Università degli Studi di Ferrara

DOTTORATO DI RICERCA IN
SCIENZE DELL' INGEGNERIA

CICLO XXIV

COORDINATORE Prof. Stefano Trillo

ARCHITETTURE, SERVIZI E MULTIMEDIALITÀ NEI NUOVI MEDIA

Settore Scientifico Disciplinare 09/F2

Dottorando
Dott. Benetti Elisa

Tutore
Prof. Mazzini Gianluca

Anni 2009/2011

Ringraziamenti

Questa tesi rappresenta per me la conclusione di tre intensi anni di lavoro e ricerca, che mi hanno permesso di assaporare la vita che da sempre ho desiderato: un susseguirsi di sfide che hanno incessantemente alimentato la mia curiosità rendendo gli obiettivi da raggiungere uguali ad una corsa, dove la soddisfazione data dall' arrivo fa dimenticare le fatiche affrontate e porta subito la mente a cercare un nuovo traguardo da tagliare.

In primo luogo vorrei ringraziare il mio tutore, il Prof. Gianluca Mazzini, che mi ha dato la possibilità di intraprendere questo viaggio, contagiandomi con il suo grande entusiasmo verso la ricerca ed indicandomi sempre nuove interessanti strade da poter esplorare. Non solo in questi anni è stato il primo ad essere partecipe dei miei successi ma ha sempre trovato il tempo di aiutarmi ed incoraggiarmi nei momenti critici, con la sua innata capacità di risolvere le mie tipiche complesse ed articolate paranoie interiori, con poche e chiare parole giuste dette nel momento giusto.

Ringrazio inoltre tutta l' azienda Lepida SpA, che ha reso possibile concretizzare sul campo i miei studi dandomi così le più grandi soddisfazioni che potessi desiderare, permettendomi di collaborare con persone eccezionali sia dal punto di vista tecnico che da quello umano.

Un grandissimo grazie va anche all' amico e collega Fabio, che dopo aver condiviso per anni i travagliati momenti di studio all' Università, ha continuato ad essere al mio fianco anche in queste ricerche, dandomi un preziosissimo aiuto per la sopravvivenza nell' insidioso mondo dell' audio/video.

Grazie alle colleghe Chiara ed Enrica per essere state per me un grande esempio nel mondo della ricerca, ma anche per avermi costantemente supportata, e soprattutto sopportata, lungo questo percorso.

Un grazie di cuore ad Emanuele, che per anni anche se lontano ha sempre condiviso ogni attimo della mia vita, aprendomi gli occhi quando mi rifiutavo di guardare ciò che sono, ridandomi energia e fiducia. Adesso che è al mio fianco, giorno dopo giorno non smette mai di ricordarmi e dimostrarmi che tutto è possibile, quando lo si desidera davvero.

Un ultimo ringraziamento speciale è invece per mia madre che uscita dal suo forte ruolo di genitore mi sta permettendo finalmente di conoscerla dav-

vero, apprezzando nuove imprevedibili sfaccettature del suo poliedrico modo di essere, ed infine per Sara, che al contrario mi sembra di conoscere da sempre ed è un punto fermo non solo per il suo affetto ma per la sua schietta sincerità che mi ha sempre fatta scontrare faccia a faccia con la realtà e con me stessa, riportandomi sulla giusta strada. Entrambe hanno creduto in me e nelle mie capacità fin dall' inizio e questo mi ha dato la forza di arrivare fino al momento in cui anche io, finalmente, sono riuscita a crederci.

Indice

Prefazione **xi**

Introduzione **xiii**

1 Infrastrutture wireless **1**

1.1 Flusso di Trasporto e standard DVB 1

1.1.1 Ottimizzazione delle bitrates 3

1.2 Standard MHP 5

1.3 Teletext 7

1.4 OpenCaster 8

2 Infrastrutture wired **11**

2.1 Architetture NGAN 11

2.2 Multicast 15

2.2.1 Algoritmi di Routing 15

2.2.2 Protocolli multicast affidabili 18

2.2.3 Schematizzazione dei protocolli tramite AUML 24

3 Servizi per il Digitale Terrestre **33**

3.1 Struttura di un servizio MHP 33

3.2 Struttura di un servizio Teletext 36

3.3 Servizi sviluppati 37

3.3.1 Launcher e menu iniziale 38

3.3.2 Informativo 39

3.3.3 Visita guidata 40

3.3.4 Accesso condizionato 41
3.3.5 Autovalutazione 42
3.3.5.1 Interfaccia Web 44
3.3.5.2 Centro servizi 45
3.3.5.3 Applicazione MHP 49
3.4 Analisi delle prestazioni 50
3.5 Riadattamento dei servizi per il web 56

4 Monitoraggio ambientale 59
4.1 Prototipo sviluppato internamente 61
4.1.1 Monitoraggio acustico 62
4.1.2 Monitoraggio del traffico 63
4.1.3 Monitoraggio frane 65
4.2 Evoluzione tramite LabICT 66
4.3 Integrazione con il sistema multicanale 69

5 Monitoraggio rete Lepida 73
5.1 Architettura 74
5.2 Interfaccia Web 76
5.3 Caching dei grafici 78
5.4 Integrazione con il sistema multicanale 79

6 Portale regionale OpenData 81
6.1 Architettura 82
6.2 Integrazione con il sistema multicanale 85

7 Contenuti audio/video e multicanalità 87
7.1 Digitale terrestre 87
7.2 Web 90
7.3 IPTV 93

8 OpenBOXware 101
8.1 Architettura 101
8.2 Integrazione servizi ed audio/video 103

9 Conclusioni e sviluppi futuri **107**

Bibliografia **111**

Pubblicazioni **117**

Elenco delle figure

1 Mappa della copertura della rete Lepida xiv
2 Famiglie per beni e servizi tecnologici disponibili xv
3 Utilizzo di televisione e internet per fasce d' età xvi

1.1 Schematizzazione di un Transport Stream 2
1.2 Ciclo di vita di una Xlet 6
1.3 Vendite decoder in Italia da Settembre 2010 a Settembre 2011 7

2.1 Architetture NGAN 12
2.2 Schema di principio di un AWG 14
2.3 Architetture WDM 14
2.4 Algoritmi di Routing Multicast 16
2.5 Protocolli affidabili Multicast 18
2.6 Schematizzazione dei protocolli multicast TMTP ed RMTP . 27
2.7 Schematizzazione dei protocolli multicast RMTP-II e SRM . 28
2.8 Schematizzazione dei protocolli multicast LMS ed MDP . . . 29
2.9 Schematizzazione del protocollo multicast RMP 31
2.10 Schematizzazione dei protocolli multicast AG ed RDG 31

3.1 Prestazioni nella navigazione usando file sequenziali e binari . 34
3.2 Catena relativa a un servizio MHP 35
3.3 Distribuzione delle pagine in un loop di Teletext 36
3.4 Catena relativa a un servizio Teletext 37
3.5 Launcher, spot e menu su LepidaTV 38
3.6 Esempio di servizio informativo su LepidaTV 39
3.7 Esempio di visita guidata su LepidaTV 41

3.8 Esempio di servizio con accesso condizionato su LepidaTV . . 42
3.9 Sequence diagram del servizio di autovalutazione 43
3.10 Tabelle del database del servizio di autovalutazione 44
3.11 Struttura dei file binari del servizio di autovalutazione 46
3.12 Decifratura (a) e cifratura (B) del codice 48
3.13 Esempio di Servizio di Autovalutazione 50
3.14 Prestazioni MHP nel caso medio e peggiore 53
3.15 Prestazioni MHP nel caso migliore 54
3.16 Prestazioni Teletext nel caso medio e peggiore 54
3.17 Prestazioni Teletext nel caso migliore 55
3.18 Confronto MHP e Teletext nel caso medio e nel caso migliore 55
3.19 Catena relativa a un servizio riadattato per il Web 56
3.20 Riadattamento dei servizi sul web 57

4.1 Architettura del Centro Gestione Dati 60
4.2 Modelli di integrazione delle reti di sensori 62
4.3 Interfaccia web per il monitoraggio acustico 63
4.4 Interfaccia web per il monitoraggio del traffico 65
4.5 Sito di installazione per il monitoraggio frane 66
4.6 Architettura del CGD proposto dai LabICT-PA 68
4.7 Interfaccia Web del CGD LabICT-PA 70
4.8 File binari di un servizio MHP collegato alle Reti di Sensori . 71

5.1 Architettura del Monitoraggio Lepida 74
5.2 Grafici relativi a un' interfaccia di rete e a un sevizio 76
5.3 Funzionalità dell' interfaccia web del monitoraggio Lepida . . 77
5.4 Flusso logico del sistema di monitoraggio Lepida 79

6.1 Homepage del portale Open Data della Regione Emilia Romagna 82
6.2 Architettura del portale messo al riuso da Regione Piemonte 83
6.3 Esempio degli Open Data riguardanti gli edifici della RER . . 84

7.1 Catena dello streaming broadcast di LepidaTV 90

7.2 Catena dello streaming simulcast di LepidaTV 91
7.3 Simulcast di LepidaTV attraverso il suo sito web 92
7.4 Esempio di ricerca e visione di un video OnDemand di LepidaTV 93
7.5 Esempio di sistema multicast per IPTV 95
7.6 Architettura VLAN per subscriber 96
7.7 Architettura VLAN per service 97
7.8 Architetture switch ottici 98
7.9 Classificazione delle protezioni delle sessioni multicast 99
7.10 Confronto euristiche Self-sharing light-trees 99
7.11 Comparazione dei tre approcci di protezione delle sessioni multicast 100

8.1 Architettura della piattaforma OpenBOXware 102
8.2 Widget di LepidaTV sviluppato per OpenBOXware 104

Prefazione

La tecnologia digitale negli ultimi anni sta prendendo sempre più velocemente il sopravvento su quella analogica ed in questo contesto la televisione digitale è senz' altro una potenziale rivoluzione del XXI secolo, considerando la recente diffusione di schermi LCD e al plasma e la conseguente affermazione di standard televisivi 576p, 720p e 1080p. Nasce inoltre una rinnovata possibilità di veicolare informazioni e servizi attraverso le ampliate potenzialità che il passaggio al digitale offre a questo mezzo di comunicazione.

Partendo da un' accurata analisi delle architetture e dall' ideazione di servizi a valore aggiunto per la televisione digitale terrestre, la conseguenza logica è stata l' estensione delle ricerche ad un più ampio concetto di *multicanalità* il cui scopo principale risulta essere quello di analizzare le opportunità delle diverse infrastrutture esistenti ed attualmente in sviluppo, i loro vincoli tecnologici, gli standard già affermati e quelli emergenti, al fine di fornire gli stessi servizi attraverso ognuna di esse tramite opportuni meccanismi, il più possibile automatizzati, di adattamento al mezzo trasmissivo, ottimizzando l' ergonomicità e la portabilità delle soluzioni proposte per le differenti piattaforme.

A seguito di un' indagine sullo stato attuale del territorio italiano rispetto alle problematiche di *digital* e *knowledge divide* e dopo aver illustrato l' ambiente di sperimentazione tramite cui è stato possibile effettuare test implementativi sul campo, questa tesi sarà strutturata nel seguente modo.

Il Capitolo 1 tratterà nello specifico il broadcasting nell' ambito della televisione digitale terrestre, delineando la struttura del flusso trasmissivo, per poi analizzarne il nuovo standard Multimedia Home Platform (MHP) e il vecchio standard Teletext.

Il Capitolo 2 si occuperà invece di un' indagine riguardante le Next Generation Networks (NGN)e le architetture possibili. Inoltre verrà effettuato uno studio degli algoritmi di generazione di alberi di distribuzione e dei protocolli multicast affidabili esistenti per un loro successivo inquadramento nell' ambito delle NGN. Un metodo per la schematizzazione uniforme dei protocolli è infine proposto.

Con il Capitolo 3 si entra invece nell' ambito dei servizi da distribuire at-

traverso le infrastrutture appena analizzate. Verrà spiegata innanzitutto la struttura di base di un servizio implementato secondo gli standard MHP e Teletext, comparandone le prestazioni, e verranno illustrate nello specifico le differenti tipologie di applicazioni sviluppate. Queste verranno infine riproposte anche sul web, spiegandone il loro riadattamento.

Vengono quindi mostrati alcuni progetti paralleli realizzati al fine di dotarsi di un' ampia raccolta di informazioni eterogenee da erogare attraverso il sistema multicanale e la loro integrazione con il sistema stesso. Il Capitolo 4 è dedicato ad un prototipo di monitoraggio ambientale, visto come centro gestione dati in grado di raccogliere, validare, storicizzare, analizzare e correlare misurazioni provenienti da sensori eterogenei dislocati sul territorio regionale. Nel Capitolo 5, invece, verrà illustrato un servizio di monitoraggio congiunto di reti e servizi che importa, uniforma, aggrega i dati riguardanti il traffico di rete e la frequenza di utilizzo dei servizi erogati sulla rete stessa. Infine il Capitolo 6 tratterà la realizzazione del portale regionale degli OpenData che è risultato un punto di convergenza non solo per informazioni provenienti dagli enti del territorio, ma anche dei dati forniti attraverso i progetti precedenti.

Il concetto di multicanalità è stato esteso anche alla parte audio/video ed il Capitolo 7 analizzerà per prima l' ottimizzazione del flusso di broadcasting, per passare in seguito al simulcast su web e terminare con alcune ricerche sull' IpTV visto come uno dei principali servizi che potranno essere offerti grazie alle NGN.

Il Capitolo 8 è interamente dedicato ad una piattaforma emersa recentemente grazie a ricerche dell' Università di Urbino, chiamata OpenBOXware, e di come siano stati facilmente riadattati uno dei servizi proposti ed il flusso audio/video per poter essere integrati anche in questo strumento.

La tesi terminerà quindi con il Capitolo 9 che riporterà le conclusioni riguardanti il lavoro e le ricerche svolte ed i possibili sviluppi futuri.

Introduzione

Nel corso degli ultimi anni l' evoluzione della tecnologia è sempre più veloce e le avanzate tecnologie di informazione e comunicazione (ICT) possono svolgere un ruolo chiave anche nel facilitare l'accesso alle informazioni pubbliche migliorando la qualità, velocità e affidabilità dei servizi erogati ai cittadini. In questo contesto con il concetto di *e-government* (governo elettronico) si intende anche l' avvicinare la pubblica amministrazione ai cittadini attraverso l' utilizzo di supporti elettronici per la diffusione di informazioni e servizi.

La diffusione delle ICT è stato un processo tanto veloce quanto disuniforme nelle diverse aree del mondo, producendo un problema ben noto chiamato *digital divide*. Questo termine si riferisce al divario tra le persone con un accesso effettivo al digitale e alle nuove tecnologie di informazione, e gli altri cittadini che hanno invece a disposizione un accesso molto limitato o addirittura nullo. Queste differenti condizioni influiscono sulla qualità della vita e sulle nuove opportunità sia di business che culturali: al giorno d'oggi essere disconnessi dalla Rete significa essere relegati ai margini della società. Possiamo inoltre trovare diversi livelli di digital divide: anche quando una connessione è presente infatti, ci possono essere interruzioni nell' accesso alla rete a banda larga che rendono difficile ottenere una quantità ingente di informazioni.

