security Industry Assocation, questi documenti sono ANSI/SIA DC-09-2007 per la trasmissione di eventi sulla rete internet e ANSI/SIA DC-07-2001 che descrive l'interfaccia di comunicazione. Questo protocollo permette

la comunicazione delle informazioni su linee ADSL o GPRS utilizzando lo stesso pacchetto tuttavia la rappresentazione dell'informazione si basa su due vecchi protocolli utilizzati per PSTN; questi due protocolli sono il SIA e il Contact ID. Partiremo perciò con la nostra trattazione con la struttura del pacchetto che è uguale in entrambi i casi.

La struttura del pacchetto

Il pacchetto di trasmissione è così formato:

$$\langle LF\rangle\langle CRC\rangle\langle 0LLL\rangle\langle''ID''\rangle\langle Sequence\#\mathbf{!}segment\#|\rangle$$
$$\langle \mathbf{R}reciver\rangle\langle \mathbf{L}line\#\rangle[data]\langle timestamp\rangle\langle CR\rangle$$

Dove i campi indicati sono rispettivamente:

LF: indica il carattere esadecimale `0x0A` e sta ad indicare l'inizio del pacchetto;

CRC: campo utilizzato per il controllo degli errori tramite un meccanismo di Cyclic redundancy Check a 16 bit;

0LLL: campo utilizzato per indicare la lunghezza del pacchetto. Essa è calcolata considerando come primo carattere le virgolette del campo ID e come ultimo il carattere ']';

"ID": il campo ID è una stringa che identifica la tipologia di messaggio contenuta nel campo `data` quelli più importanti per noi sono i tag:

- SIA-DCS,
- ADM-CID.

utilizzati per indicare che il messaggio contenuto nel campo `data` è un messaggio di allarme con la normale codifica SIA oppure Contact-ID;

Sequence#!segment#: questo campo è composto da due sotto-campi il primo, *Sequence* è un numero di quattro cifre obbligatorio e serve ad indicare il numero sequenziale del messaggio inviato, se questo valore è seguito dal carattere "`!`" allora è presente un secondo numero utilizzato soprattutto nel caso in cui il messaggio contenuto nel campo `data` fosse troppo lungo da non poter essere inviato in un solo messaggio;

Rreceiver: questo campo è valore composto da un numero variabile di cifre che precedute dalla lettera `R` che identificano chi sta trasmettendo, in molti casi questo valore conincide con il codice della centrale;

Lline#: campo variabile composto dalla lettera `L` seguita da 1 a 6 cifre ed indica la linea di ricezione;

Nome	Id	Descrizione
Tempo di occorrenza	"H"	Timestamp nel quale si è verificato l'evento
MAC Address	"M"	Mac address del dispositivo trasmettitore
Verifica	"V"	Informazioni riguardo ad audio o video associate l'evento
Testo di allarme	"I"	Breve testo che contiene informazioni riguardanti l'allarme o un commento
Nome del sito	"S"	Nome del sito nel quale è avvenuto l'allarme
Nome dell'edificio	"O"	Etichetta che contiene informazioni riguardanti l'edificio che ha generato l'evento.
Luogo	"L"	Indicazione precisa di dove è avvenuto l'evento segnalato
Longitudine	"X"	Longitudine del luogo
Latitudine	"Y"	Latitudine del luogo
Altitudine	"Z"	Altitudine del luogo

Tabella 4.1: Possibili valori iniziali per il campo `xdata`

[...data...]: stringa che contiene le informazioni da trasmettere essa è composta da una parentesi quadrata seguita dal numero identificativo della centrale preceduto dal carattere "#" e seguito dal carattere "|", dopo questo carattere è presente la vera informazione del messaggio, il delimitatore di fine stringa è una parentesi quadrata chiusa;

timestamp: un campo non obbligatorio che indica l'istante in cui il messaggio è stato accodato per l'invio, esso ha la seguente formattazione autoesplicativa

`_HH:MM:SS,MM-DD-YYYY`

CR: è il delimitatore finale del pacchetto e corrisponde al carattere ASCII corrispondente al valore esadecimale `0x0D`

Con lo standard DC09 viene introdotto anche un campo supplementare tra quello `data` ed il `timestamp` questo campo, denominato `xdata` anchesso compreso tra parentesi quadrata estende la potenza di espressione del protocollo. Il campo `xdata` inizia sempre con un carattere ASCII maiuscolo compreso nel range "G" e "Z" il cui significato è mostrato in Tabella4.1

La criptazione del pacchetto

La struttura del pacchetto appena mostrata è utilizzabile così com'è in caso di comunicazione tra una centrale di allarme e un ricevitore in ambiente locale. Tuttavia quando le informazioni devono viaggiare attraverso internet

è necessario che le informazioni siano protette in qualche modo. Per fare ciò lo standard DC09 prevede che i pacchetti siano criptati tramite algoritmo di criptazione AES che può utilizzare chiavi a 128, 192 o 256 bits.
Tuttavia non viene criptato l'intero pacchetto ma solamente il campo `data` dello stesso. Per indicare che il pacchetto è criptato si aggiunge il carattere "*" prima dell'etichetta nel campo ID.
Visto che la criptazione AES richiede che il pacchetto da criptare sia di lunghezza multipla di 16 per rispettare questo vincolo lo standard prevede l'inserimento di un campo `pad` composto da caratteri random tra il carattere "[" e il campo account all'interno del campo `data`.
Secondo lo standard è necessario che il software di ricezione implementi la criptazione tramite una qualsiasi delle tre chiavi, tuttavia nel nostro caso abbiamo implementato solo la criptazione con chiave a 128 bit lasciando ad una futura implementazione le altre due chiavi. Questa supposizione è valida in quanto le centrali di un determinato tipo implementano solamente un tipo di criptazione.

La connessione

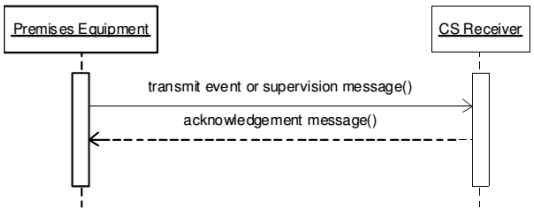
Lo standard DC09 prevede che la connessione tra il ricevitore e la centrale possa avvenire sia tramite l'utilizzo dello *User Data Protocol* (UDP) sia tramite l'utilizzo del *Transmission Control Protol* (TCP), il ricevitore da noi implementato permette attualmente solo l'utilizzo della modalità TCP, in quanto la modalità UDP è meno diffusa ed implementata tramite hardware con una scheda di espansione per il ricevitore System III.
Da notare il fatto che per la trasmissione tramite UDP nell'header del messaggio è necessario introdurre la porta della centrale d'allarme sulla quale il ricevitore dovrà inviare la risposta al messaggio. In Figura 4.1 vediamo la sequenza di messaggi che avviene durante la trasmissione di un evento.

I tipi di pacchetto

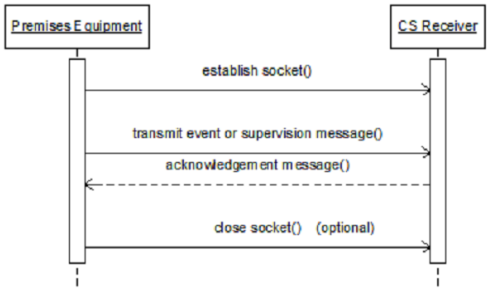
Dopo aver visto come sono strutturati i pacchetti di informazione vediamo ora quali informazioni possono essere trasmesse da questi pacchetti. Lo standard prevede una classificazione per le tipologie di pacchetti, in particolare si distinguono tre classi principali:

- Event Messages
- Supervisor Messages
- Acknowledgement Messages

Inoltre lo standard DC07 prevede per uno sviluppo futuro dei messaggi di *data/operation request* pensati per essere inviati dal ricevitore per richiede lo svolgimento di alcune operazioni o lo stato di alcuni componenti.



(a)



(b)

Figura 4.1: Esempio di trasmissione UDP 4.1(a) e TCP 4.1(b)

Event messages Gli event messages sono quei messaggi inviati dalla ceentrale di sicurezza per comunicare degli eventi al ricevitore. Essi rispettano lo standard appena descritto, il campo `ID` di questo tipo di messaggi può essere uno dei seguenti:

- SIA-DCS
- ADM-CID
- SIA-PUL
- ACR-SF
- ADM-41E
- FBI-SF
- SK-FSK1

tuttavia gli unici tag che noi supporteremo saranno quelli del SIA-DCS e ADM-CID i quali sono anche gli unici obbligatori secondo lo standard.

Supervisor message Questo tipo di messaggi non sono obbligatori, tuttavia sono molto consigliati, in quanto permettono di monitorare periodicamente lo stato della connessione. Questi messaggi sono inviati periodicamente dalla centrale di allarme al ricevitore, il tempo tra una trasmissione

e l'altra può essere impostato secondo lo standard da un minimo di 10 secondi ad un massimo di 1080 ore. Se nessun tipo di messaggio raggiunge il ricevitore in questo intervallo di tempo la comunicazione fallisce e il ricevitore dovrebbe segnalare la mancata comunicazione, inoltre, un evento di mancata comunicazione dovrebbe essere registrato dalla centrale.
Quando il sistema di supervisione è attivo, periodicamente la centrale invia un messaggio strutturato come in precedenza ma con `ID` uguale a *NULL* e campo `data` vuoto. Un esempio di messaggio è il seguente:

$$\langle LF\rangle\langle CRC\rangle\langle 0LLL\rangle\langle''NULL''\rangle\langle 0000\rangle$$
$$\langle \mathbf{R}recv\rangle\langle \mathbf{L}pref\rangle[]\langle timestamp\rangle\langle CR\rangle$$

Acknowledgment message Quando il ricevitore riceve dalla centrale un evento essp deve rispondere con un messaggio che può essere di quattro tipi:

- ACK
- NAK
- DUH
- RSP

Il messaggio di ACK corrisponde ad una risposta positiva e viene inviato quando il ricevitore riceve correttamente l'evento senza errori, un esempio di messaggio di ACK è il seguente:

$$\langle LF\rangle\langle CRC\rangle\langle 0LLL\rangle\langle''ACK''\rangle\langle seq\rangle$$
$$\langle \mathbf{R}recv\rangle\langle \mathbf{L}pref\rangle[]\langle timestamp\rangle\langle CR\rangle$$

dove i campi `seq`, `recv` e `pref` vengono copiati dal messaggio originale al quale si vuole dare una risposta.
Il messaggio di NAK è simile a quello di ACK tuttavia cambia il campo `ID` e il numero di `seq` che viene impostato a 0000, un esempio è riportato di seguito.

$$\langle LF\rangle\langle CRC\rangle\langle 0LLL\rangle\langle''NAK''\rangle\langle 0000\rangle$$
$$\langle \mathbf{R}recv\rangle\langle \mathbf{L}pref\rangle[]\langle timestamp\rangle\langle CR\rangle$$

Il pacchetto DUH viene inviato nel caso in cui il ricevitore, pur avendo verificato che il pacchetto è formattato correttamente e non contiene errori, non è in grado di tradurlo o comunque di interpretare la richiesta. In questo caso i campi `seq`, `recv` e `pref` sono uguali a quelli della richiesta ricevuta.
Il pacchetto RSP è stato introdotto come pacchetto di risposta per i messaggi di data/operation request tuttavia come questi pacchetti è pensato per un uso futuro. A differenza dei pacchetti precedenti il campo `data` contiene dei valori. Il numero di sequenza, del ricevitore e della linea sono copiati dal pacchetto di richiesta.

4.1.2 Urmet AteArgo

Il protocollo in questione è un protocollo proprietario di Urmet che serve per connettere il nostro sistema ad un ricevitore che permetta la gestione delle periferiche Webu All-In-One viste nel Capitolo2. Questa integrazione si è resa necessaria per cercare di dismettere il software proprietario di Cobra, lo *MTSfe*.
Essendo il protocollo proprietario non potremmo addentrarci in modo approfondito nel protocollo come nel precedente caso, tuttavia analizzeremo il funzionamento è la struttura generale del pacchetto.

La struttura del pacchetto

A differenza del protocollo precedente questo è un protocollo XML, ovvero, i pacchetti non sono alto che un unica stringa di caratteri che forma una strutta XML. Il pacchetto è formato da un tag inziale che delimita l'inizio del pacchetto, seguito da un `header` nel quale sono contenuti data e ora dell'evento. La parte di header è seguita dalla vera e propria informazione della segnalazione, questa parte denominata `body` ha un attributo che ne identifica il tipo di pacchetto trasmesso. All'interno del campo body, nel caso si tratti di una notifica di evento abbiamo tutte le informazioni riguardo la centrale che ha generato l'evento come il codice identificativo o i tipi di allarme generati. A differenza del protocollo precedente, è possibile inviare più eventi nello stesso pacchetto riguardanti anche zone diverse della stessa centrale. Inoltre, visto che la Webu All-In-One fornisce anche la funzione di ponte PSTN è possibile anche che vengano inviati eventi riguardanti sia la centrale di allarme principale che eventi generati dalla Webu in un unico pacchetto.
A differenza del protocollo precedente e di quello che analizzeremo in seguito in questo capitolo quello di AteArgo non prevede l'utilizzo di codici Contact-ID o SIA è stato quindi necessario prevedere anche un meccanismo di traduzione.

La connessione

La connessione diretta con le periferiche Webu è gestita dal ricevitore software AteArgo perciò la nostra connessione è di tipo locale con quest'ultimo software. Dato che le informazioni viaggiano su una rete locale e quindi potenzialmente protetta tramite altri meccanismi gli sviluppatori di Urmet non hanno incluso la criptazione del pacchetto.
La connessione con AteArgo è una connessione di tipo client-server dove il software AteArgo ricopre il ruolo di server e risponde a specifiche richieste è perciò necessario prevedere un meccanismo di *polling* per la ricezioni degli eventi, infatti, gli eventi saranno comunicati solamente come risposta ad una richiesta da parte del nostro software. Nel caso in cui vi siano eventi il

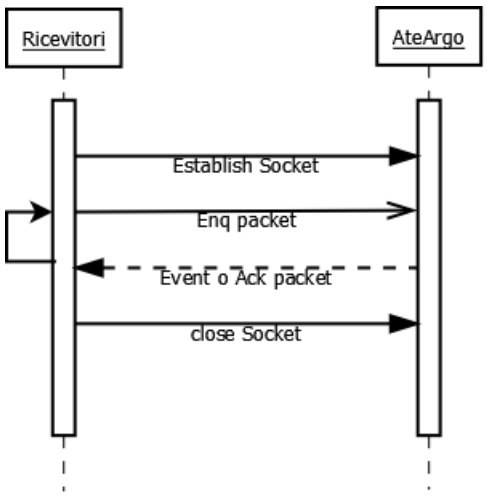


Figura 4.2: Comunicazione tra il software di ricezione e il software AteArgo

software risponde con un pacchetto di eventi altrimenti il sistema risponde con un pacchetto di ACK. Nella Figura 4.2 è mostrato il meccanismo di comunicazione tra i due software.

I tipi di pacchetto

In questo caso non possiamo addentrarci in particolare sulle tipologie di pacchetto che possiamo trovare nel protocollo tuttavia possiamo fare alcune distinzioni, i pacchetti che sicuramente incontriamo sono quattro:

- pacchetto di enquiry (richiesta);
- pacchetto di acknowledgment;
- pacchetto di evento;
- pacchetto di comando.

Pacchetto di enquiry Questo tipo di pacchetto è utilizzato dal client per richiedere informazioni al server. In questo caso il client ciclicamente continua ad inviare al server questo pacchetto ed il server può rispondere con un messaggio di acknowledgment o con un messaggio di evento.
Il blocco body di questo messaggio è praticamente vuoto se non per alcune informazioni riguardanti ora e data della richiesta necessarie per la sincronia con il server.

Pacchetto di acknowledgment In questo caso il pacchetto è inviato dal server al client ed è la risposta ad un messaggio di enquiry nel caso non vi siano eventi da segnalare oppure in risposta ad un messaggio di comando,

per confermare la corretta presa in carico della richiesta del client da parte del server.
Il blocco body di questo messaggio è praticamente vuoto se non per alcune informazioni riguardanti ora e data della richiesta necessarie per la sincronia con il server.

Pacchetto di evento Questo pacchetto è la risposta del server ad un messaggio di enquiry del client. In questo caso il messaggio contiene informazioni riguardo la centrale che ha generato un evento, su quale ingresso ha generato lo specifico evento ed un codice numerico che identifica la tipologia di evento generato. Inoltre, nel pacchetto vengono trasportate le informazioni che permettono di identificare se l'evento è stato generato dalla periferica Webu oppure, se è stato generato direttamente dalla centrale ed è passato tramite il ponte PSTN.

Pacchetto di comando Questa tipologia di pacchetto è inviata dal client verso il server per richiedere di eseguire un operazione su di una specifica periferica. Le operazioni che possono essere eseguite sono diverse tra cui l'impostazione della data e dell'ora sulla periferica, oppure l'attivazione o la disattivazione di un uscita o ancora l'esclusione di un ingresso.
In dettaglio ritorneremo su questo pacchetto nel capitolo successivo nel quale tratteremo la telegestione.

4.1.3 Bentel Visonic

In questo caso parliamo di un protocollo utilizzato per ricevere segnalazioni dalle centrali della serie *BW* tramite l'ausilio del ricevitore Visonic sempre fornito da Bentel. La particolarità di questa serie di centrali è la possibilità di associare ad un evento alcune immagini provenienti direttamente dai sensori IR e non da un impianto di videosorveglianza.

La struttura del pacchetto

Anche in questo caso il pacchetto non è altro che una stringa composta da tag XML. Tuttavia questa volta i pacchetti che possiamo ricevere sono solamente pacchetti di eventi, di immagini o di controllo delle connessione ai quali rispondiamo con pacchetti di acknowledgment.
La struttura del pacchetto è molto semplice, dopo un tag di apertura del pacchetto vi è una parte che identifica l'evento e la centrale che lo ha trasmesso, dopo queste informazioni in base al tipo di pacchetto possiamo avere diversi tag che rappresentano i diversi tipi di pacchetto, essi possono essere tag di `heartbeat`, di `frame` o di `event`.

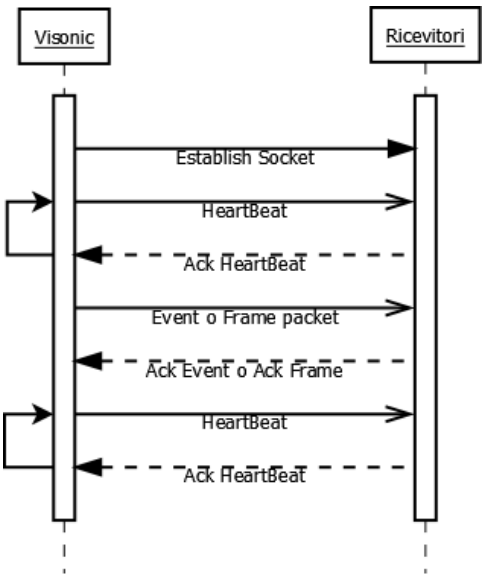


Figura 4.3: Comunicazione tra il software di ricezione e il software Visonic

La connessione

La connessione tra il nostro software e il ricevitore Visonic avviene tramite una connessione TCP/IP di tipo client-server, in questo caso il nostro software ha il ruolo di ricevitore ed è il ricevitore Visonic che effettua la connessione.
Per monitorare la connessione il ricevitore invia periodicamente dei pacchetti, in caso questi pacchetti non arrivino per un determinato intervallo di tempo si può identificare un problema nella connessione o direttamente nel ricevitore. Questi pacchetti che analizzeremo nel paragrafo successivo sono detti di *HeartBeat*. Un esempio del funzionamento è mostrato in Figura 4.3

I tipi di pacchetto

I pacchetti che il ricevitore Visonic scambia con il nostro ricevitore sono di tre tipi:

- pacchetto di HeartBeat;
- pacchetto di evento;
- pacchetto di frame.