Nel corso del 2002, la Regione Emilia Romagna ha avviato il progetto Lepida: la rete privata delle Pubbliche Amministrazioni regionali, che le collega integrando le reti locali già presenti con nuove connessioni che saranno fornite e che si amplieranno in futuro.

Lepida, infatti, è una rete omogenea, perché è stata progettata e costruita seguendo i paradigmi della NGN (Next Generation Networks), armonizzandoli al fine di fornire servizi ad alto contenuto tecnologico; in questo modo Lepida è in grado di integrare tutte le risorse disponibili e di fornire servizi correlati con le richieste e le esigenze dei singoli utenti. Nel 2008 la società Lepida SpA [4], diventa lo strumento operativo promosso dalla Regione Emilia-Romagna (RER) per la progettazione, realizzazione e gestione di reti di telecomunicazione e per fornire servizi di informazione ai propri partner e alle aziende collegate alla rete Lepida.

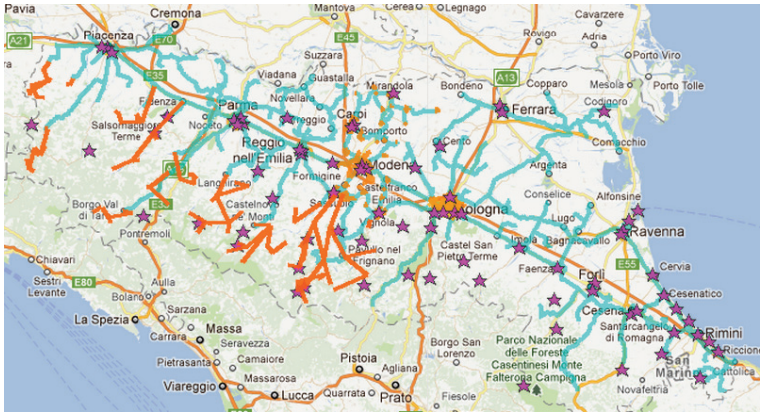


Figura 1: Mappa della copertura della rete Lepida

Al fine di garantire la copertura su tutto il territorio regionale, Lepida SpA gestisce sia la dorsale geografica sia i collegamenti che, partendo dalla dorsale, collegano tutte le istituzioni e gli enti. Questo viene fatto attraverso l' utilizzo di tre tecnologie, mostrate in Figura 1: la fibra ottica che, grazie alla sua lunghezza di oltre 2700 km, rappresenta l'infrastruttura principale della regione (tratte arancioni nella Figura 1),in secondo luogo tramite tecnologie HDSL o via satellite nelle zone non ancora raggiungibili dalla fibra. Inoltre è prevista una parte Lepida Wireless (tratte azzurre nella Figura 1) con l'obiettivo di estendere la rete ai Comuni in situazioni particolarmente critiche di digital divide, realizzando delle dorsali con tecnologie radio per il trasporto della banda. Questo permette di assicurare pari opportunità tecnologiche e di sviluppo economico e sociale anche in zone in cui vi è una mancanza di infrastrutture disponibili, come ad esempio nel territorio appenninico ed in alcune zone della bassa pianura. Diverse MAN (Metropolitan Area Networks), inoltre, sono state aggiunte alla dorsale, e consentono la connessione locale di diversi uffici della pubblica amministrazione situati nella stessa area, creando una grande rete di distribuzione locale (aree arancioni nella Figura 1). Infine Lepida SpA realizza e gestisce anche la Rete Radiomobile Regionale (ERretre) sul territorio dell'Emilia-Romagna, dedicata ad interventi di emergenza, opportunamente collegata alla Rete in fibra ottica (stazioni radio-base rappresentate dalle stelle nella Figura 1). Si tratta di una rete cellulare digitale, basata su standard europeo TETRA [5], che fornisce servizi di comunicazione voce e dati sull' intero territorio della regione e, dato il suo scopo primario, risulta fondamentale il grado di sicurezza che ERretre assicura tramite la cifratura delle chiamate (a protezione delle comunicazioni da possibili intercettazioni), l'autenticazione dei terminali (a protezione dall'accesso di apparati non autorizzati), la ridondanza dei collegamenti e dei relativi apparati, la funzionalità avanzata di disaster recovery e l'operatività dei terminali in modalità locale (fall-back).

In aggiunta alla gestione della rete e del suo sviluppo, Lepida SpA fornisce servizi di supporto tecnologico rispettando le esigenze telematiche degli enti. L'azienda mette anche a disposizione una piattaforma uniforme utilizzata per lo sviluppo di nuovi servizi innovativi e la loro integrazione attraverso la rete Lepida. Questo viene fatto prima valutando la fattibilità dei servizi, quindi giocando il ruolo di project manager durante la loro realizzazione.

Una collaborazione costante con l' azienda Lepida SpA durante questi tre anni di ricerche ha permesso quindi non solo di avere a disposizione una rete territorialmente diffusa e tecnologicamente eterogenea, ma di avere anche l' opportunità di sviluppare e testare differenti tipologie di servizi erogati alla pubblica amministrazione, e conseguentemente ai cittadini, tramite la rete stessa.

La Regione Emilia Romagna è una punta d' eccellenza riguardo il digital divide: può infatti vantare una copertura a banda larga, wired e wireless, stimata del territtorio del 95,20%, ma ben diversa è la generale situazione italiana, dove gli ultimi dati ISTAT del mese di Dicembre 2011 [6] rendono noto che le famiglie del Centro-nord che dispongono di un accesso a Internet sono oltre il 56%, mentre circa il 49% dispone di una connessione a banda larga. Le famiglie del Sud Italia invece, rispettivamente dispongono al 48,6% di accesso alla Rete e solo il 37,5% di esse ha la banda larga.

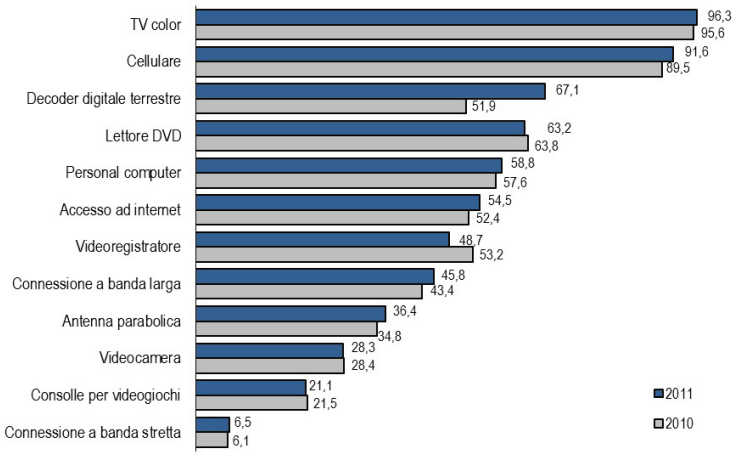


Figura 2: Famiglie per beni e servizi tecnologici disponibili

Il *knowledge divide* è un altro concetto importante che descrive invece il divario nelle condizioni di vita tra coloro che sono in grado di trovare, gestire ed elaborare informazioni o conoscenze, e coloro che invece non riescono, per molte ragioni differenti. La conoscenza specializzata è diventata una componente sempre più in crescita nella società, e la diffusione di questa conoscenza diventa ancor più veloce con la moderne tecnologie. L' ISTAT infatti rileva che ad oggi il 41,7% delle famiglie dichiara di non possedere

l' accesso a Internet perchè non ha le competenze per utilizzarlo; il 26,7% considera Internet inutile e non interessante, il 12,7% non ha accesso a Internet da casa perchè accede da un altro luogo, l' 8,5% perchè considera costosi gli strumenti necessari per connettersi e il 9,2% perchè ritiene eccessivo il costo del collegamento. Per quanto sia in aumento la presenza in percentuale di beni e servizi tecnologici nelle famiglie italiane, come mostrato dalla Figura 2, rimane un notevole divario tecnologico, da ricondurre a fattori di tipo generazionale, culturale ed economico. Ad esempio, le famiglie costituite esclusivamente da persone di 65 anni e più si confermano quelle meno provviste di beni e servizi tecnologici: appena l' 11,3% di esse possiede il personal computer e soltanto il 9,4% dispone di una connessione per navigare in Internet.

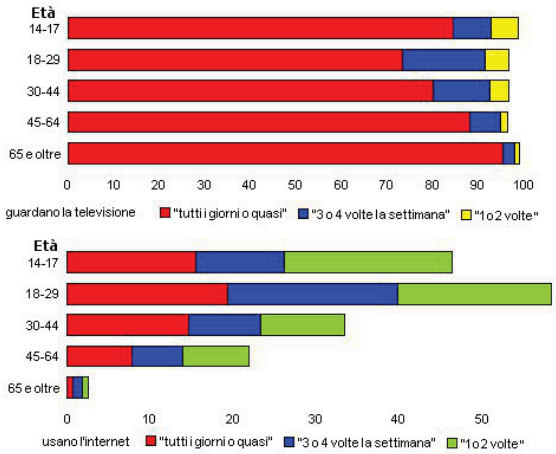


Figura 3: Utilizzo di televisione e internet per fasce d' età

Soffermandosi invece sull' utilizzo delle risorse e non solo sulla disponibilità, in Figura 3 è mostrato un confronto, svolto dal Censis [7], delle percentuali di utilizzo della televisione e di internet nella popolazione italiana. In generale la televisione è il mezzo di comunicazione più conosciuto: in particolare, i cittadini con più di 65 anni sono il gruppo che sia utilizza meno la connessione ad internet, sia guarda di più la televisione.

LepidaTV è uno dei servizi gestiti e sviluppati da LepidaSpA e nasce alla luce delle problematiche appena illustrate: il suo scopo quindi non è quello di essere un' *altra televisione* ma essere uno strumento che permette di:

- superare digital divide: come si evince dalle statistiche in Italia l' utilizzo della banda larga si limita ad una media del 43% delle famiglie del territorio. I cittadini con almeno una televisione rappresentano invece attualmente oltre il 95% della popolazione e l'utilizzo di questo

supporto permette di fornire informazioni già esistenti su Internet, ad un target più ampio;

- superare knowledge divide: anche se è disponibile una connettività a banda larga ci potrebbe essere una mancanza di consapevolezza delle sue potenzialità e la mancanza di capacità di un suo corretto utilizzo. I cittadini sono meno diffidenti verso la televisione, che resta la tecnologia più familiare con un meccanismo di funzionamento che viene ritenuto più semplice.

Nello specifico LepidaTV [8] è un canale in tecnologia digitale terrestre a disposizione di tutti gli enti della Regione Emilia Romagna per aumentare l'erogazione dei propri servizi che si serve di un insieme di broadcaster, che coprono ad oggi il 97% del territorio regionale, e di un editor terzo rispetto alle Pubbliche Amministrazioni, in modo da realizzare un sistema pienamente compatibile con il complesso panorama legislativo del settore televisivo.

Grazie allo switch-off dal segnale analogico al digitale terrestre avvenuto nella regione Emilia Romagna nel mese di Novembre dell' anno 2010, come stabilito dal calendario nazionale [9], è stato quindi possibile effettuare una sperimentazione reale dei servizi implementati attraverso la loro messa in onda su LepidaTV, raggiungendo quasi tutti i cittadini del territorio regionale nell'anno 2011.

Capitolo 1

Infrastrutture wireless

Come anticipato, le infrastrutture trasmissive *via radio* che verranno analizzate nello specifico in questa tesi, riguardano il sistema di broadcasting per la Televisione Digitale Terrestre (Digital Terrestrial Television, DTT). Partendo da una breve introduzione che illustri la struttura del *transport stream* ci soffemeremo sugli standard MHP e Teletext utilizzati per l' implementazione dei servizi presentati nel Capitolo 3.

1.1 Flusso di Trasporto e standard DVB

Il segnale digitale televisivo [10] è trasmesso come uno stream, ovvero un flusso, di dati MPEG-2 la cui bit rate ammessa varia da circa 5 a circa 32 Mbps, a seconda dei parametri scelti per la modulazione: costellazione, tasso di codifica del codice correttore interno e dalla durata dell' intervallo di guardia. Ad esempio, per la stima del massimo numero di programmi TV che possono essere allocati in un canale a 8 MHz, la configurazione più idonea è il 64-QAM a tasso 2/3 che fornisce una capacità di flusso binario di circa 24 Mbit/s,consentendo tipicamente la trasmissione di 4 programmi a qualità convenzionale (SDTV a 6 Mbit/s ciascuno) o 6 programmi a qualità news (LDTV, 4 Mbit/s ciascuno).

Lo stream, inoltre è composto da un insieme di sotto-stream, chiamati *elementary stream*, ognuno dei quali può contenere o una traccia audio, o una traccia video o dati sempre incapsulati in uno stream MPEG-2. Ogni stream elementare ha un PID, come identificativo di pacchetto, che lo identifica univocamente all' interno del transport stream. La generazione del flusso di trasporto finale per il broadcasting avviene multiplexando più flussi elementari che sono tipicamente un flusso audio, uno video e eventuali flussi dati, tutti suddivisi in transport packet, dove ogni pacchetto è lungo 188 byte: il multiplexer prende quindi questi pacchetti e li inserisce nel transport stream. Per riconoscere e ricostruire questi elementary stream, vengono aggiunte in

fase di multiplexing altre informazioni dette di *servizio*. Le più importanti da segnalare sono:

- **PAT** (*Program Association Table*): è una tabella fondamentale, l' unica che viene trasmessa con PID fisso e descrive la quantità dei servizi contenuti nel transport stream indicando per ognuno di essi un riferimento alla tabella PMT, spiegata di seguito;
- **PMT** (*Program Map Table*): contiene l'elenco degli elementi che compongono un servizio. In base alle informazioni in essa contenute, il decoder individua in quali pacchetti del transport stream sono contenuti il video, l'audio e gli elementi aggiuntivi da mostrare sullo schermo. Esiste un PMT per ogni servizio contenuto nel transport stream;
- **AIT** (*Application Information Table*): Tabella extra, definita da MHP segnala la presenza di un'applicazione interattiva all'interno del transport stream (TS), e contiene le informazioni necessarie al decoder per eseguire l'applicazione ed avvisare l'utente della sua disponibilità. Avrà quindi una voce per ogni applicazione presente per quel servizio.