Il nostro ricevitore risponde a questi messaggi con una risposta di ACK. Analizziamo ora, per quanto possibile i diversi pacchetti.

Pacchetto di HeartBeat Questo pacchetto serve a monitorare la connessione e in caso di problemi a ristabilirla. Esso ha la stessa struttura XML di un pacchetto d'evento tuttavia il suo contenuto è vuoto. In esso compaiono solo i tag che lo identificano come un pacchetto di heartbeat. Questo tipo di pacchetto viene inviato periodicamente al nostro software ricevitore. Nel caso in cui questo pacchetto non arrivi per un determinato periodo di tempo si può supporre che vi siano problemi con la connessione o più probabilmente con il ricevitori e si potrebbero prevedere delle contromisure da mettere in atto quando queste situazioni si verificano.

Pacchetto di evento Questo tipo di pacchetto viene inviato dal ricevitore Visonic quando un evento viene comunicato da una centrale, ogni pacchetto inviato contiene un unico evento che viene codificato sia tramite il protocollo proprietario Visonic sia tramite una stringa formattata secondo il protocollo Contact-ID. Per noi risulta più semplice tradurre la stringa formattata in questo formato in quanto il resto del sistema lavora sfruttando questi codici e avremmo comunque dovuto effettuare una traduzione di questo tipo.

Pacchetto di frame Questo particolare pacchetto è utilizzato per trasmettere le immagini provenienti dal sensore che ha generato l'allarme in particolare il pacchetto che viene generato è composto dalla prima parte nella quale viene riportato un identificativo dell'evento alla quale l'immagine è associata e all'interno dei tag `frame` viene codificata l'immagine proveniente dal sensore.
Per ogni evento si possono avere da uno a cinque fotogrammi, per ogni fotogramma generato viene inviato un pacchetto frame che conterrà anche il numero sequenziale del frame.

Pacchetto di acknowledgment Per ogni pacchetto appena visto il nostro ricevitore deve rispondere con un pacchetto di acknowledgment il quale sarà leggermente diverso per ogni tipologia di pacchetto ricevuto, infatti, l'acknowledgment per un pacchetto di heartbeat differisce dall'acknowledgment del pacchetto di event. In particolare i pacchetti di ACK sono strutturati come i corrispettivi pacchetti ricevuti con lo stesso id del pacchetto ricevuto ed il campo informativo vuoto.

4.2 La struttura dati

Come abbiamo detto nel Capitolo3 uno dei nostri vincoli è quello di mantenere la retrocompatibilità con il vecchio software di Lis fino al completo aggiornamento dei moduli. Per fare ciò è stato necessario mantenere la struttura dati precedente per far si che il resto del software potesse prelevare i dati senza nessuna complicazione. In particolare la tabella più importante

per la ricezione degli eventi era quella `allarmi_contact_id` i cui campi si ricavano dallo script di creazione del Listato 4.1

```
1  CREATE TABLE allarmi_contact_id
2  (
3          ac_id integer NOT NULL DEFAULT nextval(('"
               allarmi_contact_id_seq"'::text)::regclass),
4          ac_centrale character varying(20),
5          ac_allarme character varying(10),
6          ac_area bigint,
7          ac_zona bigint,
8          ac_giorno smallint,
9          ac_mese smallint,
10         ac_anno smallint,
11         ac_ora smallint,
12         ac_minuto smallint,
13         ac_secondo smallint,
14         ac_data_inserimento timestamp without time zone DEFAULT now()
               ,
15         ac_pending character(1) DEFAULT 's'::character varying,
16         ac_porta_seriale integer,
17         ac_n_ricevitore smallint,
18         ac_n_gruppo smallint,
19         CONSTRAINT allarmi_contact_id_pkey PRIMARY KEY (ac_id)
20 )
```

Listing 4.1: Tabella allarmi_contact_id

Come si nota i campi da compilare sono diversi anche se non tutti necessari. Il campo allarme contiene il codice Contact ID dell'allarme ricevuto, nel paragrafo successivo vedremo come questo meccanismo sia stato adattato per l'utilizzo anche dei codici di allarme SIA. Il campo `pending` serve al cp200_4 per identificare quali allarmi sono già stati processati in quanto questa tabella mantiene anche lo storico giornaliero degli allarmi ricevuti.
Oltre a questa tabella si è deciso anche di aggiornare una serie di tabelle collegate tra loro che hanno la funzione di monitorare lo stato dei ricevitori. Nel vecchio software per verificare la corretta esecuzione del software, ogni qualvolta che una segnalazione giungeva ad uno dei ricevitori veniva aggiornato un campo nella tabella `seriale` il quale è collegato alla tabella `ricevitori`; questo collegamento è mostrato in Figura 4.4. Il codice di creazione di queste tabelle è mostrato nel Listato 4.2.

```
1  CREATE TABLE seriale
2  (
3          se_id integer NOT NULL DEFAULT   nextval(('"seriale_se_id_seq
               "'::text)::regclass),
4          se_numero integer,
5          se_stato character varying(1) DEFAULT 's'::character varying,
6          se_allarme integer,
```

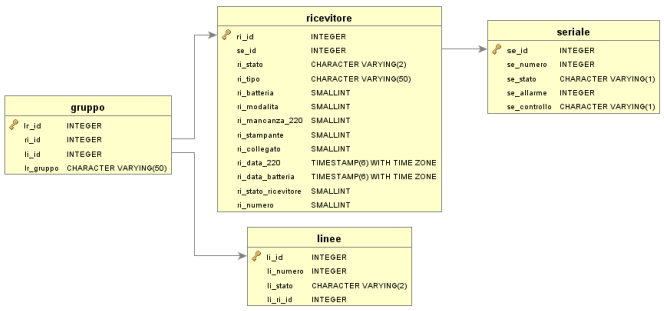


Figura 4.4: Schema relazionale delle tabelle che controllano i ricevitori

```
7         se_controllo character varying(1),
8         CONSTRAINT pk_seriale PRIMARY KEY (se_id)
9 )
10
11 CREATE TABLE ricevitore
12 (
13         ri_id integer NOT NULL DEFAULT  nextval(('"
               ricevitore_ri_id_seq"'::text)::regclass),
14         se_id integer,
15         ri_stato character varying(2) DEFAULT 'n'::character varying,
16         ri_tipo character varying(50),
17         ri_batteria smallint,
18         ri_modalita smallint,
19         ri_mancanza_220 smallint,
20         ri_stampante smallint,
21         ri_collegato smallint,
22         ri_data_220 timestamp with time zone,
23         ri_data_batteria timestamp with time zone,
24         ri_stato_ricevitore smallint,
25         ri_numero smallint,
26         CONSTRAINT pk_ricevitore PRIMARY KEY (ri_id),
27         CONSTRAINT fk_ricevito_reference_seriale FOREIGN KEY (se_id)
28         REFERENCES seriale (se_id) MATCH SIMPLE
29         ON UPDATE RESTRICT ON DELETE RESTRICT
30 )
```

Listing 4.2: Tabelle ricevitori

Questa meccanismo se pur non necessario per il corretto funzionamento del vecchio software è stato mantenuto, con dei piccoli adattamenti, per monitorare il funzionamento del nuovo software di ricezione. Tuttavia come si nota dalla complessità delle tabelle questo sistema porta con se anche degli elementi caratteristici del passato come l'utilizzo di una tabella `SERIALE` utilizzata dai tempi in cui i ricevitori erano ancora collegati tramite questo tipo di connessione cablata.
In realtà la soluzione migliore sarebbe stata quella di rappresentare i rice-

vitori in un unica tabella, e per quelli che presentavano possibilità di multithreading aggiungere una tabella collegata a ricevitori che tenesse traccia dei thread aperti e nel caso di blocco di uno di questi mettesse in atto le opportune contromisure.

4.3 Architettura del sistema

Analizziamo ora come è stato sviluppato il sistema. Si è deciso di passare ad un software strutturato per classi. L'idea era quella di avere un controllore che monitorasse periodicamente i diversi ricevitori e nel caso questi non fossero avviati oppure bloccati li riavviasse in automatico. I ricevitori dal canto loro dovevano comportarsi tutti pressoché alla stessa maniera, ovvero, le funzioni principali che dovevano svolgere erano quella di aggiornare il campo `controllo` della tabella seriale e caricare l'allarme sul database rispettando lo standard Contact-ID e i diversi campi della tabella `allarmi_contact_id` inoltre dovevano creare ed aggiornare i diversi campi delle tabelle seriale e ricevitore in modo da permettere al controllore di monitorare il loro stato. In Figura 4.5 vediamo lo schema delle classi del software Ricevitore.

4.3.1 Le classi

Analizziamo ora in dettaglio le classi principali del software di ricezione multi-ricevitore.

Ricevitore

La classe `Ricevitore` è una classe astratta che serve per generalizzare l'entità ricevitore, questa classe dovrà essere estesa ogni qualvolta si voglia integrare un nuovo ricevitore. Per quanto riguarda i suoi metodi i nomi sono abbastanza autoesplicativi, tuttavia in dettaglio il metodo *AggiornaRicevitore* serve per controllare i valori nelle tabelle `Ricevitori` e `Seriale` in caso esistano già i valori corrispondenti non fa altro che aggiornare il campo `se_controllo`, in caso contrario il metodo deve riempire la tabella con i valori assegnati dal ricevitore che invoca il metodo.
Il metodo *CaricaAllarme*, invece, si occupa di caricare l'allarme proveniente da uno dei ricevitori sul database. Esso effettua semplicemente una query di `INSERT` nella tabella `allarmi_contact_id`. Per gestire la concorrenzialità delle interrogazioni al database è stato necessario prevedere un meccanismo di esclusione tramite l'utilizzo di un *mutex* necessario per gestire l'esecuzione delle query in modo seriale. Inoltre sempre per impedire problemi di concorrenza gli attributi `nseriale` e `nricevitore` sono stati dichiarati *atomici*.
Il metodo `Run` è il metodo astratto che deve essere implementato dalla classe figlia e contiene la routine che il thread deve eseguire e deve contenere tutta

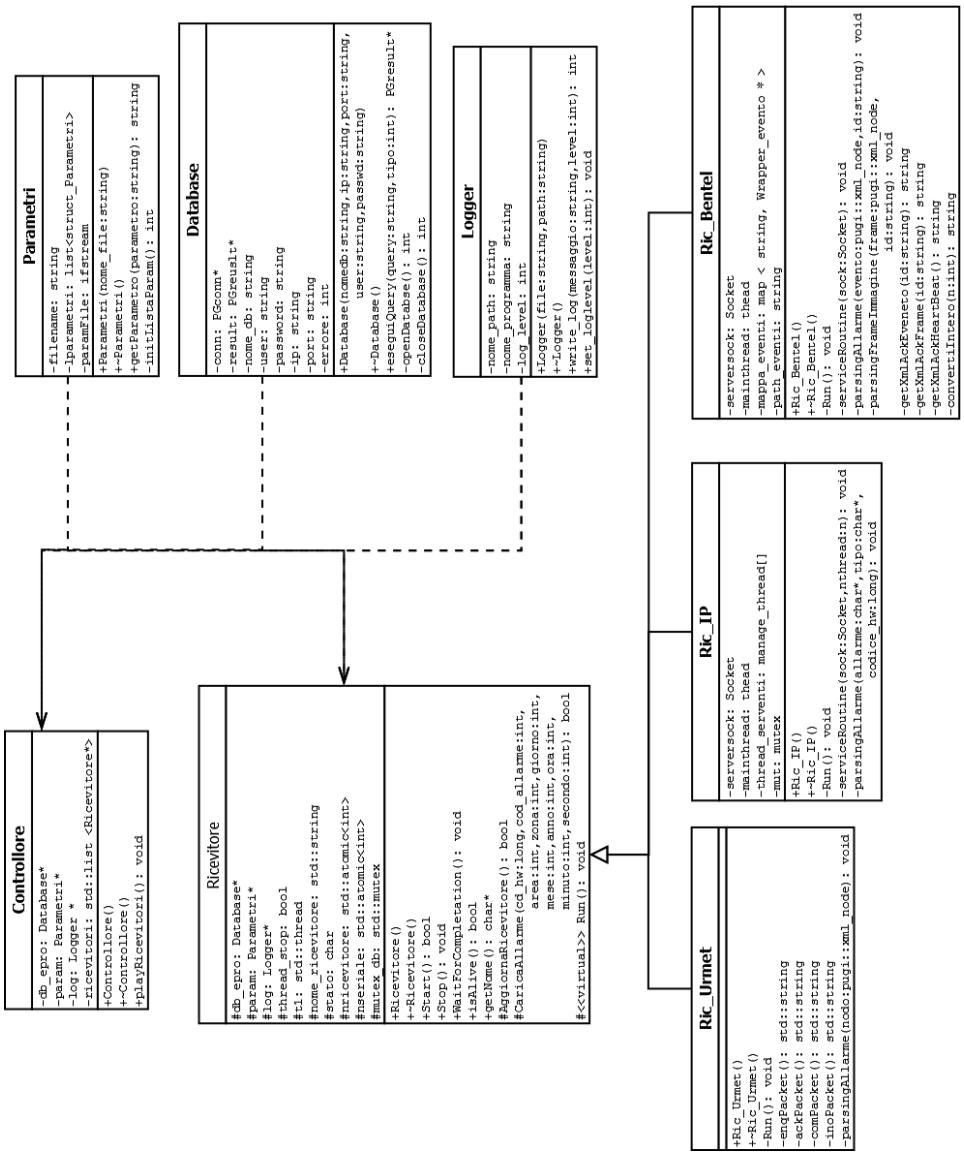


Figura 4.5: Schema delle classi Ricevitori

la logica di ricezione e parsing dell'allarme. Gli altri metodi della classe `Ricevitore` servono per gestire l'avvio, il blocco o il controllo del thread che esegue la routine.

Ric_IP

La classe `Ric_IP` è la classe che gestisce la ricezione degli allarmi inviati tramite protocollo *SIA over IP* ed estende la classe *Ricevitore* implementandone il metodo `Run`. In questo caso la classe è stata pensata come un server socket che dovrà essere eseguito all'interno del metodo `Run` il quale ad ogni nuova connessione inizializzerà un nuovo thread tramite la `serviceRoutine` questo metodo non fa altro che ricevere le informazioni, identificare il pacchetto ed inviare il corrispondente messaggio di acknowledgment o di NACK. Nel caso di un pacchetto di allarme la `serviceRoutine` invoca il metodo `parsingAllarme` il quale spacchetta le informazioni effettua eventuali traduzioni dei codici ed a sua volta invoca il metodo `CaricaAllarme` con gli opportuni parametri.

Ric_Urmet

La classe `Ric_Urmet` gestisce la ricezione degli allarmi generati dalle Webu collegandosi al ricevitore AteArgo. Anche in questo caso questa classe estende la classe astratta *Ricevitore*.
Per gestire la ricezione degli allarmi tramite il ricevitore AteArgo il metodo `Run` implementa un client socket che si connette al server AteArgo e inizia l'invio ciclico di pacchetti di *enqery* generati richiamando il metodo `enqPacket`, nel caso in cui riceva un messaggio di evento come risposta, tramite il metodo `parsingAllarme`, il software lo elabora e lo carica sul database. Come vedremo nel Capitolo5 questa classe si deve anche occupare dell'invio di comandi alle periferiche Webu.

Ric_Bentel

Anche questa classe estende quella *Ricevitore*, essa è pensata per ricevere e gestire gli allarmi provenienti dal ricevitore Visonic. Per fare questo il metodo `Run` esegue un socket server che attende la connessione del ricevitore Visonic. Pur attendendo una sola connessione si è pensato di strutturare il server in modo tale da eseguire un thread per soddisfare le richieste di un client in modo tale da prevedere la possibilità di gestire anche più ricevitori Visonic in parallelo. Per soddisfare questa esigenza quando il server riceve una connessione esso lancia un thread che esegue la `serviceRoutine` che gestisce la ricezione dei pacchetti. In caso la service routine riceva un pacchetto di HeartBeat essa risponde con il corrispondente ack generato tramite

il metodo `getXmlAckHeartBeat`, in caso il pacchetto ricevuto sia un pacchetto di segnalazione di evento allora il software analizza l'allarme tramite il metodo `parsingAllarme` una volta che il pacchetto è stato analizzato allora il sistema risponde con un messaggio di ack generato tramite il metodo `getXmlAckEvento`. Nel caso in cui la serviceRoutine riceva un pacchetto contenente un frame immagine invoca prima il metodo `parsingFrameImmagine` per analizzare il pacchetto ed estrapolarne le informazioni ed in secondo luogo risponde tramite un pacchetto generato da `getXmlAckFrame`.

Controllore

Questa classe ha il compito di creare e lanciare i diversi ricevitori, inoltre essa si occupa di monitorarli ed in caso di anomalia di rieseguirli. Per fare questo la classe mantiene una lista di `ricevitori` ed esegue perennemente il metodo `playRicevitori` il quale si occupa per ogni elemento della lista di controllare che esso sia in esecuzione controllando sia che esso sia vivo sia che esso abbia cambiato il suo valore di controllo e nel caso in cui esso sia bloccato si occupa di re-istanziarlo e di eseguirlo.

Parametri

Questa classe è stata pensata per supportare l'esecuzione del programma, in particolare essa si occupa di aprire e leggere da un file i parametri necessari per l'esecuzione del programma come ad esempio l'indirizzo IP del database, il nome utente necessario per la connessione, o ancora i parametri di connessione dei vari ricevitori. In particolare il metodo `initListaParametri` apre il file legge i diversi parametri suddivisi per riga e li carica in una lista chimata `lparametri` la quale è una struttura che contiene il nome del parametro ed il suo valore.
Tramite il metodo `getParametro` le altre classi possono recuperare il valore assegnato ad un determinato parametro.

Database

Questa è un'altra casse di supporto che gestisce la connessione e le interrogazioni al database, in particolare il metodo `eseguiQuery` riceve in ingresso una stringa che contiene la query da eseguire ed un campo integer che indica se la query si aspetta un risultato oppure no. Questo meccanismo ci permette di eseguire la query in modo tale da non dover analizzare il risultato in caso di inserimenti del database.

Logger

Questa è l'ultima classe di supporto e fornisce gli strumenti per eseguire un sistema di log anche con diversi livelli di notifica. In particolare, istanziando

un nuovo logger per istanze diverse di classi è possibile specificare per ogni istanza quale sia il livello di log da mantenere e quale nome associare al messaggio di log.