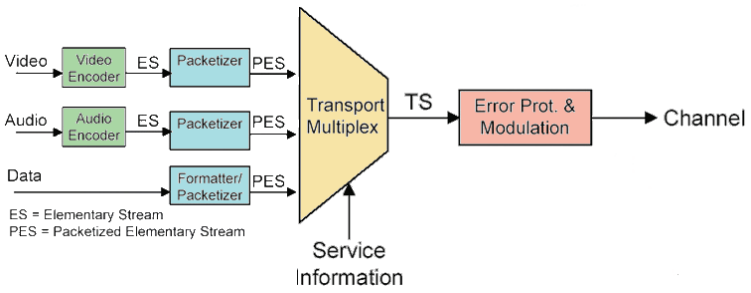


Figura 1.1: Schematizzazione di un Transport Stream

La struttura appena spiegata si può riassumere con lo schema di Figura 1.1 dove si può notare che uno stream di trasporto è un flusso MPEG-2 che contiene numerosi programmi (TS), ognuno dei quali legato ad un singolo canale televisivo, formati a loro volta da più flussi elementari che possono essere dati, audio e video codificati in MPEG-2.

Una volta creato lo stream finale, questo dev' essere trasportato fino al ricevitore dell' utente ed il Digital Video Broadcasting - Terrestrial (DVB-T) è lo standard del consorzio europeo DVB per una modalità di trasmissione televisiva digitale terrestre. Il sistema è basato sull' adozione degli standard MPEG-2 per la codifica del segnale audio/video di sorgente e per la multiplazione: è stato sviluppato per la trasmissione di segnali televisivi multi-programma a definizione convenzionale nel formato MPEG-2, ma, nonostante la definizione della specifica DVB-T risalga al novembre 1995 con

approvazione come standard ETSI nel febbraio 1997, il processo di codifica era già stato aperto all' evoluzione verso l' alta definizione (HDTV) mediante l' uso di livelli e profili MPEG-2 più elevati. Altri sistemi di trasmissione video digitale della famiglia DVB sono il DVB-S per le trasmissioni satellitari, il DVB-C per le trasmissioni via cavo, e il DVB-H, per le trasmissioni digitali terrestri rivolte ai cellulari di nuova generazione, ma ovviamente rimane il DVB-T quello utilizzato per questa tesi.

1.1.1 Ottimizzazione delle bitrates

Essendo possibile, tramite il software utilizzato durante le sperimentazioni ed illustrato al paragrafo 1.4, al momento del multiplexing specificare la bitrate da assegnare ad ogni specifico servizio, è emersa la necessità di un' ottimizzazione di questi valori così da minimizzare il tempo medio di attesa per il caricamento dei servizi. L' ottimizzazione delle bitrate da assegnare ai vari servizi è stata effettiata tramite un modello di programmazione lineare con:

- Parametri noti:
  - Numero dei servizi: ad oggi 8
  - Dimensione media dei servizi: ad oggi valori impostati *[1600000, 440000, 560000, 480000, 2400000, 320000, 400000, 1120000]* Relativi ai servizi in quest' ordine: *['Cartellone', 'ProvinciaFerrara', 'Civetta', 'RifugioCaneGatto', 'ComuneArgenta', 'Launcher', 'MenuApps', 'Forli']*
  - Bitrate totale disponibili per i servizi: ad oggi 1783284
- Vincoli lineari di base, dove BX_i Bitrate da assegnare, SX_i dimensione dei servizi:
  - Ogni bitrate deve avere un valore positivo > 0

$$BX_i \geq 1 \qquad \forall i = 1..n \tag{1.1}$$

  - La somma delle bitrate deve essere uguale alla bitrate totale disponibile per servizi

$$\sum_{i=1}^{n} BX_i = B_{tot} \tag{1.2}$$

- Funzione obiettivo: minimizzare la media dei secondi necessari allo scaricamento di un servizio

$$\frac{1}{n} \sum_{i=1}^{n} \frac{BX_i}{SX_i} \tag{1.3}$$

E' stato quindi utilizzato il software Xpress-Mosel, nella sua versione free, per la risoluzione del modello LP. I vincoli di cui sopra sono stati leggermente modificati per l' impossibilità di dividere un intero per una variabile Mosel e quindi è stata massimizzata la frazione inversa:

```
!vincoli
forall (i in BITRATES) BX(i) is_integer
!la bitrate deve essere un valore intero
forall (i in BITRATES) BX(i)>=1
!le bitrate devono esser tutte maggiori di zero
sum(i in BITRATES) BX(i)=BTOT
!la somma delle bitrate deve essere uguale alla bitrate
!disponibile per i servizi
SOL = sum(i in BITRATES)(BX(i)/SX(i))
!sommatoria degli inversi dei secondi
AVGSOL = SOL/N !media della sommatoria
!funzione obiettivo
minimize(AVGSOL)!essendo l' inverso dei secondi la massimizzo
```

Lasciando solo i vincoli base il risultato ottenuto è il seguente:

Cartellone : 1 bps, si caricherà in 1.6e+006 secondi
ProvinciaFerrara : 1 bps, si caricherà in 440000 secondi
Civetta : 1 bps, si caricherà in 560000 secondi
RifugioCaneGatto : 1 bps, si caricherà in 480000 secondi
ComuneArgenta : 1.78328e+006 bps, si caricherà in 1.34584 secondi
Launcher : 1 bps, si caricherà in 320000 secondi
MenuApps : 1 bps, si caricherà in 400000 secondi
Forli : 1 bps, si caricherà in 1.12e+006 secondi

Questo perchè il modello LP punta a rendere velocissimo un servizio (partendo dal piu corposo, Argenta) a discapito di tutti gli altri. E' stato quindi aggiunto un ulteriore vincolo:

```
forall (i in BITRATES) (BX(i)/SX(i))>=(1/SEC)
!do un massimo di secondi di attesa
```

Dove appunto con SEC vengono indicati quanti secondi massimo un servizio può impiegare per essere scaricato e aperto. E' interessante notare che si puo così facilmente valutare quel' è il SEC minimo che permette di avere soluzioni, e nel nostro caso e circa 4.2 secondi

Soluzione minimizzata : 4.16317 secondi in media
Cartellone : 380953 bps, si caricherà in 4.19999 secondi
ProvinciaFerrara : 104762 bps, si caricherà in 4.2 secondi
Civetta : 133334 bps, si caricherà in 4.19998 secondi
RifugioCaneGatto : 114286 bps, si caricherà in 4.19999 secondi

ComuneArgenta : 611852 bps, si caricherà in 3.92252 secondi
Launcher : 76191 bps, si caricherà in 4.19997 secondi
MenuApps : 95239 bps, si caricherà in 4.19996 secondi
Forli : 266667 bps, si caricherà in 4.19999 secondi

Partendo da questi vincoli necessari, è possibile fare moltissime altre prove a seconda delle esigenze, ad esempio aggiungendo il vincolo che il Launcher (applicazione che mostra la scritta *Premi il pulsante rosso*) non ci impieghi piu di 2 secondi per caricarsi essendo sempre mostrato e quindi risultando più importante degli altri:

```
BX(6)/SX(6) >= (1/2) !il Launcher non voglio
!che impieghi più di 2 secondi a partire
```

Il minimo SEC diventa ora 4.5 secondi ed il risultato è:

Soluzione minimizzata : 3.83918 secondi in media
Cartellone : 355556 bps, si caricherà in 4.49999 secondi
ProvinciaFerrara : 97778 bps, si caricherà in 4.49999 secondi
Civetta : 124445 bps, si caricherà in 4.49998 secondi
RifugioCaneGatto : 106667 bps, si caricherà in 4.49999 secondi
ComuneArgenta : 601060 bps, si caricherà in 3.99295 secondi
Launcher : 160000 bps, si caricherà in 2 secondi
MenuApps : 88889 bps, si caricherà in 4.49999 secondi
Forli : 248889 bps, si caricherà in 4.5 secondi

Il numero di vincoli può ulteriormente crescere, ad esempio specificando che un servizio debba impiegare meno tempo di un altro e così via. Questo strumento di ottimizzazione verrà ripreso per effettuare un' analisi prestazionale dei servizi al paragrafo 3.4.

1.2 Standard MHP

MHP è uno standard molto giovane [11], definito nell' anno 2000 dove, durante la creazione del pacchetto MHP, il DVB Project ha scelto di utilizzare Java per il core del middleware così da permettere ai fornitori di servizi di poter sviluppare i propri servizi in modo indipendente dal sistema di ricezione sottostante. Lo standard di base è definito dalla specifica MHP 1.0 che contiene, principalmente:

- l' architettura di base dell' MHP [12], strutturata su tre livelli: *Resources* dedicate alle risorse e periferiche dell' MHP, *System Software* che include la Java Virtual Machine che interpreta ed esegue le applicazioni gestendo inoltre il loro ciclo di vita e le periferiche utilizzate, *Applications* che rappresentano l' interfaccia tra un servizio interattivo e il display di visualizzazione;

- informazioni dettagliate sul profilo *Enhanced Broadcasting*: questo è il profilo base che permette servizi di arricchimento del contenuto audio-video come contenuti multimediali, EPG, super teletext, e giochi. Non sono quindi richieste prestazioni elevate nel decoder, è semplicemente necessario essere provvisti di un accesso al canale di trasmissione broadcast mentre la presenza di un canale di ritorno è opzionale;

- informazioni dettagliate sul profilo *Interactive TV*: tramite un canale di ritorno obbligatorio, al contrario del profilo precedente, è in grado di fornire servizi multimediali interattivi più complessi, come la pubblicità interattiva, le transazioni (home-banking, e-commerce);

- diversi formati contenuti in MHP tra cui JPEG e MPEG-2 video e audio;

- protocolli di trasporto, compreso DSM-CC per la trasmissione broadcast e IP per il canale di ritorno eventuale. Nello specifico caso dei dati, infatti, questi vengono impacchettati tramite il protocollo DSM-CC, ovvero Digital Storage Media - Command and Control, che è uno standard per la trasmissione di dati basato sullo stream MPEG, e viene usato per trasmettere le applicazioni ai ricevitori. Un ricevitore non può richiedere un file specifico dal server, come avviene invece per i PC e come soluzione più ovvia il broadcaster trasmette periodicamente ogni file del filesystem contenente i file delle applicazioni, ed il ricevitore resta in attesa per quelli che gli interessano: questo tipo di soluzione è chiamata carousel.

In un secondo momento è stata rilasciata la specifica MHP 1.1, per implementare il profilo Internet Access che permette, sempre tramite un canale di ritorno obbligatorio, di accedere ai contenuti di Internet.

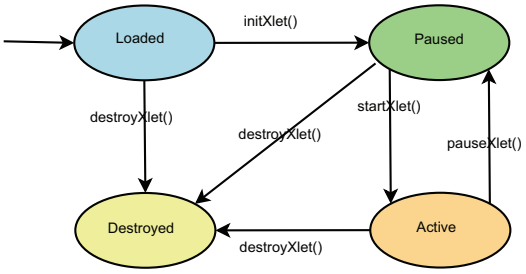


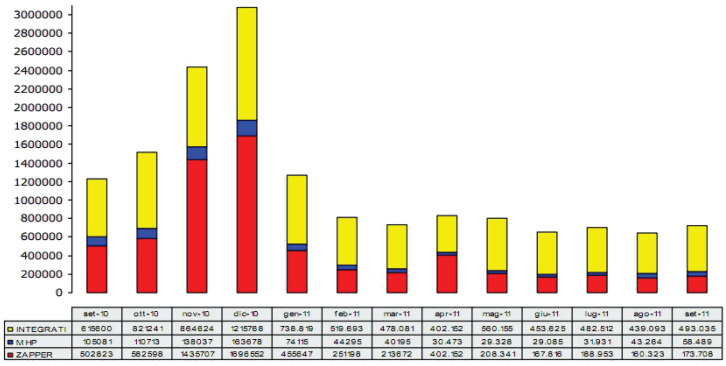
Figura 1.2: Ciclo di vita di una Xlet

Le API utilizzate per l' implementazione di una applicazione MHP, anche chiamata *Xlet*, sono principalmente due. Le API JavaTV [13] sono necessarie al fine di gestire il ciclo di vita dell' Xlet, specificando quali azioni compiere nelle fasi illustrate in Figura 1.2 e permettono inoltre di notificare i passaggi

da uno stato all' altro. Per l' interfaccia grafica vengono invece utilizzate le API HAVi (Home Audio Video interoperability). Queste costituiscono un ampliamento delle GUI Java standard, definendone di nuove create appositamente per l' utilizzo della grafica su un display televisivo che prevede tre livelli: Background layer, Video layer e Graphic layer. Prevede inoltre un set standard di widgets come bottoni, check-boxes e radio buttons, icone, liste corredate di scroll, finestre di dialog e campi di testo. Infine gestisce anche tutti gli eventi connessi alla pressione di un tasto del telecomando, rendendo possibile quindi associarvi diverse azioni sia grafiche che legate al ciclo di vita di una delle Xlet presenti nel TS.

1.3 Teletext

Il passaggio della televisione al digitale, oltre ai miglioramenti della qualità di audio e video, permette una diffusione di dati in una forma grafica più completa ed user-friendly rispetto alle tecnologie passate ma purtroppo il mercato è diventato un grande limite riguardo lo standard MHP. Questa tipologia di applicazioni infatti è utilizzabile solamente se si possiede un decoder di tipo *interattivo*, comprendente cioè una JVM. Come si può notare dagli ultimi dati forniti da DigiTV [14] riguardanti l'anno trascorso tra Settembre 2010 e Settembre 2011, mostrati in Figura 1.3, la maggiorparte delle famiglie italiane ha optato per l' acquisto di un nuovo televisore con decoder integrato. Tra questi solo una bassissima percentuale è interattiva essendo questa tipologia da poco tempo comparsa sul mercato. Nell' acquisto di un decoder esterno invece è stato scelto principalmente lo zapper all' interattivo, intuitivamente per la differenza di prezzo che porta a preferire il primo. Anche nell' ultimo anno le vendite di decoder MHP sono state in calo, segnando un 12% totale, in diminuzione, sui decoder presenti nelle famiglie italiane.



	set-10	ott-10	nov-10	dic-10	gen-11	feb-11	mar-11	apr-11	mag-11	giu-11	lug-11	ago-11	set-11
INTEGRATI	615600	821241	864624	1215768	736.819	519.693	478.081	402.152	560.155	453.625	482.512	439.093	493.035
MHP	105081	110713	138037	183678	74115	44295	40195	30.473	29.328	29.085	31.931	43.264	58.489
ZAPPER	502823	582598	1435707	1696552	455647	251198	213672	402.152	208.341	167.816	188.953	160.323	173.708

Figura 1.3: Vendite decoder in Italia da Settembre 2010 a Settembre 2011

Data la scarsa diffusione della tecnologia MHP si è rivelato indispensabile cercare di fornire le stesse informazioni anche attraverso lo standard Teletext. Nella televisione analogica, infatti, il Teletext è quasi sempre una funzione opzionale e possono esistere quindi televisori che non lo supportano, benchè tutti quelli messi in commercio dopo il 1990 siano invece predisposti per riceverlo. Nella televisione digitale è diventata una caratteristica standard. Il Teletext consiste in semplici pagine di testo, visualizzabili sullo schermo televisivo a richiesta dell'utente ed è utilizzato per fornire agli utenti ogni genere di informazioni. Le immagini sono create tramite caratteri grafici ed i colori a disposizione sono solo i primari e secondari: nero, bianco, ciano, magenta, verde, rosso, blu, giallo. Il Teletext è sempre abbinato ad una emittente televisiva e ogni emittente televisiva non può averne più di uno.

Le cosiddette pagine sono le unità informative pari allo schermo televisivo e basate su una struttura molto rigida composta da 24 righe da 40 caratteri ciascuna: I valori sono stringenti, se i caratteri da mostrare sono meno di 40 bisogna comunque specificare i rimanenti come spazi. Le pagine vengono trasmesse in sequenza e a ciclo continuo e sono identificate da un numero decimale a 3 cifre: quando l'utente seleziona il numero di pagina predispone l'apparecchio televisivo alla memorizzazione e visualizzazione della pagina scelta e non appena il ciclo continuo giunge alla pagina specificata, questa viene visualizzata. Le pagine possono essere a loro volta costituite da sottopagine che hanno la stessa struttura ma sono associate allo stesso identificativo di pagina e vengono mostrate in successione secondo una temporizzazione automatica decisa a priori. Le sottopagine sono identificate da una scritta del tipo X/Y, dove Y rappresenta il numero di sottopagine in cui è suddivisa la pagina e X un numero progressivo che identifica la sottopagina.

Durante le sperimentazioni è stato notato come i tasti colorati del telecomando possano essere associati a una precisa azione ma, a seconda delle impostazioni dispositivo finale, queste specifiche a livello di programmazione potranno essere rispettate o meno, seguendo invece politiche standard che solitamente associano al tasto rosso l'incremento della pagina di una unità, al tasto verde il decremento di una unità, al tasto giallo l'incremento di una decina e al tasto blu l'incremento di un centinaio.

1.4 OpenCaster

I test e le messe in onda che verranno illustrate hanno sfruttato il software open source *OpenCaster*, rilasciato nella sua prima versione il 20 giugno del 2008 e giunto ad oggi alla versione 3.1.4 [15]. OpenCaster è un generatore di transport stream MPEG-2 e manipolatore di pacchetti che permette di creare i flussi che abbiamo precedentemente spiegato, trasformare un file-

system in un DSM-CC carousel e multipexare transport stream a bit-rate costante. I passaggi necessari per creare un TS sono i seguenti:

- **Generazione delle tabelle PMT ed AIT conseguentemente ai flussi che si intende trasmettere**: OpenCaster offre infatti diversi file in linguaggio python in grado di generare le tabelle informative necessarie da inserire nel transport stream, che devono ovviamente essere precedentemente configurate specificando ad esempio i nomi dei servizi, i PID degli elementary stream, e così via.

- **Generazione di uno stream DSM-CC**: una directory contenente tutti i file necessari al funzionamento dell' applicazione, come ad esempio i file grafici ed i file di testo con i contenuti, dev' essere presente sul server che genera i flussi, per poterla inserire poi in un transport stream che andrà ricreato ogni volta che vi è un cambiamento nella directory. Il comando che, nell' ultima versione di OpenCaster, genera il transport stream a partire dalla directory è il seguente:

  ```
  oc-update.sh object_carousel_directory association_tag
  module_version dsmcc_pid carousel_id [compress_mode]
  [padding_on] [clean_off] [DDB_size] [update_flag]
  [mount_frequency]
  ```

  Dove:

  - carousel_directory: la directory da inserire nel transport stream. Il suo nome sarà quello che verrà dato al file .ts di uscita;
  - association_tag: dve essere quello referenziato nella PMT ed AIT;
  - Modules_version: intero che va da 0 a 15 e deve essere lo stesso per tutti i moduli;
  - dsmcc_pid: pid del transport stream che verrà generato. Anche questo dovrà corrispondere a quello associato in PMT;
  - carousel_id: Identificativa di ogni carousel che viene generato, è referenziato in PMT;
  - compress_mode: intero che specifica una tra 3 possibili compressioni del carousel;
  - padding_on: booleano che indica se debba essere inserito del padding nelle varie sezioni;
  - clean_off: booleano che indica se debbano essere eliminati i file temporanei;
  - DDB_size: specifica un payload differente rispetto a quello di default pari a 4006;

  - update_flag: booleano che indica se vi è stata o meno una modifica all' interno della cartella così da forzare il decoder a scaricare ed utilizzare i nuovi file;
  - mount_frequency: indica la frequenza di inserimento di determinate informazioni che accelerino la fase di mount del carousel, di default posta a 2 inserimenti per carousel.

- **Generazione del loop infinito per le pagine di Teletext**: per prima cosa tutte le pagine vengono unite in una unica che è pacchettizzata tramite il comando:

  ```
  txt2pes pages.txt 15 3600 1800 > txtpage.pes &
  ```

  Dove 3600 è il numero di Presentation Time Stamp (PTS) clock per il primo pacchetto, mentre 1800 è l'incremento di PTS clock per ogni pacchetto PES: 1800 indica effettivamente 50 pacchetti Program Elementary Stream (PES) al secondo, quindi il risultato finale è che ogni 15 pacchetti TXT un pacchetto nuovo per il PES sarà generato. In seguito viene creato il TS tramite un secondo comando:

  ```
  pesdata2ts txtpage.pes [teletext_PID] > txt.ts &
  ```

  dove *teletext_PID* è l' identificativo associato al Teletext nelle tabelle informative.

- **Multiplexing**: in Opencaster il mux tra i differenti elementary stream viene effettuato tramite il seguente comando:

  ```
  tscbrmuxer [bitrate_ts1][ts1_name]...
  ...[bitrate_ts1][ts1_name ] > muxed.ts
  ```

  Dove per ogni transport stream da trasmettere si specifica prima il suo bit rate poi il suo nome. Il file in uscita avrà bitrate pari alla somma dei bitrate dei vari transport stream multiplexati.

Capitolo 2

Infrastrutture wired

La Regione Emilia-Romagna (RER) ha recentemente sviluppato il proprio Piano Telematico 2011-2013 [16] anche in sintonia con l' Agenda Digitale Europea [17] e con gli obiettivi che essa pone e per questo esistono numerose comunanze tra le due, tra le quali quelle di interesse per queste ricerche riguardano il Digital Divide e lo strettamente connesso sviluppo delle reti di nuova generazione (Next Generation Networks, NGN). Nello specifico due obiettivi EU sulla banda larga sono quelli di coprire tutto il territorio europeo con banda larga a 2Mb/s entro il 2013 e con banda larga a 30Mb/s (100% della popolazione) e a 100 Mbit/s (al 50% della popolazione) entro il 2020. Il primo è un obiettivo comune con il Piano Telematico della RER e LepidaSpA, su mandato della Regione interverrà in tutte le attività per il superamento del divario digitale, per l' ottimizzazione delle risorse infrastrutturali e per il relativo utilizzo. Inevitabilmente le evoluzioni della rete porteranno verso NGAN (Next Generation Access Networks) basate su fibra ottica e le possibili architetture e soluzioni verranno di seguito mostrate.

2.1 Architetture NGAN

Lo sviluppo di internet negli ultimi anni ha portato a una preponderanza, nell' ambito delle telecomunicazioni, del traffico dati rispetto a quello voce. Sia il numero di uenti connessi che la richiesta di banda stanno aumentando incessantemente e questa tendenza fa sì che il cosiddetto *ultimo miglio* sia un collo di bottiglia a causa dell' infrastruttura ad oggi per la maggior parte in rame [18].

Schematicamente una rete si può rappresentare attraverso un modello gerarchico dove al livello più alto troviamo il *core layer* che rappresenta il *backbone*, la rete a grande distanza. Segue un *distribution layer*, o *backhaul* che rappresenta le reti di raccordo tra il backbone e il terzo livello: l' *access layer* che collega alla rete l'utente finale. In questo contesto l' ultimo miglio,

o *rete di accesso* è riferito all' infrastruttura di rete che collega l' ultima centrale di commutazione all' utente. E' in atto una rivisitazione della rete di accesso che sia in grado di supportare futuri servizi innovativi volti verso la multimedialità e l' interattività.

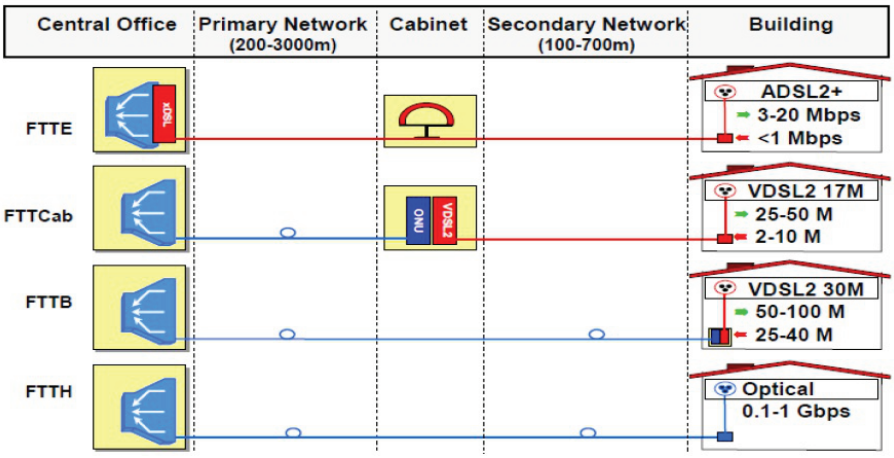


Figura 2.1: Architetture NGAN

Le NGAN nascono in questo scenario con l' idea di partenza di portare in maniera parziale o totale la fibra ottica al posto del rame, secondo le seguenti architetture, riassunte in Figura 2.1, dove la Fiber To The Exchange (FTTE) rappresenta l' attuale situazione adottata, basata sull' impiego di solo linea in rame:

- **Fiber to the Cabinet (FTTC)**: si basa sulla sostituzione del rame nella rete primaria collegando lo Stadio di Linea con l' armadio stradale tramite fibra ottica. Un pro di questa architettura è il limitato investimento iniziale ma rimangono come contro il collo di bottiglia della tratta in rame ed inoltre alla necessità di introdurre armadi stradali più grandi o multipli, tenendo conto della presenza di più operatori alternativi.

- **Fiber to the Building (FTTB)**: consiste nel collegare l'edificio direttamente allo Stadio di Linea con la fibra ottica eliminando i cabinet stradali. Quest' architettura permette di risparmiare i costi di cablatura verticali dei palazzi ed assicurare comunque le massime prestazioni del doppino dato che le distanze coperte risultano molto più brevi del caso precedente.

- **Fiber to the Home (FFTH)**: in questo caso la fibra ottica viene portata direttamente fino in casa dell' utente finale.

I sistemi di accesso per la rete in fibra ottica si possono distinguere invece tra *sistemi punto-punto* (Point-to-Point, P2P) e sistemi basati su *reti passive in fibra ottica* (Passive Optical Network, PON).

Il sistema di tipo punto-punto, prevede collegamenti dedicati in fibra tra l' OLT (dispositivo ottico collocato in centrale) e l' interfaccia ONU (l' elemento ottico dislocato nelle vicinanze dell' utente finale) utilizzando tecnologie Fast Ethernet (100 Mbit/s) oppure Gigabit Ethernet (1 Gbit/s). Essendo il mezzo dedicato, questa soluzione può tranquillamente soddisfare anche esigenze future ma ha costi superiori rispetto alle tecniche d' accesso passive che verranno tra poco illustrate, a causa del numero superiore di fibre da installare nella rete primaria, che comporta una gestione più complessa delle fibre all' interno della centrale.

Una PON (Passive Optical Network) è una rete ottica priva di elementi attivi nel percorso, utilizzando infatti solo uno splitter/accoppiatore passivo poco costoso e senza necessità di alimentazione, che ha il vantaggio di minimizzare l' uso della fibra. Esistono principalmente tre standard di PON:

- **Broadband PON (BPON)**: utilizzano ATM (Asynchronous Transfer Mode) come protocollo di trasporto e per questo sono diventate obsolete in quanto ATM è stato nel frattempo soppiantato dallo standard Ethernet. Prevede bit-rate in upstream compresi tra 155.52 Mbps e 622.08 Mbps e in downstream tra 155.52 Mbps e 1244.16, in combinazioni simmetriche e asimmetriche;

- **Ethernet PON (EPON)**: utilizza il formato di trama ethernet, offrendo la frequenza di cifra simmetrica 1,25 Gbit/s, ed una futura estensione a 10 Gbit/s (10GEPON). Da notare che in questo caso può essere effettuata una disaggregazione *fisica* del collegamento ottico dalla centrale all' utente, nel caso GPON è possibile solo una disaggregazione *logica*;

- **Gigabit-capable-PON (GPON)**: le reti GPON, a differenza delle EPON, non trasportano in modo nativo frame Ethernet , ma tramite il protocollo di incapsulamento GEM (GPON Encapsulation Method), supportano varie tipologie di traffico. Sono in questo caso possibili varie combinazioni di velocità sia simmetriche che asimmetriche, comprese tra 155.52 Mb/s e 2.48 Gb/s. Anche in questo caso vi sarà una futura estensione a 10 Gbit/s (10GPON).

Ad oggi si può notare come la scelta effettuata in altri continenti ed anche quella proposta e prevista dai principali operatori italiani, sia l' architettura FTTC o FTTH con sistema GPON. Questo sistema può basarsi su due differenti tecniche di accesso multiplo al mezzo condiviso: Nella multiplazione e accesso multiplo a divisione di tempo (TDM) vengono utilizzate due lunghezze d' onda differenti per la trasmissione in downstream e upstream

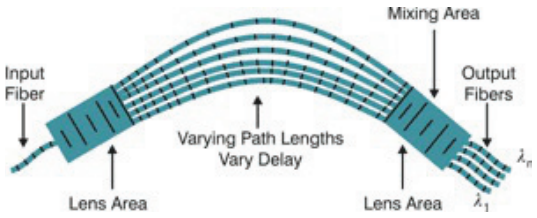


Figura 2.2: Schema di principio di un AWG

(entrambi collegamenti a fibra singola) e le logiche di controllo dell' accesso al mezzo condiviso sono realizzate dal dispositivo OLT che gestisce la banda condivisa. La multiplazione di lunghezza d' onda (Wavelength Division Multiplexing, WDM) si basa invece su tecniche di multiplazione a divisione di lunghezza d' onda che utilizzano come dispositivi degli *Arrayed Waveguide Grating* (AWG), dispositivi passivi che permettono di combinare e di separare lunghezze d' onda multiple. Come mostrato in Figura 2.2 il dispositivo passivo AWG è un dispositivo ottico 1:n che accoglie nella fibra di ingresso un segnale ottico multiplex composto da n colori e lo sfiocca su n fibre di uscita, ciascuna portante una sola lunghezza d' onda. In questo caso si hanno le seguenti principali architetture, a seconda di un differente utilizzo e tipologia degli splitter:

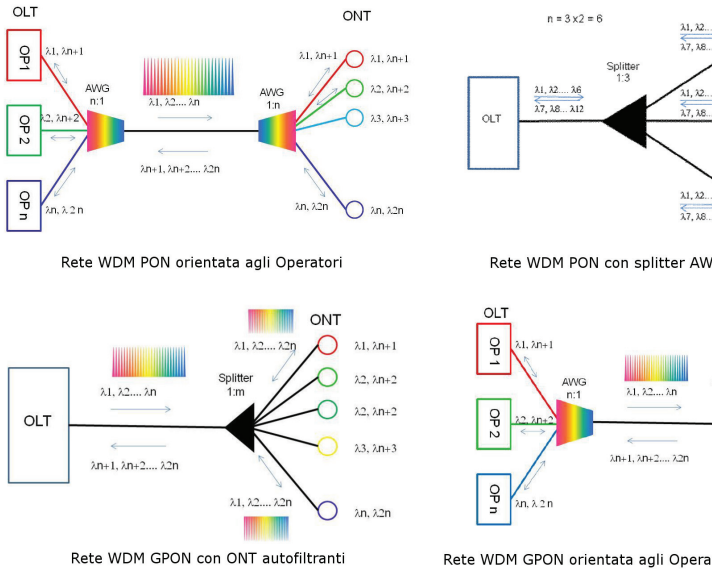


Figura 2.3: Architetture WDM

- **Architettura orientata alla condivisione tra operatori**: In cen-

trale un AWG multipla i segnali ottici prodotti dagli OLT dei vari operatori. Il secondo AWG di demultiplazione va posto in un pozzetto o al basamento dell' edificio.