4.3.2 Implementazione

Innanzitutto si è deciso di sviluppare il software in linguaggio C++ aggiornato allo standard del 2011 (ISO/IEC 14882:2011[2]) il quale supporta meglio il multi-threading e per fare ciò si è deciso di utilizzare il compilatore gcc-4.8 l'ultimo rilasciato al momento dell'implementazione. I vantaggi sono la possibilità di utilizzare i thread nativi del linguaggio e anche i tipi *atomici* non presenti nelle versioni precedenti.
Oltre alla libreria standard, inoltre, si è deciso di utilizzare la libreria esterna *pugixml*[3] per il parsing e l'analisi dei pacchetti XML. La scelta è ricaduta su questa libreria in quanto molto leggera e di facile utilizzo inoltre è supportata da diverso tempo ed è quindi molto stabile.
Analizziamo ora in particolare il codice di alcune classi e metodi significativi.

Controllore

Per la classe `Controllore` analizziamo il metodo `playRicevitori` mostrato nel Listato 4.3

```
1  void Controllore::playRicevitori()
2  {
3          std::list<Ricevitore *>::iterator it;
4
5          cout<<"Avvio␣dei␣ricevitori␣tramite␣metodo␣playRicevitori"<<
               endl;
6          syslog(LOG_INFO, "Avvio␣dei␣ricevitori␣tramite␣metodo␣
               playRicevitori");
7          for (it = ricevitori.begin(); it != ricevitori.end(); it++)
8          {
9                  syslog(LOG_WARNING,"Il␣ricevitore␣%s␣e'␣non␣avviato␣lo
                       ␣avvio",(*it)->getNome());
10                 (*it)->Start();
11         }
12         while (true)
13         {
14                 for (it = ricevitori.begin(); it != ricevitori.end();
                       it++)
15                 {
16                         if(!(*it)->isAlive())
17                         {
18                                 syslog(LOG_WARNING,"Il␣ricevitore␣%s␣e'
                                       ␣non␣avviato␣lo␣avvio",(*it)->
                                       getNome());
19                                 (*it)->Start();
```

```
20                          }
21                  }
22                  sleep(50);
23          }
24  }
```

Listing 4.3: Metodo playRicevitori

Questo è il metodo richiamato dal main dopo l'istanziazzione di un Controllore, dopo una prima fase di avvio di tutti i ricevitori si controlla ciclicamente che essi siano in vita tramite il metodo `isAlive()` e nel caso non lo siano si riavviano. Questo controllo ciclico viene fatto su tutti i ricevitori presenti in una lista di puntatori a `Ricevitori` e per scorrere questa lista si usa un iteratore.

Ricevitori

La classe Ricevitori è una classe astratta dato che il metodo Run è dichiarato astratto. Al contrario del JAVA in C++ la classe non deve essere dichiarata astratta ma basta che essa contenga un metodo astratto perchè la classe lo sia. In questo caso il metodo `Run` è così dichiarato:

```
1  virtual void Run ()=0;
```

Listing 4.4: Metodo astratto Run

Il metodo `Start` invocato da playRicevitori inizializza ed esegue il metodo Run come thread tramite il codice illustrato nel Listato 4.5.

```
1  void Ricevitore::Start()
2  {
3          t1 = std::move(std::thread(&Ricevitore::Run, this));
4          thread_stop = false;
5  }
```

Listing 4.5: Metodo Start

Per quanto riguarda i due metodi principali della classe ricevitore essi sono mostrati nel Listato 4.6 e Listato 4.7.

```
1  bool Ricevitore::AggiornaRicevitore(int nseriale, int nricevitore) {
2          std::string query;
3          long serial_id, ri_id, li_id;
4          PGresult * res;
5          syslog(LOG_DEBUG,"Acquisizione␣mutex␣update␣seriale;");
6          mutex_db.lock();
7          query.clear();
8          query="SELECT␣se_id␣FROM␣ricevitore␣WHERE␣ri_tipo␣=␣'"+
               nome_ricevitore+"';";
9          syslog(LOG_DEBUG,"Query:␣%s",query.c_str());
10         res = db_epro->eseguiQuery(query,1);
```

```
11          if(PQntuples(res) > 0) {
12                  serial_id=atol(PQgetvalue(res,0,0));
13                  PQclear(res);
14                  stato++;
15                  if(stato > 'z') stato = 'a';
16                  query = "UPDATE seriale SET se_controllo = '";
17                  query += stato;
18                  query += "' WHERE se_id = '"+std::to_string(serial_id)
                        +"';";
19                  db_epro->eseguiQuery(query,0);
20                  mutex_db.unlock();
21                  syslog(LOG_DEBUG,"Rilascio mutex update seriale;");
22                  return true;
23          } else {
24                  PQclear(res);
25                  query="INSERT INTO seriale (se_numero, se_stato,
                        se_allarme, se_controllo) VALUES ('"+std::
                        to_string(nseriale)+"','s',0,'a');";
26                  db_epro->eseguiQuery(query,0);
27                  query.clear();
28                  query = "SELECT se_id FROM seriale WHERE se_numero = "
                        +std::to_string(nseriale)+";";
29                  syslog(LOG_DEBUG,"Query: %s",query.c_str());
30                  res = db_epro->eseguiQuery(query,1);
31                  serial_id = atol(PQgetvalue(res,0,0));
32                  PQclear(res);
33                  query.clear();
34                  query = "INSERT INTO ricevitore (se_id,ri_stato,
                        ri_tipo,ri_batteria,ri_modalita,ri_mancanza_220,
                        ri_stampante,ri_collegato,ri_numero) VALUES('"+
                        std::to_string(serial_id)+"',0,'"+nome_ricevitore+
                        "',0,0,0,0,0,"+std::to_string(nricevitore)+");";
35                  syslog(LOG_DEBUG,"Query: %s",query.c_str());
36                  db_epro->eseguiQuery(query,0);
37                  query.clear();
38                  query = "SELECT ri_id FROM ricevitore WHERE ri_tipo =
                        '"+nome_ricevitore+"';";
39                  syslog(LOG_DEBUG,"Query: %s",query.c_str());
40                  res = db_epro->eseguiQuery(query,1);
41                  ri_id = atol(PQgetvalue(res,0,0));
42                  PQclear(res);
43                  mutex_db.unlock();
44                  syslog(LOG_DEBUG,"Rilascio mutex insert ricevitore;");
45                  query.clear();
46                  return true;
47          }
48  }
```

Listing 4.6: Metodo AggiornaRicevitore

Nel metodo `AggiornaRicevitore` la prima operazione eseguita è quella di richiedere il *lock* sul database in modo da sincronizzare i diversi thread e non avere problemi di concorrenzialità sull'accesso al database, come secondo passo si effettua una query sul database per verificare l'esistenza del ricevitore sul database. Se l'esito di questa interrogazione è positivo ed esiste almeno un valore allora viene aggiornato il campo `se_controllo` della tabella seriale tramite un operazione di update sul database, dopo di che viene rilasciato il *mutex* sul databse. Nel caso in cui invece, la prima interrogazione non restituisca alcun risultato significa che vi sono stati problemi con il controllore ed il sistema è stato riavviato in questo caso il ricevitore si occupa di reinserire i suoi valori nelle diverse tabelle tramite una serie di operazioni di insert, dopo di che anche qui viene rilasciato il mutex e il metodo termina.

```
1  bool Ricevitore::CaricaAllarme(long cd_hw, int cod_allarme, int area
       , int zona, int giorno, int mese, int anno, int ora, int minuto,
        int secondo) {
2        std::string query;
3        std::time_t tnow = std::time(NULL);
4        struct tm * now = localtime( &tnow );
5
6        syslog(LOG_NOTICE, "Scrittura␣allarme␣contact␣Id␣Centrale␣%ld
             ␣Allarme:␣%d,␣Area:␣%d␣Zona:␣%d␣Data:␣%d:%d:%d␣%d/%d/%d",
             cd_hw, cod_allarme, area, zona, ora, minuto, secondo,
             giorno,mese,anno);
7        if((giorno == 0) || (mese == 0) || (anno == 0)) {
8              giorno = now->tm_mday;
9              mese = now->tm_mon+1;
10             anno = now->tm_year+1900;
11             ora = now->tm_hour;
12             minuto = now->tm_min;
13             secondo = now->tm_sec;
14       }
15       query = "INSERT␣INTO␣allarmi_contact_id␣(ac_centrale,␣
             ac_allarme,␣ac_area,␣ac_zona,␣ac_giorno,␣ac_mese,␣ac_anno
             ,␣ac_ora,␣ac_minuto,␣ac_secondo,ac_porta_seriale,
             ac_n_ricevitore,ac_n_gruppo)␣VALUES␣('"+std::to_string(
             cd_hw)+"','"+std::to_string(cod_allarme)+"','"+std::
             to_string(area)+"','"+std::to_string(zona)+"','"+std::
             to_string(giorno)+"','"+std::to_string(mese)+"','"+std::
             to_string(anno)+"','"+std::to_string(ora)+"','"+std::
             to_string(minuto)+"','"+std::to_string(secondo)+"','"+std
             ::to_string(nseriale)+"','"+std::to_string(nricevitore)+"
             ','1');";
16       cout<<query<<endl;
17       syslog(LOG_DEBUG,"Query:␣%s",query.c_str());
18       mutex_db.lock();
19       syslog(LOG_DEBUG,"Acquisito␣mutex␣query␣Carica␣allarmi");
```

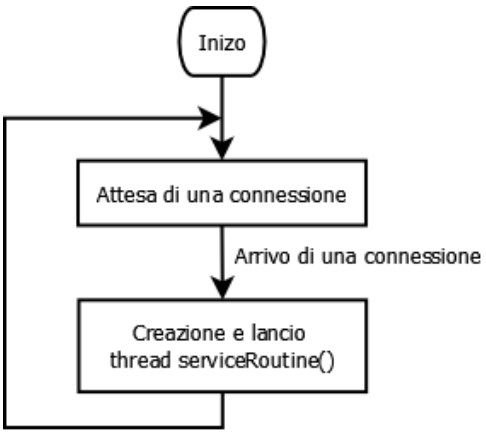


Figura 4.6: Diagramma di flusso per l'esecuzione del metodo Run

```
20          db_epro->eseguiQuery(query,0);
21          mutex_db.unlock();
22          syslog(LOG_DEBUG,"Rilasciato␣mutex␣query␣Carica␣allarmi");
23          return true;
24  }
```

Listing 4.7: Metodo CaricaAllarme

Nella prima parte del metodo CaricaAllarmi si verifica che la data inserita sia valida, in quanto nel caso in cui l'allarme non porti con se le informazioni riguardanti il timestamp dell'evento si è deciso di utilizzare il timestamp di ricezione. Per fare ciò si passano al metodo il valore *0* per i parametri giorno, mese ed anno. In questo caso il sistema si occupa di prelevare la data corrente e inserirla nei campi adeguati. Dopo questa operazione si prepara la stringa per la query di inserimento ed infine si acquisisce il mutex e si esegue la query sul database.