- **AWG preconfigurati**: restano gli splitter nei pozzetti nodali mentre gli AWG messi in cascata a livello di edificio sono differentemente preconfigurati per demultiplare solo la porzione a loro dedicata nella schiera dei colori che ricevono in broadcast dal primo splitter.

- **ONT auto configuranti**: ONT auto filtranti ovvero capaci di selezionare autonomamente la coppia di colori a loro destinata. I terminali utente ricevono tutta la schiera dei segnali mentre i programmi sono selezionati localmente.

- **Architettura orientata all' operatore**: a ognuno degli n operatori viene affidata una coppia di colori tra le n disponibili. La rete di accesso prevedera uno splitter e ONT auto filtranti.

La loro schematizzazione è illustrata in Figura 2.3

2.2 Multicast

Nell' ambito delle NGN si deve prevedere l' opportunità di ottimizzare la diffusione di pacchetti di servizi differenziati a seconda delle richieste degli utenti ed il multicasting diventa senz' altro una necessità. E' stata quindi effettuata una ricerca esaustiva ed una successiva schematizzazione sia degli algoritmi di routing che dei protocolli affidabili multicast proposti in letteratura.

2.2.1 Algoritmi di Routing

I principali algoritmi di routing sono riassunti nello schema di Figura 2.4 e notiamo come si basino su due approcci principali: *Group-Shared-Tree*, dove un unico albero di distribuzione condiviso da tutti i sender, ha lo scopo di minimizzare la somma dei costi dei link dell' albero multicast e si basa sullo Steiner Tree Problem; *Source-Shared-Tree* dove per ogni sender viene costruito un albero specifico che minimizzi il costo dalla sorgente ad ognuna delle destinazioni. Raggruppandoli secondo queste categorie, analizziamo per primi gli algoritmi **Group-Shared-Tree centralizzati**. Come già accennato si basano sul problema dell' albero di Steiner, ovvero: *dato un grafo G formato da N router ed E link caratterizzati da un costo, dati T <= N nodi che identificano i router che gestiscono gli host che formano il gruppo multicast e appartengono ad un insieme S, l' obiettivo è trovare l' albero di Steiner per S e G che abbia peso complessivo minimo.* Questo

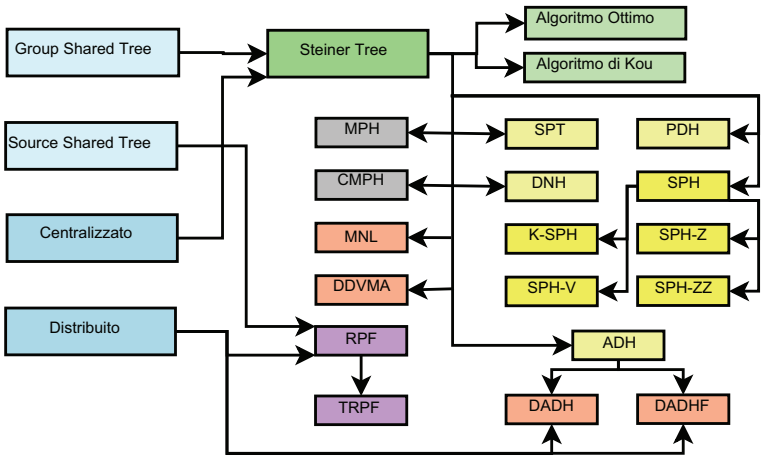


Figura 2.4: Algoritmi di Routing Multicast

problema è NP-complesso: esistono un algoritmo ottimo ed un algoritmo euristico, detto di Kou, che approssima molto bene l' ottimo ma entrambi hanno problemi implementativi notevoli su reti reali in cui si usano quindi euristiche che creino spanning tree non ottimi ma sufficientemente stabili affinchè, date delle variazioni della topologia, il numero di rami aggiunti o eliminati sia minimo [19].

- **PDH** (Pruned Dijkstra Heuristic): deriva direttamente da Dijkstra che restituisce l' albero dei percorsi minimi dalla radice agli altri nodi, tramite ripetuti pruning si tolgono quindi tutte le foglie non appartenenti a S. Ha un basso costo computazionale ma un limite superiore di competitività inaccettabile;
- **DNH** (Distance Network Heuristic): viene generata la rete delle distanze dei nodi di S, che risulta essere un grafo totalmente connesso che ha per vertici i nodi S e i cui archi hanno un costo pari al costo del cammino minimo tra i nodi della rete di partenza;
- **SPH** (Shortest Path Heuristic): inizializza l' albero con un nodo S arbitrario, ad esempio il sorgente, per poi unire i nodi di multicast secondo la loro distanza dall' albero, fino a connetterli tutti. Varianti dell' euristica:
  - **K-SPH** (Kruskal based): invece di partire con un nodo gestisce una *foresta* di alberi che vengono man mano uniti per formare l' albero di Steiner;
  - **SPH-Z**: ripete SPH una volta per ogni nodo di S;
  - **SPH-V**: ripete SPH una volta per ogni nodo della rete, se il nodo iniziale non e di multicast viene temporaneamente unito a S e poi eliminato;

  - **SPH-ZZ**: per ogni coppia di nodi S inizializza l' albero con il loro cammino minimo poi esegue SPH e lo ripete per $s(s-1)$ volte, con s nodi appartenenti a S.

- **ADH** (Avarage Distance Heuristic): ha un meccanismo piu sofisticato per la scelta di sottoalberi da unire, invece di unire i due piu vicini si sceglie un nodo $v*$ intermedio da cui passare tale che sia il *più centrale* per un gruppo di sottoalberi così da facilitarne il successivo collegamento.

Passando ai **Group-Shared-Tree distribuiti**, notiamo che dall' algoritmo ADH sono state ideate due euristiche di tipo distribuito. Le euristiche centralizzate infatti sono impraticabili in caso di reti molto estese dal momento che richiedono una completa conoscenza della topologia di rete, in questi casi e piu efficiente un euristica di tipo distribuito benche sia in generale piu lenta delle centralizzate [20]:

- **DADH** (Distributed ADH): comincia con una foresta di nodi e li connette in successivi alberi sempre piu grandi fino ad arrivare a quello finale;

- **DADHF** (Distributed ADH with Full Connection): identico al caso precedente tranne per una lieve differenza nella scelta del nodo centrale ad ogni round di creazione dell' albero.

Per quanto riguarda i **Group-Shared-Tree con minimizzazione del numero di links**, partendo dall' MPH (Minimum cost Path Heuristic) che consiste nel trovare la distanza tra tutte le coppie di nodi,per poi trovare percorso minimo verso una destinazione e man mano aggiungerlo all' albero precedente finchè tutte le destinazioni vengono connesse, ne è stata proposta una variante [21]:

- **CMPH** (Constrained MHP, ovvero con vincolo sui hop): calcola le distanze tra ogni coppia di nodi tenendo conto del vincolo di hop, cerca un percorso minimo verso una destinazione sempre valutando il vincolo di hop, aggiunge il percorso ottenuto all' albero precedente, ripete il procedimento più volte finchè tutte le destinazioni sono connesse.

Inoltre troviamo il seguente algoritmo:

- **MNL** (Minimum Number of Link Algorithm): l' algoritmo calcola il percorso da una destinazione alla sorgente minimizzando il numero di hop, poi cerca il percorso piu vicino da un' altra destinazione verso l' albero che è stato calcolato [22].

Troviamo anche un algoritmo di tipo **Group-Shared-Tree con minimizzazione della multicast delay variation**, il **DDVMA** (Delay and Delay Variation Multicast Algorithm) pensato per risolvere il problema di variazione di delay in una rete eterogenea [23]. Questa ricerca termina infine con i Source-Shared-Tree distribuiti:

- **RPF** (Reverse Path Forwarding): quando un router riceve un pacchetto multicast con un dato indirizzo sorgente, lo trasmette su tutte le interfacce di uscita solo se il pacchetto è giunto da un link appartenente al proprio shortest path verso il sender in questione;

- **TRPF** (Truncated RPF): i router che ricevono pacchetti multicast pur non essendo connessi ad host appartenenti al gruppo destinazione, inviano un messaggio di *pruning* a monte. Un router che riceve tale messaggio da tutti i suoi successori, itera il procedimento.

2.2.2 Protocolli multicast affidabili

I primi protocolli proposti erano di due tipologie: *Sender-initiated*, dove la responsabilità dell' affidabilità nella ricezione e data alla sorgente, la quale deve ricevere un ACK (Acknowledge) da ogni destinatario prima di liberare la memoria dal pacchetto inviato. Questo porta al problema di ACK implosion. La seconda tipologia è la *Receiver-initiated*, dove la responsabilità in questo caso è data alle destinazioni che, denotando un gap nella sequenza dei pacchetti ricevuti, manda un NACK (Not-Acknowledge) indicando la necessità di una ritrasmissione.Anche in questo caso si puo presentare il problema di NACK implosion [24].
Queste due tipologie, con l' aggiunta di metodi che prevengano i problemi descritti, sono ora parti dei cinque macro-gruppi di tipologie di protocollo che andiamo ad analizzare, sintetizzati in Figura 2.5.

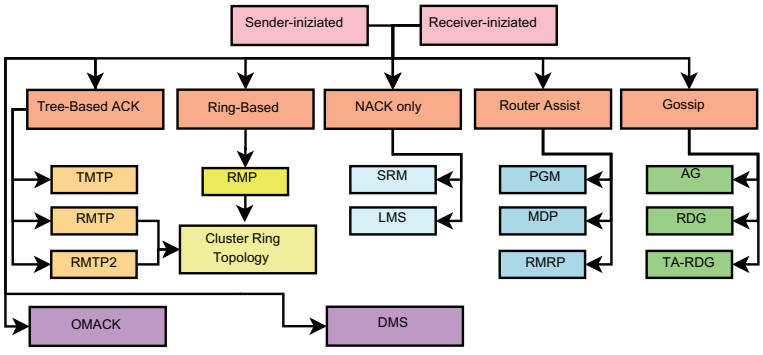


Figura 2.5: Protocolli affidabili Multicast

I **Tree-based Acknowledge** sono caratterizzati dalla divisione dei riceventi in gruppi, e dalla ridistribuzione di responsabilità di affidabilità ad un albero di Acknowledge:

- **TMTP** (Tree-based Multicast Transport Protocol): divide la rete in una gerarchia di domini, ognuno dei quali ha un domain manager [25]:
  - un sender crea un dissemination group tramite un expanding ring search e manda multicast;
  - il sender aspetta ACK dai suoi figli diretti per rilasciare la memoria, quando un domain manager riceve un messaggio dalla sorgente S manda un ACK ai suoi parent di livello superiore
  - se scade il timeout senza ACK il sender rimanda il pacchetto in multicast;
  - per la richiesta di ritrasmissione quando un ricevente nota la mancanza di un pacchetto manda NACK al livello superiore con TTL piccolo, ma prima aspetta un tot di tempo. Se vede passare un NACK per lo stesso pacchetto, reprime la trasmissione del suo.
- **RMTP** (Reliable Multicast Transport Protocol): analogamente a prima suddivide la rete in regioni ognuna delle quali con un DR (Designated Receiver) responsabile degli ACK della sua regione verso il sender. Questo protocollo è indicato per reti wireless o che comunque sono soggette a congestioni [26]:
  - il sender S manda una window di pacchetti a tutti i ricevitori $R_{i,j}$;
  - ogni $R_{i,1}$ (DR della regione) manda il suo stato a S a intervalli periodici informandolo di quali pacchetti abbia ricevuto. Basandosi su questo, S determina che pacchetti mandare e a seconda del numero di richeste decide se farlo in multicast o in unicast;
  - ogni $R_{i,j}$ manda il suo stato a $R_{i,1}$ a intervalli regolari. $R_{i,1}$ localmente manderà pacchetti in unicast o in multicast se il numero di richieste è maggiore di una certa soglia;
  - se c' è ancora spazio nella window S manda altri pacchetti.
- **RMTP II**: prevede un ottimo supporto nelle reti asimmetriche e prevede uno schema in cui alcuni sender mandano dati a molti Receiver [27].
  - i DR mandano richiesta di join al gruppo a TN (top Node), un sender joina a sua volta al gruppo e crea nuovo canale al quale joinano i ricevitori creando l' albero;
  - TN e DR si mandano pacchetti di Heartbeat e i loro figli rispondono, inoltre il sender manda pacchetti null allo scopo di informare di essere attivo;

  - il sender manda un multicast, i receiver e i *DR* lo ricevono e mandano un messaggio di TRACK (Trasmission Acknowledge) propagandolo verso l' alto, quando il sender riceve il TRACK per un pacchetto, lo elimina dalla memoria;
  - se manca un pacchetto nel successivo TRACK verrà segnalato esplicitamente, o in alternativa manda un NACK;
  - se un parent ha il pacchetto richiesto glielo manda altrimente propaga la richiesta al livello superiore;
  - se arriva fino al sender questo lo ritrasmette in multicast;
  - se è abilitato il FEC (Forward Error Correction), non previsto invece in TMTP e RMTP, invece di chiedere il singolo pacchetto si possono chiedere tutti i pacchetti persi nella ultima finestra di w pacchetti;

I **NACK-Only** ovvero Router-assist, invece, non prevedono,ad esclusione di RMRP che è ibrido di tipologia essondo un router-assist ma basato su ACK, la conferma esplicita dell' avvenuta consegna dei pacchetti:

- **SRM** (Scalable Reliable Multicast): caratterizzato da buona affidabilità ma pessima scalabilità [28].
  - quando un membro del gruppo genera dati nuovi li manda in multicast. Ogni membro è responsabile di controllare le perdite trovando un gap nella sequenza dei pacchetti ricevuti;
  - ogni membro manda periodicamente messaggi di sessione annunciando il numero più alto di pacchetto ricevuto;
  - i ricevitori aspettano prima di mandare un pacchetto di controllo e non lo mandano se ne vedono passare uno con le stesse informazioni;
  - qualunque host abbia i dati richiesti li manda (perchè questo non venga mandato da troppi ci si basa sulla gestione di un multicast repair timer in ogni host).
- **PGM** (Pragmatic General Multicast)[29]:
  - la sorgente manda pacchetti multicast sequenziali ai riceventi;
  - quando un ricevente vede che manca un pacchetto, manda un NACK ai suoi parent del PGM tree;
  - i parent confermano a tutti i loro figli con un NCF di aver ricevuto il NACK;
  - i Repair Data sono generati dalla sorgente o da un DLR (Designated Local Repairer);
  - questo pacchetto puo essere il rinvio del pacchetto perso o un FEC;

  - prima di mandare un NACK aspetta per uno slot di tempo in cui se si riceve NCF (NACK Confirmation), Data o Repair Data corrispondenti, si reprime il proprio NACK.