Ric_Ip

Per quanto riguarda questa classe analizzeremo in dettaglio il metodo `Run()` ed il metodo associato `serviceRutine()` e per farlo utilizzeremo i diagrammi di flusso per mostrarne l'esecuzione. In Figura 4.6 vediamo come avviene l'esecuzione del metodo `Run()`. Vedaimo come la sua logica sia veramente semplice infatti si tratta di un server socket che attende la ricezione di una connessione. All'arrivo di una nuova connessione esso crea una nuova istanza di thread e tramite questo esegue il metodo `serviceRoutine()`. Dopo di che si rimette in attesa di una nuova connessione. Per quanto riguarda la `serviceRuntine` il suo funzionamento è illustrato in Figura 4.7 in questo caso il metodo inizia la sua esecuzione ponendosi in attesa di ricevere un pacchetto, nel caso che, dopo un tempo prestabilito, non si riceva alcun pacchetto il metodo va in timeout e termina la sua esecuzione. Nel caso,

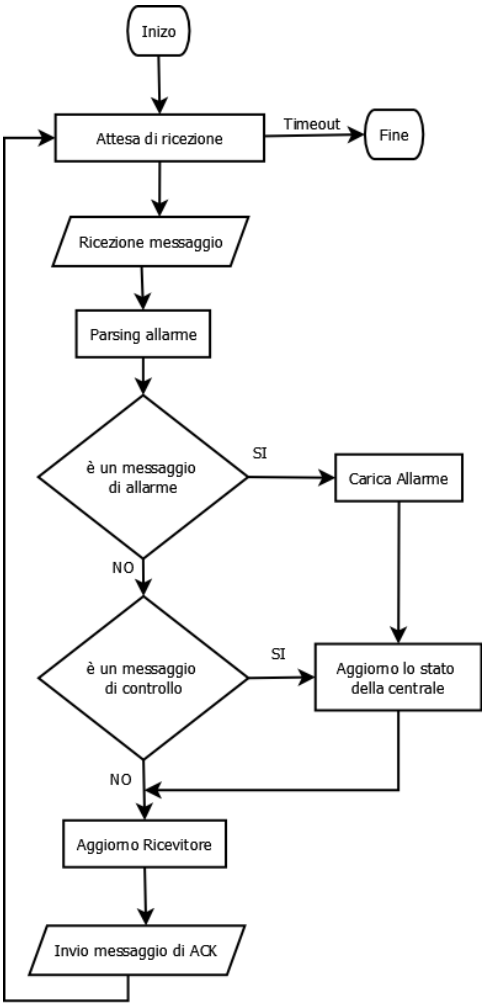


Figura 4.7: Diagramma di flusso per l'esecuzione del metodo serviceRoutine

invece, in cui si riceva un messaggio si suddivide il messaggio in parti e controllando ognuna delle parti si decide se esso è un messaggio di allarme, allora si invoca il metodo `CaricaAllarme` se esso è un messaggio di controllo allora si aggiorna solamente lo stato della centrale per far si che non venga segnalata una anomali di connessione. Fatto ciò si invoca il metodo `AggiornaRicevitore` che va a modificare il campo di controllo e si invia il messaggio di ACK alla centrale che sta comunicando.

Ric_Urmet

Per quanto riguarda il funzionamento del ricevitore AteArgo la logica è leggermente diversa, anche in questo caso analizzeremo in particolare il metodo `Run()` tramite l'utilizzo dei diagrammi di flusso. In Figura 4.8 vediamo il diagramma di flusso che mostra il funzionamento del metodo. Vediamo come la prima operazione che svolge il metodo è quella di connettersi al ricevitore AteArgo, a questo punto il metodo invia un pacchetto di ENQ ed attende la risposta. Quando riceve tale risposta controlla se si tratta di un allarme ed in caso affermativo lo memorizza tramite il `CaricaAllarme`, nel caso invece sia un pacchetto di ACK aggiorna solamente il ricevitore. Oltre a questo meccanismo che viene ripetuto all'infinito a meno di non avere problemi di connessione, il metodo si occupa anche della gestione dei comandi da inviare alle diverse periferiche, questa caratteristica sarà però analizzata in particolare nel capitolo successivo.

Ric_Bentel

La classe Ric_Bentel è simile a quella Ric_IP anche in questo caso il metodo `Run()` si mette in attesa di una connessione ed il suo comportamento è esattamente uguale a quello della classe Ric_IP. Quello che cambia è il comportamento del metodo `serviceRoutine` come viene mostrato in Figura 4.9. In questo caso il metodo si mette in attesa di ricevere un messaggio a questo punto controlla che esso sia uno dei tre possibili messaggi che possono arrivare, nel caso sia un messaggio di evento allora il metodo invoca il `CaricaAllarme`, nel caso in cui il messaggio contenga un frame di un immagine allora il metodo lo elabora e salva l'immagine, nel caso in cui invece il messaggio sia un semplice HeartBeat allora il metodo invoca `AggiornaRicevitore` e risponde al messaggio con un ACK dipendente dalla tipologia di messaggio ricevuto e poi si rimette in attesa di ricevere il pacchetto successivo.

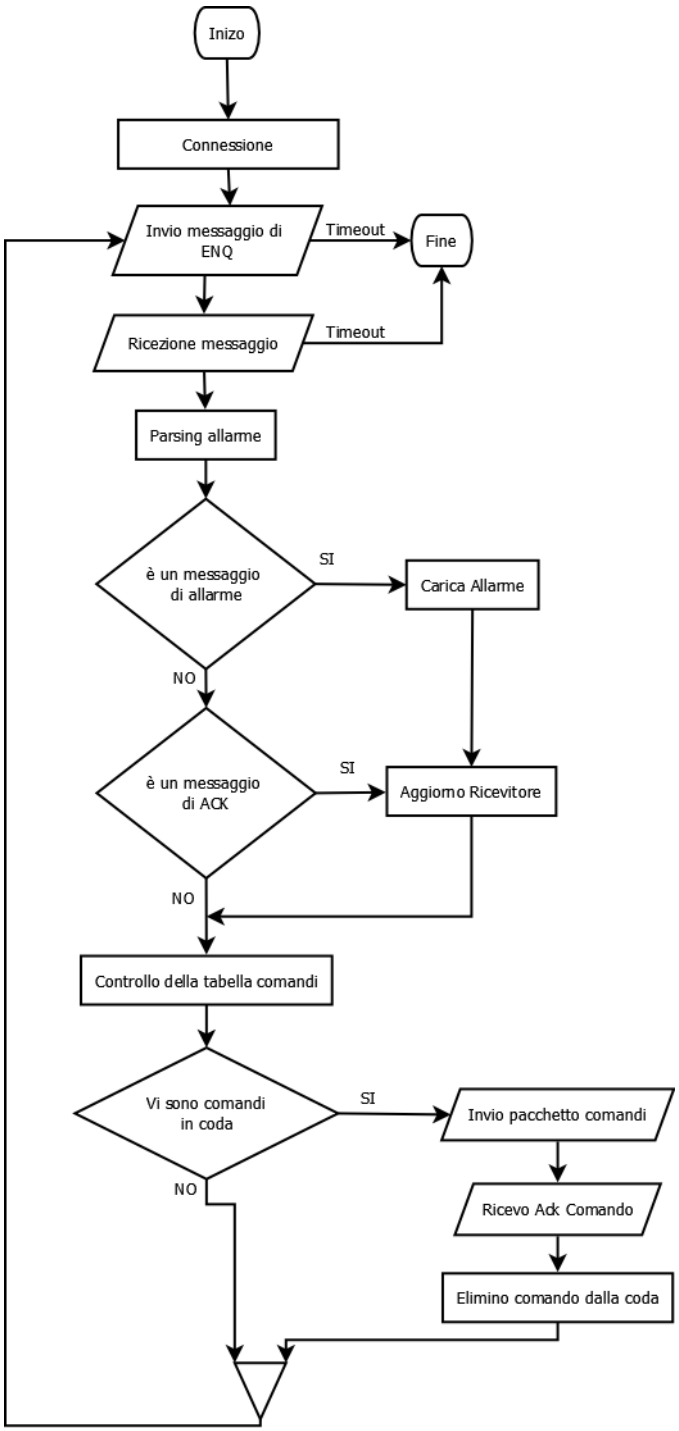


Figura 4.8: Diagramma di flusso per l'esecuzione del metodo Run della classe Ric_Urmet

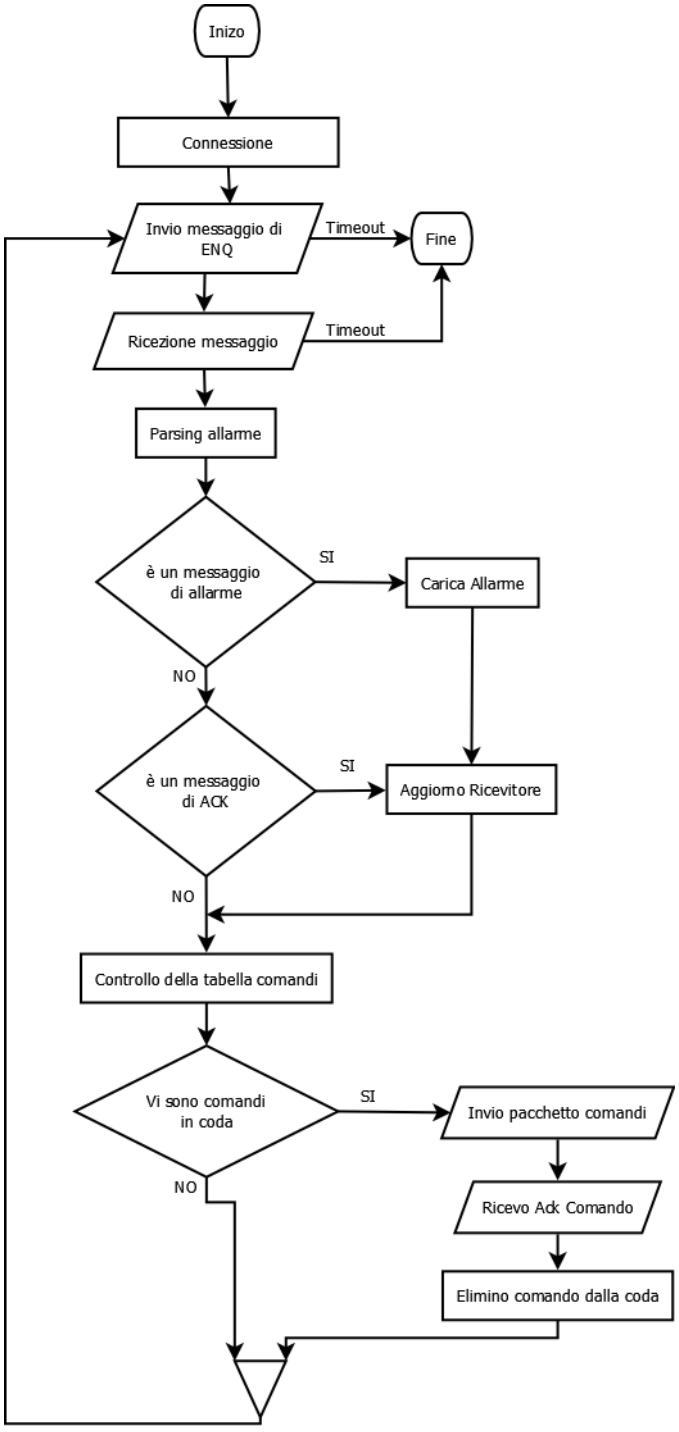


Figura 4.9: Diagramma di flusso per l'esecuzione del metodo serviceRoutine della classe Ric_Bentel

Capitolo 5

Telegestore

In questo capitolo analizzeremo la realizzazione di un software pensato per inviare dei comandi alle centrali e alle periferiche che permettono la telegestione. In particolare in questo capitolo ci concentreremo sul software che gestisce la comunicazione tra la centrale operativa e le diverse centrali sia come comunicazione diretta sia tramite l'utilizzo di ricevitori.
Nel capitolo successivo, invece, concentreremo la nostra analisi sul modulo che permette all'operatore di inviare i comandi al software che analizzeremo in questo capitolo.
In particolare questo software comunicherà con le centrali della marca *Tecnoalarm* che permettono una connessione diretta per la telegestione tramite l'ausilio di un protocollo proprietario, il *TecnoOut*. Inoltre, come accennato nel capitolo precedente, sfrutteremo il ricevitore AteArgo per inviare dei comandi anche alle periferiche Webu All-In-One.
Per quanto riguarda le funzionalità di questo software il suo scopo principale è quello di inviare comandi alle diverse periferiche, tuttavia per fornire maggiori informazioni all'operatore, ogni qualvolta che viene avviata la telegestione si richiedono alle periferiche ed alle centrali lo stato delle zone e delle partizioni in modo da avere sempre sotto controllo la situazione corrente.

5.1 I protocolli di comunicazione

Analizziamo ora come questo software comunicherà con le centrali, in particolare non ci addentreremo molto nei protocolli in quanto essi sono proprietari, ma analizzeremo la struttura del pacchetto e la comunicazione con la centrale o con il ricevitore interessato. In particolare analizzeremo i pacchetti di controllo del protocollo AteArgo per l'invio di comandi tra questo software e il corrispettivo ricevitore, e il protocollo *TecnoOut* per la comunicazione con le centrali Tecnoalarm.

5.1.1 Protocollo TecnoOut

Il protocollo TecnoOut è un protocollo proprietario di proprietà di TecnoAlarm studiato per la comunicazione tra le centrali e sistemi di gestione remota come software di domotica o, come nel nostro caso, sistemi di telegestione.

La struttura del pacchetto

Pur non potendo addentrarci in particolare nella struttura del pacchetto analizzeremo alcuni punti salienti. Questo protocollo è orientato al byte e più in particolare il paccetto ha una struttura molto semplice, esso è composto da:

$$\langle STX \rangle \langle codice \rangle \langle comando \rangle \langle len \rangle \langle dati \rangle \langle CRC16 \rangle$$

I diversi campi sono rispettivamente:

STX: campo composto da un unico byte e che delimita l'inizio del pacchetto;

codice: questo campo è composto da 3 byte e contiene il codice utente per permettere l'accesso alla centrale

comando: campo composta da un unico byte che identifica il comando da eseguire sulla centrale;

len: indica la lunghezza del campo dati, essa varia in base al tipo di operazione da eseguire sulla centrale;

dati: questo campo contiene le informazioni aggiuntive da utilizzare insieme al campo *comando*;

CRC16: questo è il campo di controllo errori che sfrutta un algoritmo di CRC a 16 bit calcolato sul resto del pacchetto. Questo campo ha una lunghezza di due byte.

La criptazione del pacchetto

Questo protocollo sfrutta la criptazione AES a 128 bit. Ogni attore della comunicazione deve conoscere una chiave denominata *PassPhrase* la quale verrà utilizzata insieme al vettore di inizializzazione.
Durante la prima fase il client invia un vettore di 17 byte contenete nei primi 16 byte il vettore di inizializzazione e nel diciassettesimo un valore predefinito criptato con il vettore appena inviato e con la PassPhrase. Il server quando riceve questo pacchetto salva i primi 16 byte come vettore e decripta il diciassettesimo con tale vettore e con la PassPhrase se il diciassettesimo byte decriptato corrisponde con il valore predefinito allora l'inizializzazione si conclude con successo e tutti i pacchetti successivi saranno criptati con tale vettore. In caso contrario il client chiude la connessione e riprova.

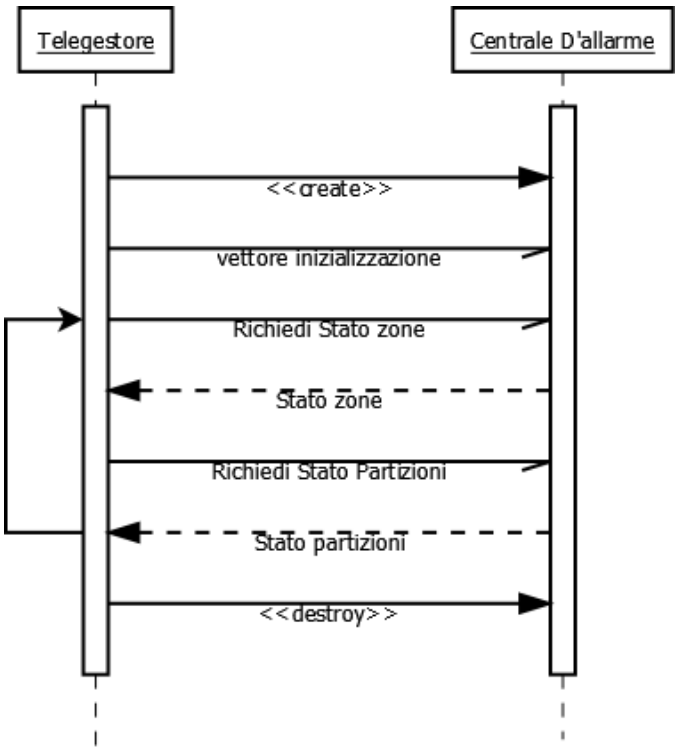


Figura 5.1: Scambio di messaggi tra software e centrali TecnoAlarm

Non ci soffermeremmo oltre sulla criptazione in quanto nel codice questa operazione sarà effettuata da una libreria esterna e non implementata direttamente.

La connessione

La connessione è una semplice connessione client server tramite protocollo TCP/IP nel quale il nostro software svolge la funzione di client e la centrale di allarme svolge quella di server. Questo significa che la centrale risponderà alle nostre richieste e non invierà messaggi se non interrogata. Durante la telegestione uno dei requisiti è quello di conoscere lo stato delle zone e delle partizioni perciò è necessario effettuare un polling per richiedere continuamente questi stati, ed all'inizio di ogni ciclo si controlla la presenza di nuovi comandi in coda. In Figura 5.1 vediamo come avviene la comunicazione, dopo la prima fase di inizializzazione del vettore di criptazione si prosegue con il ciclo di polling fino alla chiususra della connessione.

I tipi di pacchetto

In questo protocollo possiamo distinguere tre tipologie di pacchetto che sono:

- messaggi di monitoring
- messaggi di pilotaggio
- messaggi di risposta

I messaggi di monitoring servono per richiedere alla centrale di fornire informazioni riguardo al suo stato e a quello delle sue zone, i messaggi di pilotaggio sono quelli inviati dal nostro software verso la centrale per farle eseguire delle operazioni, infine, i messaggi di risposta sono quelli inviati dalla centrale per rispondere alle nostre richieste.