- **LMS** (Light-weight Multicast Serveces): si basa su un recovery gerarchico, la scelta del replier può essere sia statica che dinamica. Un router ritrasmette il dato usando DMCAST (Directed Multicast) cioè creando un messaggio multicast e incapsulandolo in un unicast verso il router che a sua volta lo decapsula per mandare il multicast verso l' interfaccia specificata[30].

  - quando un pacchetto viene perso su un link L , se un vicino ricevente ha il dato e può ritrasmetterlo, questo diventa un replier e chi ha richiesto il dato un requestor;
  - il requestor manderà quindi un NACK solo al replier il quale lo manderà multicast sul link L.

- **MDP** (Multicast Dissemination Protocol): usato in infrastrutture WAN eterogenee, supporta anche gruppi dinamici. Si basa sulla comunicazione tramite i seguenti pacchetti [31]:

  - un sender manda messaggi di due tipologie: MDP_DATA ed MDP_PARITY, quest' ultimo o mandato su esplicita richiesta o proattivamente;
  - MDP_INFO può essere mandato opzionalmente per mandare ulteriori informazioni non essenziali, MDP_CMD invece si usa per indicare operazioni particolari come End Of Transmission;
  - un ricevente manda messaggi di NACK o opzionalmente di ACK: i messaggi di NACK sono generati in corrispondenza di dati persi e le informazioni di sequenza incluse in ogni pacchetto sono usate per rintracciare le eventuali perdite;

- **RMRP** (Reliable Multicast Routing Protocol): E' implementato ed incorporato in ODMRP (On Demand Multicast Routing Protocol) che gestisce, tramite join queries e join replies la membership tra sorgente e membri del gruppo [32]:

  - tutti i pacchetti hanno time-stamp con numeri sequenziali così da identificare dai gap il pacchetto mancante. Ogni nodo mantiene una bitmap di k-bit con lo stato di k pacchetti relativi a quello attuale e la manda ai nodi successivi;
  - i nodi intermedi fanno cahing di alcuni pacchetti del flusso multicast. La dimensione della cache è la più grande tra la dimensione della bitmap e il numero di nodi in downstream, dal momento che quando un nodo intermedio riceve k-bit bitmap deve essere in grado di mandare uno qualunque dei k pacchetti indicati;

  - un pacchetto eècancellato dalla cache quando gli arriva l' ACK da tutti i vicini in downstream.

I **Ring-Based** sono stati inizialmente pensati per applicazioni che richiedessero un ordeinamento totale ed atomicità per i riceventi. Si ha un token site responsabile degli ACK verso la sorgente, i riceventi mandano i NACK al token site . L' ACK mandato verso la sorgente serve anche al meccanismo di passaggio del token, una volta che il token e passato il site puo liberare la memoria.

- **RMP** (Reliable Multicast Protocol): tutti I pacchetti sono identificati dalla tupla (ID, # sequenziale processo, QoS level), sono mandati in multicast e mantenuti da un ricevitore primario detto Token Site [33]:
  - quando un token site riceve uno o piu pacchetti, manda un ACK ai membri, contenente da zero a più identificativi e un timestamp. Così il sender sa che il pacchetto e stato ricevuto ed il timestamp permette l' ordinamento e il riconoscimento dei pacchetti persi;
  - ogni ACK passa il token al site successivo che può accettarlo solo se ha ricevuto tutti i pacchetti fino a quel momento, altrimenti richiede quelli persi. In questo modo si sa quando è stato compiuto un giro completo del ring e quindi quando il pacchetto è diventato stabile;
  - per i NACK ci si basa su due liste: Datalist ha la lista dei pacchetti, nello specifico degli OrderingQ slot che puntano ai pacchetti, e per ognuno di questi indica lo stato di ricezione e la tupla identificativa. Quando in OrderingQ c' è un gap, si manda un NACK per richiedere i pacchetti al gruppo.

Dall' RMP è stata proposta una **Cluster-ring topology** che aumenta la scalabilità dell' RMP [34]:

- **Fase 1**: Clusterizzazione della rete. Detto D il massimo diametro del cluster e $K = Deg_{Est}$ la stima del massimo grado di un nodo della rete:
  - ogni nodo controlla il suo grado, se è almeno pari a 4 comincia a costruire un cluster: ogni potenziale leader manda un *offer* ai vicini, un nodo senza leader sceglie con *accept* l' *offer*. Se un leader ha offer da un nodo di address minore accetta e notifica di non esser più leader. Ogni nodo manda offer fino a distanza D;
  - ripete gli step precedenti da $k = k - 1$ a $K = 3$;
  - gli orfani sono presi nel cluster più vicino, se sono presenti catene tra due cluster vengono suddivise equamente in questi, se il numero di cluster è elevato si fa il merge dei più piccoli e si diminuisce k per poi reiniziare.

- **Fase 2** : Una volta clusterizzata la rete
  - tra i leaders dei cluster si usa un protocollo RMP-Like;
  - dentro i cluster si applica RMTP, in questo modo prima si danno ACK/NACK ai leader poi tra questi si chiede ritrasmissione a chi detiene il token.

L' ultimo gruppo è quello dei **Gossip** che si basano su un maggior trade-off tra affidabilità e scalabilità, rivalutati per reti AdHoc che non sono deterministiche e sono soggette a frequenti variazioni topologiche.

- **AG** (Anonymous Gossip) [35]:
  - nella prima fase un qualunque protocollo non affidabile è usato per mandare il messaggio in multicast;
  - nella seconda fase si usa gossip per recuperare i messaggi persi da altri membri che li hanno invece ricevuti, tramite i seguenti periodici gossip rounds: per prima cosa il nodo A sceglie nodo B, altro membro del gruppo; in seguito A manda a B informazioni su quali pacchetti abbia o meno ricevuto; infine B controlla se ha ricevuto i pacchetti mancanti ad A e si scambiano i pacchetti.
- **RDG** (Route Driven Gossip): lo scopo è raggiungere una affidabilità probabilistica in cui se un gruppo di membri manda un flusso di M pacchetti, un membro riceve una frazione m degli M pacchetti con probabilità $PM(m)$ dove m e il grado di affidabilità è P la distribuzione di probabilità dell' affidabilità [36]:
  - un nodo manda un Group Request cercando gli altri menbri del gruppo a cui joina;
  - i membri che ricevono questo messaggio aggiornano la loro lista A_{view} e tornano un Group Reply al sender;
  - il sender a sua volta con i Reply aggiorna la sua A_{view};
  - la validità di ogni relationship e controllata periodicamente e quando la A_{view} ha una dimensione minore di una certa soglia il nodo forza una join session;
  - ogni membro periodicamente manda un Gossip Message a F nodi ascelti randomicamente della sua A_{view}, inserendo i pacchetti che ha ricevuto e storato nel suo Buffer, e l' id dei pacchetti non ricevuti;
  - un pacchetto viene rimosso da Buffer dopo esser stato mandato T_q volte in un Gossip Message;
  - un membro che riceve un Gossip Message: rimuove i membri obsoleti dalla sua A_{view}, aggiorna la A_{view} con il nuovo membro,

aggiorna il Buffer con i nuovi pacchetti ricevuti, risponde al Gossip Pull solo se il pacchetto ricevuto non sara subito reinserito in un Gossip Message;

- i nodi sul percorso, se passa un pacchetto che serve anche a loro, lo prendono e aggiornano il loro Buffer.

- **TA-RDG**(Topology Aware RDG): come RDG ma ad Aview sono assegnati dei pesi: più è lungo il percorso più è basso il peso, così un nodo sceglie con probabilità più alta un membro più vicino. Ad esempio: $Peso = \frac{1}{Lunghezzacammino}$.

2.2.3 Schematizzazione dei protocolli tramite AUML

Data l' eterogeneità del numero e tipologia di messaggi che vengono inviati/ricevuti tra i diversi attori presenti nei protocolli multicast appena illustrati, si è ritenuto che trovare un unico modo per schematizzare lo scambio di pacchetti sia indispensabile per un rapido confronto grafico dei protocolli che permetta di notarne subito le complessità e differenze principali [37]. Per questo scopo è stata utilizzata un' estensione dell' UML (Unified Modeling Language). L' UML è nato con l' intento di uniformare e formalizzare le metodologie di molti approcci object-oriented ed è diventato uno standard OMG (Object Management Group) nel 1997 [38]. Si basa su differenti tipi di diagrammi suddivisi in modelli:

- **Modello use-case**: specifica delle azioni che un sistema può effettuare interagendo con altri attori;

- **Modelli statici**: differenziati in class diagrams, object diagrams e package diagrams, descrivono semantiche statiche di dati e messaggi in una modalità concettuale;

- **Modelli dinamici**: suddivisi in sequence diagrams, collaboration diagrams, activity diagrams e statechart. Descrivono le relazioni ed interazioni tra diversi oggetti e le loro transizioni di stato;

- **Modelli di implementazione**: component diagrams e deployment diagrams, che descrivono la distribuzione dei componenti su diverse piattaforme.

UML è a sua volta basato su un OCL(Object Constraint Language). Questo è un semplice linguaggio formale che esprime più semantiche all' interno delle specifiche UML e definisce vincoli, pre e post condizioni delle operazioni e così via. Nonostante ciò UML è carente su alcuni importanti punti:

- nella rappresentazione MAS (Multi Agent Systems) è utile avere a disposizione messaggi sia sincroni che asincroni;

- in UML non ci sono modelli di protocollo ed è necessario modellare un protocollo di comunicazione tra uno o più ruoli che uno stesso agente può assumere nel suo ciclo di vita;
- infine, UML descrive solo oggetti, i quali hanno una interfaccia *statica*: quando il loro ruolo cambia, offrono comunque gli stessi servizi. Gli agenti sono più dinamici e quando variano il ruolo, variano anche i loro servizi.

Nel 2000, Bauer, Muller and Odell introducono una nuova importante classe di diagrammi chiamata appunto Protocol Diagrams. I diagrammi UML sono stati estesi includendo ruoli degli agenti, lifelines che supportano azioni multithread, semantiche dei messaggi ampliate, protocolli innestati e parametrizzati. Queste specifiche sono diventate un FIPA (Foundation for Intelligent Phisical Agents) Working draft nel 2003 a seguito di qualche minima modifica. Quindi, Agent UML è un' estensione pensata per la schematizzazione di ambiemti MAS [39] [40].

Tornando alla nostra necessità di schematizzazione di protocolli multicast, abbiamo bisogno di rappresentare le seguenti situazioni:

- i nodi di rete devono poter cambiare il loro ruolo durante il loro ciclo di vita passando da root a leaf, da leader a non leader di un gruppo, e così via;
- la trasmissione dei pacchetti tra gli host deve poter avvenire sia in modalità sincrona che asincrona;
- le trasmissioni possono essere multiple da un sender a più receiver e devono prevedere come risposta molteplici comportamenti possibili.

Questo contesto è decisamente similare a un MAS, così abbiamo deciso di utilizzare le specifiche Agent UML per descrivere i protocolli multicast. Il Working Draft dal titolo *FIPA Modeling: Interaction Diagrams* contiene appunto le specifiche riguardanti gli AUML Interaction Diagrams, che verranno impiegate nella schematizzazione:

- **Protocol Frame**: è rappresentato da un rettangolo che incapsula tutti gli elementi coinvolti all' interno del protocollo di interazione. L' etichetta in alto a sinistra contiene il nome del protocollo e parole chiave optionali necessarie per il diagram;
- **Lifeline**: rappresenta un ruolo impersonato da un agente. Quando viene creata una lifeline per un preciso ruolo, questo ruolo diventa attivo per il protocollo, quindi la lifeline è presente tanto a lungo quanto il ruolo rimane in uso. E' rappresentata da un rettangolo in testa, contenente il nome del ruolo, seguito da una linea tratteggiata verticale.

Da notare che nel caso in cui più di un agente dovessero avere quel ruolo, si specificherà il nome con <*nome-agente*>:<*ruolo*> I messaggi che possono essere scambiati tra diversi agenti sono:

  - **Messaggi asincroni**: relativo a quando un agente manda un messaggio asincrono senza cedere il controllo, è rappresentato da una freccia con punta semplice;
  - **Messaggi sincroni**: relativo a quando un agente manda un messaggio cedendo il controllo del thread, è rappresentato da una freccia con punta piena;
  - I vincoli sui messaggi sono mostrati tra parentesi quadre accanto alla lifeline del sender;
  - I vincoli temporali invece sono scritti tra parentesi graffe accanto alla lifeline del receiver;

- **Splitting/Merging paths**: sono cammini che si suddividono o si uniscono tra loro all' interno del protocollo. Nella prima versione dell' Agent UML erano realizzati tramite connettori AND, OR, XOR e sdoppiamenti della lifeline in uscita. Questo Working Draft invece ha aderito alle proposte dell' UML 2.0 usando i *Combined Fragments*: questi sono rettangoli contenenti i differenti cammini possibili risultanti dall' interazione. Ogni rettangolo ha un' etichetta in alto a sinistra che specifica la tipologia di operatore coinvolta:

  - **alt** (Alternative - XOR): rappresenta il fatto che più cammini sono possibili partendo da uno stato specifico dell' interazione e solo uno di essi verrà scelto;
  - **brk** (Break): descrive le eccezioni nelle interazioni, fermando la normale interazione ed eseguendo una serie di messaggi;
  - **par** (Parallel - AND): esprime l' esecuzione parallela di cammini diversi e rappresenta l' invio concorrente di diversi messaggi;
  - **neg** (Negative): messaggi non validi per la specifica interazione;
  - **ign** (Ignore): versione più debole di *neg*, dove i messaggi devono essere considerati vuoti nell' interazione;
  - **loop**: un set ordinato di messaggi che deve essere inviato più volte. Il numero di volte viene specificato o da un limite superiore/inferiore o da una condizione di guardia.