Messaggi di monitoring I messaggi di monitoring sono quei messaggi che il nostro software dovrà inviare alla centrale per richiedere lo stato di alcuni elementi, in particolare a noi interessa richiedere lo stato dei seguenti elementi:

- lo stato delle zone;
- lo stato delle partizioni;
- lo stato della centrale;
- lo stato dei programmatori orari.

Per stato delle zone noi intendiamo se il singolo sensore è escluso incluso oppure in allarme. Lo stato delle partizioni comporta sapere se esse sono inserite o disinserite o inserite solo in modo parziale. Per quanto riguarda le informazioni da conoscere sulla centrale, quelle che interessano a noi sono lo stato dell'alimentazione elettrica, quello della batteria e quello del tamper. Mentre l'unica informazione necessaria per i programmatori orari e se essi sono bloccati o in funzione.
Per reperire queste informazioni è necessario inviare alla centrale d'allarme il pacchetto formattato come precedentemente descritto con all'interno del campo codice un numero che ne identifica il comando. La centrale risponderà a questo comando inviando nel campo dati del pacchetto di risposta le informazioni. Prendendo come esempio le informazioni riguardanti la centrale noi inviamo il pacchetto specifico per richiedere queste informazioni e la centrale risponderà con un pacchetto con un unico byte nel campo dati. A questo byte deve essere applicata una maschera in AND per estrapolare le tre informazioni di cui necessitiamo.

Messaggi di pilotaggio I messaggi di pilotaggio servono per far eseguire delle azioni alla centrale d'allarme, per inviare i comandi da eseguire si utilizzano il pacchetto formattato così come indicato in precedenza con l'utilizzo di opportuni codici. A differenza dei comandi di monitoraggio il campo dati

questa volta contiene i valori da impostare sull'elemento selezionato. Ad esempio volendo escludo un opportuno sensore si invia alla centrale di allarme un pacchetto con l'opportuno valore nel campo comando e nei dati si inserisce il numero di zona e l'operazione da eseguire.
I comandi che interessano a noi sono solamente tre e riguardano le operazioni più comuni che gli operatori svolgono sulle centrali, ovvero, l'inclusione e l'esclusione di una zona, l'inserimento o il disinserimento di una partizione, ed infine il blocco o lo sblocco di un programmatore orario.

Messaggi di risposta I messaggi di risposta sono quelli che la centrale di allarme invia al nostro software essi possono essere di tre tipi:

- ACK: in questo caso nel campo dati sono presenti eventuali risposte al messaggio inviato dal software come lo stato degli elementi richiesti.
- NACK: questo messaggio si riceve quando la centrale d'allarme non è in grado di interpretare il messaggio appena ricevuto e quindi non è in grado di dare una risposta.
- BUSY: questo tipo di messaggio di risposta si ottiene quando la centrale di allarme non è in grado di soddisfare le richieste perchè occupata a svolgere altre operazioni o a soddisfare richieste precedenti.

L'ultimo messaggio è da tenere molto in considerazione in quanto esso limita il tempo di polling per gli stati della centrale, il protocollo infatti specifica che le richieste devono essere inviate con un intervallo minimo di 500ms.

5.1.2 Protocollo Urmet

Questo protocollo è lo stesso del capitolo precedente, in questo caso però analizzeremo il pacchetto nel caso di ricezione degli stati degli ingressi e delle uscite e i pacchetti per impostare dei valori per le uscite.

La struttura del pacchetto

La struttura è quella che abbiamo visto nel capitolo precedente, si tratta di un protocollo basato su stringhe che formano una struttura XML con un tag di apertura seguito da una parte di header nella quali sono contenuti ora e data della trasmissione del pacchetto. Nel corpo del pacchetto invece troviamo le informazioni vere e proprie che dipendono dal tipo di pacchetto.

La connessione

Come si è visto la connessione con il software AteArgo è una connessione di tipo locale client-server nel quale il software di Urmet ricopre il ruolo di server. Tuttavia, pur essendo esso un server non permette la connessione

di molteplici client questo ha comportato una problematica in quanto la connessione è necessaria al software ricevitore per mantenere la ricezione degli allarmi. Questo ci ha obbligati a pensare ad un meccanismo veloce e sicuro per far si che il ricevitore potesse inviare i comandi generati dal telegestore. Il meccanismo adottato è stato quello di una connessione socket tra Ricevitore e Telegestore, il telegestore invia una stringa ad un server che viene eseguito ogni volta che viene avviato il software di Ricezione. Questo meccanismo permette una comunicazione più rapida di quella che si avrebbe inserendo il comando in una tabella del database. Inoltre, questo tipo di comunicazione è stata pensata con un meccanismo di *Reques-Replay* ed è quindi piuttosto affidabile.
Come nel caso precedente si attua un meccanismo di polling per richiedere lo stato degli ingressi e delle uscite di una determinata centrale, tuttavia in questo caso il polling no può essere molto stringente in quanto il software AteArgo effettua ogni volta la connessione con la centrale e non la mantiene aperta.

I tipi di pacchetto

Come abbiamo detto i tipi di pacchetto necessari per permettere di implementare le funzionalità richieste sono tre:

- Pacchetto di richiesta degli stati
- Pacchetto di comunicazione degli stati
- Pacchetto di comando

Pacchetti di richiesta In questo tipo di pacchetto abbiamo che l'attributo del campo body è di tipo *INO* nel campo body è presente solo l'identificativo della periferica di cui si vogliono conoscere gli stati.

Pacchetti di comunicazione degli stati In questo tipo di pacchetto abbiamo che l'attributo del campo body è uguale a quello della richiesta degli stati. In questo campo dopo le normali informazioni per identificare la periferica si ha una serie di campi per ogni ingresso uscita che ne stabiliscono se è un contatto d'ingresso oppure un contatto d'uscita, se esso è impostato in modalità *"normalmente aperto"* oppure in modalità *"normalmente chiuso"* ed infine lo stato del contatto.

Pacchetto di comando In questo caso il pacchetto di comando contiene nella parte body le informazioni per identificare la centrale su cui effettuare i comandi e i conttati da attivare o disattivare.

5.1.3 Protocollo Telegestore-Ricevitore

Come abbiamo visto nella paragrafo precedente per permettere una comunicazione veloce ed efficente tra il telegestore e il ricevitore che invia i comandi si è deciso di instaurare una connessione socket di tipo request-reply tra i due software.
I due software si scambiano dei messaggi basati sullo stesso protocollo interno pensato per la comunicazione tra il server JBoss e il software di telegestione che vedremo nel capitolo successivo. Tuttavia qui accenniamo ad alcuni aspetti di questo pseudo-protocollo.
I pacchetti non sono altro che stringhe contenenti diversi campi separati da un carattere *";"*. I messaggi scambiati tra il telegestore e il software di ricezione hanno il seguente formato:

ce_id;COM;tipo;numero;comando

dove *ce_id* è il codice che identifica la centrale, *tipo* indica su quale elemento eseguire il comando se esso è una zona o una partizione. Il campo *numero* indica il numero dell'elemento sul quale eseguire e, infine, il campo *comando* contiene un codice per indicare quale azione sull'elemento identificato. Mentre il ricevitore, una volta eseguito il comando rimanda il pacchetto con indietro con l'aggiunta di un campo dopo comando con *OK* se il comando è andato a buon fine o con *KO* se il comando è fallito.

5.1.4 Altri protocolli

Mentre scriviamo sono in implementazione nuovi protocolli di telegestione, alcuni simili o comunque riconducibili a quelli già analizzati come il *CEI-ABI* protocollo standard per la ricezione di allarmi e la gestione di centrali d'allarme. La particolarità di questo protocollo è che mantiene sempre aperta la connessione con la centrale. Questa particolarità comporta un meccanismo tipo quello adottato con urmet per inviare i comandi in quanto è possibile aprire un'unica connessione ed è necessaria per la ricezione degli eventi.
Una seconda tipologia di telegestione in fase di sviluppo invece si basa sull'invio alle centrali di SMS preformattati e la centrale d'allarme risponde contattando la centrale operativa ed inviando le informazione richieste tramite i normali canali di comunicazione. Per questo tipo di comunicazione non è prevista la verifica dell'invio del comando e la comunicazione avviene tramite dei modem GPRS collegati su porte seriali.

5.2 La struttura dati

Bibliografia

[1] Dallmeier, cur. *PRemote-HD*. 2015. URL: http://www.dallmeier.com/fileadmin/user_upload/upload_dallmeier.com/PDFs/Downloads/Broschures/PRemote-Handout_en.pdf.

[2] ISO. *Information technology – Programming languages – C++*. ISO ISO/IEC JTC 1/SC 22. Geneva, Switzerland: International Organization for Standardization, 2011.

[3] pugixml, cur. *Pugixml*. 2014. URL: http://pugixml.org.

[4] LIS s.r.l., cur. *LIS, Chi Siamo*. 2015. URL: http://www.lis-srl.it/chi-siamo.asp.

[5] Wikipedia, cur. *Sicurezza*. 2015. URL: http://it.wikipedia.org/wiki/Sicurezza.

[6] Wikipedia, cur. *Vigilanza Privata*. 2015. URL: http://it.wikipedia.org/wiki/Vigilanza_privata.

Appendice A

Documentazione del progetto logico

Documentazione del progetto logico dove si documenta il progetto logico del sistema e se è il caso si mostra la progettazione in grande del SW e dell'HW. Quest'appendice mostra l'architettura logica implementativa (nella Sezione 4 c'era la descrizione, qui ci vanno gli schemi a blocchi e i diagrammi).