Mostriamo quindi come possono essere schematizzati i protocolli multicast, analizzandone alcuni tra quelli illustrati nel paragrafo 2.2.2 attraverso l' Agent UML.

I protocolli **tree-acknowledge based** sono i primi che schematizzeremo:

- **TMTP**: Il protocollo è mostrato in Figura 2.6, i ruoli e i messaggi scambiati sono i seguenti:

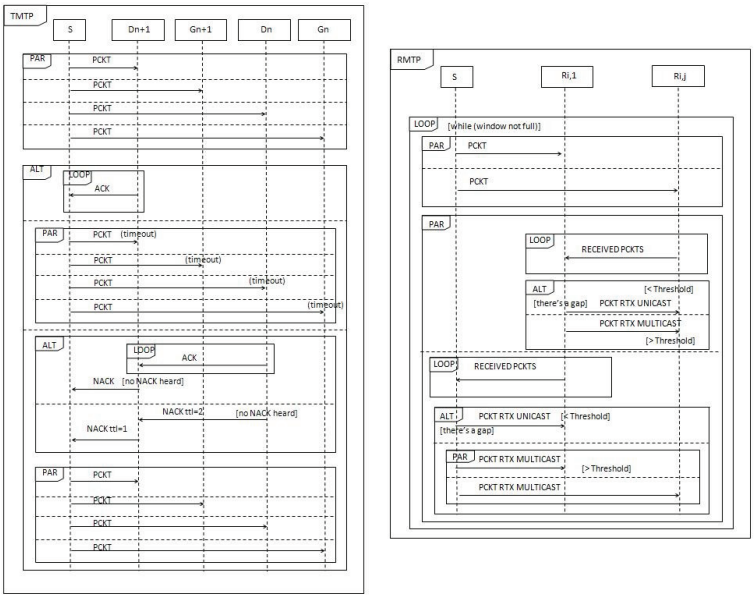


Figura 2.6: Schematizzazione dei protocolli multicast TMTP ed RMTP

  - S: Sender
  - D_{n+1}: Domain Manager in un dominio di livello $n+1$
  - G_{n+1}: Group member in un dominio di livello $n+1$
  - D_n: Domain Manager in un dominio di livello n
  - G_n: Group member in un dominio di livello n
  - $Timeout$: se non vi è un ACK in risposta, quando scade il timeout il sender ritrasmette il pacchetto multicast.
  - $NoNACKheard$: per prevenire un' implosione di NACK, un NACK è mandato solo se il group member non vede un pacchetto con le stesse informazioni proveniente da un altro membro dello stesso gruppo.

- **RMTP** : come mostrato in Figura 2.6, include i seguenti ruoli e messaggi:

  - S: Sender
  - $R_{i,1}$: Designated Receiver in una determinata regione della rete j
  - $R_{i,j}$: Receiver nella regione j
  - $Whilewindownotfull$: se questa condizione è verificata, il sender invia un altro pacchetto
  - $ReceivedPackets$: lista dei pacchetti ricevuti

  - *Threshold*: se il numero di richieste per lo stesso pacchetto supera questa soglia, il pacchetto sarà ritrasmesso in modalità multicast, se è minore della soglia invece in unicast
  - $There's agap$: Questo nodo trova una lacuna nella lista dei pacchetti ricevuti.

- **RMTP-II** : la sua schematizzazione proposta in Figura 2.7 include i seguenti ruoli e messaggi scambiati:
  - S: Sender
  - TN: Top node
  - DR: Designated Receivers
  - $TN - B$: TN Backup
  - LN: nodi Receiver
  - $HeartbeatandHeartbeatreply$: utilizzato per mantenere una connessione tra un TN e i suoi DR e tra un DR e i suoi LNs
  - $NoMissing/Missing$: Nel TRACK MESSAGE sono previste informazioni sul fatto che un pacchetto sia o meno mancante.

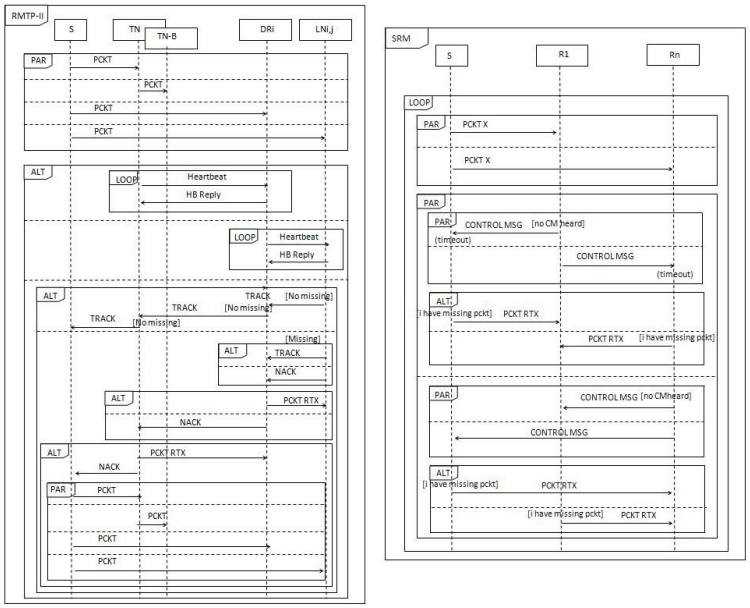


Figura 2.7: Schematizzazione dei protocolli multicast RMTP-II e SRM

La seconda categoria è quella dei protocolli **NACK-Only and Router Assist**:

- **SRM** : mostrato in Figura 2.7 comprende i seguenti ruoli e messaggi:

- S: Sender
- R_i: Group Members
- $ControlMessage$: contiene il più alti ID di pacchetto che il membro del gruppo abbia ricevuto
- $NoCMheard$:per prevenire un' implosione di NACK, un NACK è mandato solo se il group member non vede un pacchetto con le stesse informazioni proveniente da un altro membro dello stesso gruppo
- $Ihavemissingpacket$: se uno dei nodi ha il pacchetto richiesti, lo ritrasmette.

- **LMS** : Si basa sui seguenti ruoli e messaggi scambiati ed è rappresentato in Figura 2.8
  - S: Sender
  - RP: Replier
  - $R_1 ... R_n$: Receivers
  - $Ihavethepacket$: Questo nodo ha il pacchetto mancante
  - $lostpacket$: Questo nodo ha perso un pacchetto

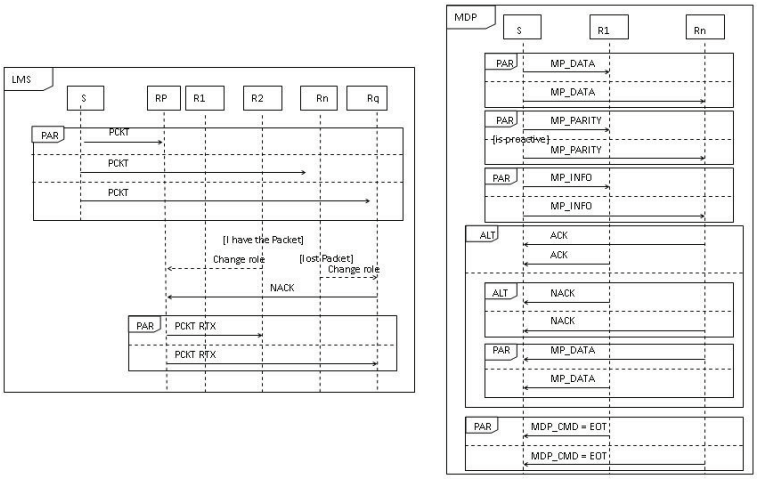


Figura 2.8: Schematizzazione dei protocolli multicast LMS ed MDP

- **MDP** : E' basato su differenti tipologie di pacchetti ed i seguenti ruoli, come illustrato in Figura 2.8.
  - S: Sender
  - $R_1, ... R_n$: Receivers
  - MDP_DATA: Data packets

  - MDP_PARITY: usato per recuperare i dati persi, può essere proattivo oppure richiesto
  - MDP_INFO: informazioni opzionali
  - MDP_CMD: differenti tipologie di operazioni come ad esempio *End Of Transmission*.

La terza categoria analizzata è quella dei protocolli **Ring-based**, dove viene introdotta la presenza di un Token:

- **RMP** : La schematizzazione è presentata in Figura 2.9 e comprende i seguenti elementi:
  - S: Sender
  - G_i: Group Member
  - TS: Token Site
  - $Timeout$: se non vi è un ACK di risposta, quando scade il timeout, il sender ritrasmette il pacchetto multicast
  - $NoNACKheard$: per prevenire un' implosione di NACK, un NACK è mandato solo se il group member non vede un pacchetto con le stesse informazioni proveniente da un altro membro dello stesso gruppo
  - $Changerole$: azione che avviene quando vi è un passaggio di token.

L'ultima categoria che schematizzeremo è quella dei protocolli **Gossip-Based**:

- **AG** : sono coinvolti i seguenti ruoli e messaggi, come schematizzato in Figura 2.10.
  - S: Sender
  - A,B,C: Group members
  - $ChosenB/C$: ogni nodo sceglie un altro membro del gruppo e scambia con esso i propri pacchetti
  - $PCKTLIST$: lista dei pacchetti che un nodo ha ricevuto e lista di quelli mancanti.

- **RDG** : illustrato in Figura 2.10, prevede i seguenti attori, ruoli e messaggi.
  - S: Sender
  - $R_1, ...R_n$: Receivers

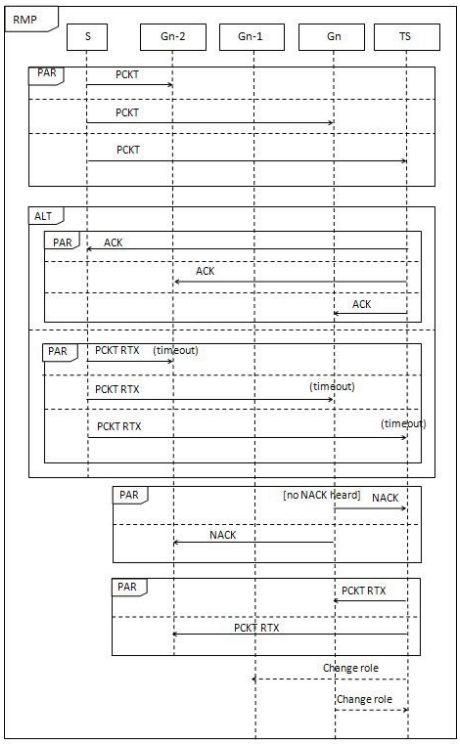


Figura 2.9: Schematizzazione del protocollo multicast RMP

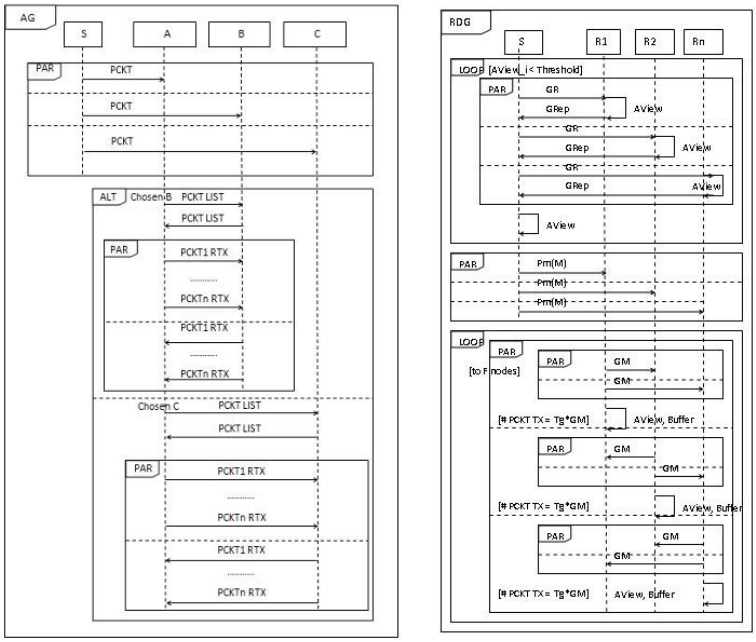


Figura 2.10: Schematizzazione dei protocolli multicast AG ed RDG

- A_view: lista dei membri
- $Buffer$: lista dei pacchetti ricevuti
- GR: Group Request
- G_rep: Group Reply
- GM: Gossip message che contiene i pacchetti ricevuti e gli ID dei pacchetti persi
- $toFnodes$: Il GM iè inviato a un numero F di nodi random
- $PCKTTX = T_g * GM$: un pacchetto viene inviato da questo nodo nel suo GM solo se non è già stato inviato in altri T_g Gossip Messages.

Abbiamo quindi mostrato in questo paragrafo come si possa semplicemente schematizzare, in modo omogeneo, qualunque tipologia di protocollo multicast.

Capitolo 3

Servizi per il Digitale Terrestre

Come introdotto nel Capitolo 1, grazie alla possibilità di effettuare prove sul campo tramite il canale digitale terrestre regionale LepidaTV, sono state implementate diverse tipologie di servizi, richiesti dagli enti della RER, per il cittadino. Illustriamo innanzitutto le strutture base di un servizio implementato con tecnologia MHP e di uno per Teletext.

3.1 Struttura di un servizio MHP

Le scelte strategiche effettuate lungo il percorso sono state molteplici. Per prima cosa nell' ambito della televisione digitale esistono due tipi di interattività:

- quella definita *forte* sfrutta un canale di ritorno, ovvero il canale che trasmette le informazioni in una direzione opposta a quella associata al normale canale di comunicazione: la trasmissione broadcast. Qeusto canale è realizzato attraverso una linea telefonica PSTN o un cavo Ethernet logicamente collocati tra un decoder e un Centro Servizi.;

- l' interattività *debole*, invece, non utilizza un canale di ritorno e fornisce tutte le informazioni date dall' ente attraverso il carousel dei dati trasmesso dal broadcaster. Solo le informazioni richieste dall' utente durante la navigazione dell' applicazione MHP saranno scaricate dal carousel sul decoder.

LepidaTV ha scelto quest' ultimo tipo di interattività allo scopo di avere uno schema trasmissivo più semplice e maggiore sicurezza nella distribuzione delle informazioni. Inoltre alcune fasce di popolazione risultano ancora

diffidenti verso una connessione, che spesso risulta anche fisicamente difficile da effettuare non essendo televisori e telefoni solitamente vicini nelle abitazioni, tra un decoder e una linea telefonica, temendo inaspettate spese.

Un' ulteriore scelta, concernente l' usabilità, è stata effettuata: tutti i servizi sono stati progettati su un layout standard così che l' utente possa comprendere più facilmente la navigazione, uguale per tutti i servizi. La navigazione è basata esclusivamente su tasti numerici del telecomando per aiutare i cittadini abituati alla televisione analogica che non hanno mai utilizzato precedentemente frecce direzionali, tasto OK, o tasti colorati. I colori del layout sono stati scelti in modo appropriato per avere un buon contrasto tra sfondo e testo e creando una semplice connessione visiva tra una specifica area del layout ed i tasti utilizzati per la sua navigazione, che richiamano il colore dell' area. Nello specifico è prevista un' area *azzurra* di primo filtraggio dei contenuti (ad esempio, per provincia o categoria di interesse) navigabile con i tasti 1 e 3, una seconda area di filtraggio *verde* (ad esempio per data) navigabile tramite 4 e 6, un' area *rossa* dove viene mostrato il testo della notizia che si può scorrere con 7 e 9. Infine è prevista una pagina di help, mostrata alla pressione del tasto zero, che spiega nel dettaglio le aree dello specifico servizio, come può essere navigato e fornisce i contatti dell' ente che eroga tutti i suoi contenuti. Il layout comprende anche una versione ridimensionata dell' audio/video in onda così che l' utente possa sempre sapere quale contenuto sia trasmesso e quindi uscire dal servizio se ne è interessato.

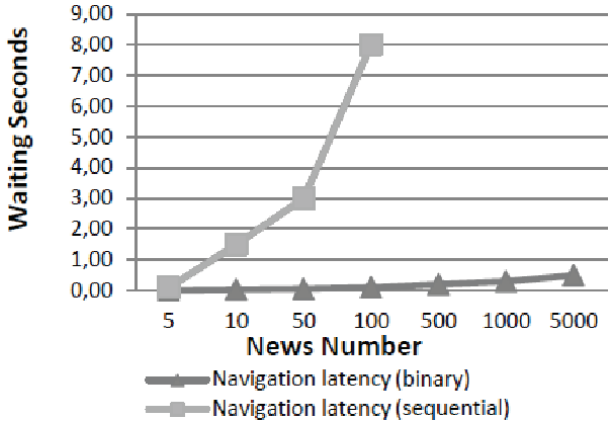


Figura 3.1: Prestazioni nella navigazione usando file sequenziali e binari

Un' ulteriore problematica è stata quella di trovare un modo per navigare istantaneamente, sia in avanti che indietro, una grande quantità di notizie. E' stato scelto quindi di usare file ad accesso randomico [41], dove le informazioni sono memorizzate in strutture binarie. L' accesso randomico permette l' utilizzo di puntatori e di leggere il file partendo da un byte specifico, attraverso il metodo *fseek()*. Così facendo, quando desideriamo scorrere all'

indietro le notizie, mentre con un normale file sequenziale dovremmo leggere tutto il file ogni volta fino alla notizia richiesta, con un file binario possiamo semplicemente usare un puntatore che è stato precedentemente salvato in un vettore, mentre si scorrevano le notizie in avanti.

Come si può vedere in Figura 3.1, usando una struttura binaria infatti, non solo la visualizzazione della prima notizia, ma anche nella successiva navigazione, sia in avanti che indietro, dei contenuti disponibili, sono state quasi istantanee per tutti i test effettuati. Al contrario, una navigazione in avanti risulta abbastanza veloce anche se si sta utilizzando un file sequenziale, mentre una navigazione all' indietro richiede molto tempo di attesa già con file che contengono un numero di notizie abbastanza basso, ad esempio sono necessari 3 secondi per navigare 50-60 notizie [42].

Figura 3.2: Catena relativa a un servizio MHP

Partendo da queste specifiche, un servizio MHP viene implementato seguendo i passi schematizzati in Figura 3.2. Per prima cosa uno o più feed RSS sono scaricati su un server: è stata scelta questa tipologia di feed perchè è già presente su gran parte dei siti web delle organizzazioni locali così che gli enti non debbano fare nulla di nuovo rispetto a ciò che già è esistente, per questa ulteriore diffusione delle loro informazioni attraverso la televisione digitale terrestre. Questi download di RSS sono effettuati da un eseguibile Java posto in cron sul server e lanciato ogni notte. Le informazioni sono conseguentemente memorizzate in uno o più file binari [43], per i motivi già spiegati.

Come già accennato, tutti i decoder definiti *interattivi* sono basati sullo standard MHP e sono in grado di interpretare le applicazioni attraverso la JVM a bordo. Sul server è presente per ogni applicazione una cartella, al cui interno sono posti tutti i file statici necessari alla visualizzazione dell' Xlet: la classe Java dell' applicazione stessa, i file binari e le immagini. La classe è implementata rispettando gli standard illustrati nel Capitolo 1, basata quindi sulle JavaTV API che descrivono le azioni da compiere nelle fasi di load, start e delete dell' applicazione, e sulle API HAVi per il layer

grafico. Implementa inoltre la lettura delle notizie contenute nei file binari e la loro visualizzazione nella navigazione. Ogni cartella è pacchettizzata e trasformata in un TS, successivamente multiplexato con i flussi audio e video e trasmesso in broadcast.

3.2 Struttura di un servizio Teletext

La principale criticità è in questo caso rappresentata dal fatto che il tempo di refresh delle sottopagine è basato sul numero di pagine presenti nel carousel tra una sottopagina e quella successiva [44]: è necessario quindi avere la giusta sequenza di pagine, eventualmente generando copie di esse se non sufficienti, che permetta di ottenere un tempo di refresh tale che l' utente riesca a leggere tutto il testo della notizia prima che la sottopagina cambi. In un esempio possiamo avere che, se ogni secondo 250 pagine del TS possono essere scaricate e lette dal decoder, per avere un refresh di 8 secondi, dovremmo avere $250 * 8 = 2000$ pagine tra ogni coppia di sottopagine. I blocchi di informazioni, cioè le singole pagine, sono quindi poste e copiate in un loop che segua uno schema della tipologia rappresentata in Figura 3.3.

	Danza(2)	Lirica(0)	Musica(7)	Prosa(10)	Ragazzi(1)	Altro(3)	Pagina 100	Menu Servizio
Loop1	0	0	0	0	0	0	0	0
Loop2	0	0	0	0	0	0	0	0
Loop3	0	0	0	0	0	0	0	0
Loop4	1	0	1	1	0	1	0	0
Loop5	1	0	1	1	0	1	0	0
Loop6	1	0	1	1	0	1	0	0
Loop7	0	0	2	2	0	2	0	0
Loop8	0	0	2	2	0	2	0	0
Loop9	0	0	2	2	0	2	0	0
Loop10	1	0	3	3	0	0	0	0
Loop11	1	0	3	3	0	0	0	0
Loop12	1	0	3	3	0	0	0	0
Loop13	0	0	4	4	0	1	0	0
Loop14	0	0	4	4	0	1	0	0
Loop15	0	0	4	4	0	1	0	0
Loop16	1	0	5	5	0	2	0	0
Loop17	1	0	5	5	0	2	0	0
Loop18	1	0	5	5	0	2	0	0
Loop19	0	0	6	6	0	0	0	0
Loop20	0	0	6	6	0	0	0	0
Loop21	0	0	6	6	0	0	0	0
Loop22	1	0	0	7	0	1	0	0
Loop23	1	0	0	7	0	1	0	0
Loop24	1	0	0	7	0	1	0	0
Loop25	0	0	1	8	0	2	0	0
Loop26	0	0	1	8	0	2	0	0
Loop27	0	0	1	8	0	2	0	0
Loop28	1	0	2	9	0	0	0	0
Loop29	1	0	2	9	0	0	0	0
Loop30	1	0	2	9	0	0	0	0

Figura 3.3: Distribuzione delle pagine in un loop di Teletext

Un programma Java, partendo dal totale numero di pagine, calcola quale pagina abbia il massimo numero di sottopagine e questa sarà considerata la pagina *pilota*, ovvero la quarta colonna della Figura 3.3 d' esempio, in cui ogni sottopagina è ripetuta 2 o più volte a seconda del numero totale di pagine. Tutte le pagine avranno un numero globale di sottopagine equivalente a questo numero di ripetizioni, dove ogni sottopagina è ripetuta in accordo col numero finale. Tutte le sottopagine dell' esempio sono scritte, riga dopo riga, nel carousel finale ripetuto in loop e permetterà di avere lo stesso tempo di refresh in ogni pagina. Infatti ricordiamo che quando un

utente apre il Teletext e richiede una specifica pagina, questa verrà mostrata quando sarà il suo turno nel loop del carousel.

Come mostrato in Figura 3.4, lo schema che porta alla realizzazione di un servizio Teletext è il seguente: il feed RSS è anche in questo caso letto da un eseguibile Java ogni notte e le notizie sono memorizzate in file binari. Una seconda applicazione Java usa questi file e incapsula le loro informazioni in un file python, seguendo le specifiche dello standard e creando tutte le pagine del Teletext. Partendo da questo file python, come spiegato nel Ca-

Figura 3.4: Catena relativa a un servizio Teletext

pitolo 1, tramite OpenCaster sono create le pagine e pacchetizzate in un TS pronto per essere multiplexato con i servizi interattivi e i flussi audio/video, mettendo a disposizione dell' utente un classico Teletext basato su caratteri ASCII, 8 colori, che può essere navigato premendo tre cifre relative a una specifica pagina, o attraverso i tasti colorati (rosso, verde, giallo e blu) presenti su tutti i telecomandi.

Ad oggi il Teletext di LepidaTV include un servizio informativo riadattato, basato su doppio filtraggio, presente anche tra i servizi MHP. In questo specifico caso viene scelta per prima una delle province dell' Emilia Romagna, poi una categoria di evento ed infine tutte le notizie relative a queste due scelte sono mostrate tramite sottopagine temporizzate. Inoltre sono proposti i programmi TV della giornata presente e successiva, suddivisi in tre slot giornalieri: dalle 00.00 alle 10.00, dalle 10.00 alle 18.00 e dalle 18.00 alle 24.00. Le diverse sottopagine di una pagina, a seconda della tipologia del decoder o televisore, possono seguire il loop automatico ma anche essere salvate sul dispositivo e navigate successivamente con le frecce direzionali.

3.3 Servizi sviluppati

I servizi ad oggi implementati si riferiscono a quattro diverse categorie, basate su specifiche richieste emerse da diversi enti del territorio.

3.3.1 Launcher e menu iniziale

Inizialmente la lista dei servizi disponibili poteva essere acceduta tramite il tasto APP del telecomando, che legge la tabella informativa AIT con la lista delle Xlet e le propone all' utente in un menu standard. In seguito all' incremento del numero dei servizi proposti si è invece scelto di seguire il normale iter al quale si è abituati con i canali più conosciuti. Come mostrato in Figura 3.5 infatti, tra le Xlet presenti una, chiamata *Launcher* è specificata come *autostart*, ovvero con partenza automatica senza richiesta dell' utente. Questa mostra un' immagine che invita l' utente a premere il tasto rosso.

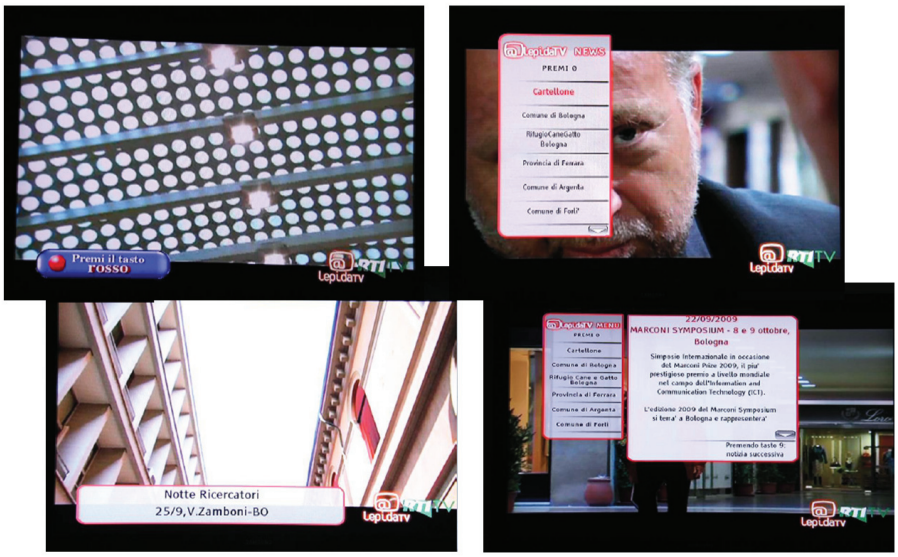


Figura 3.5: Launcher, spot e menu su LepidaTV

Se questo non accade vengono mostrati degli *spot* ogni 5 minuti che riportano notizie importanti riguardanti LepidaTV o eventi sul territorio. Su un database sono presenti i testi degli spot, le date di inizio e fine di validità, la loro frequenza di visualizzazione e la loro priorità. Un eseguibile Java ogni notte li legge e a seconda della frequenza e priorità specificate decide in che sequenza mostrare tutti quelli validi, memorizzandoli poi in un file binario che verrà letto dalla classe del Launcher.

Se al contrario l'utente preme il tasto rosso, come si può sempre vedere in Figura 3.5, viene mostrato il *Menu* dov'è possibile sia leggere le ultime novità di LepidaTV sia scorrere con le frecce direzionali i servizi della lista e selezionare con OK quello di interesse. Alla sua apertura viene distrutta l' Xlet launcher, così come alla scelta di uno dei servizi viene distrutto il menu. All' interno di ogni servizio invece, premendo il tasto BACK del telecomando

viene distrutta l' Xlet del servizio e caricato il menu, premendo EXIT invece si carica sempre il Launcher, distruggendo le altre applicazioni aperte.

3.3.2 Informativo

Il primo servizio interattivo implementato [45] è stato definito con l' intento di mostrare all' utente contenuti informativi. In questo caso il feed RSS è sempre scritto in uno stile standard di RSS ed è attualmente utilizzato dal Comune di Bologna, Provincia di Ferrara, Comune di Argenta, Rifugio del cane e del gatto di Bologna, Comune di Forlì. Il tag <*category*> sarà usato per avere una prima categorizzazione all' interno dell' applicazione, mentre la data specificata nel tag <*date*> costituirà un secondo filtraggio temporale, ad esempio suddividendo ogni categoria in settimane, mostrando le ultime cinque.

In alcuni casi, come ad esempio il Cartellone della RER, il secondo filtraggio può essere effettuato anche attraverso altri criteri, ad esempio una partizione territoriale basata sulle province coinvolte. Per ogni doppio filtraggio un appropriato file binario viene creato e includerà, in ordine: il numero delle notizie disponibili; il peso in byte del titolo della prima notizia; il testo del titolo precedentemente letto all' interno del tag <*title*>; il numero di pagine in cui il testo della notizia dev' essere suddiviso al fine di una corretta visualizzazione sul monitor televisivo e, per ogni pagina il peso in byte della parte del testo in essa contenuto ed infine il testo della notizia letto dal tag <*content*> del feed RSS. Quando un servizio è stato selezionato e viene

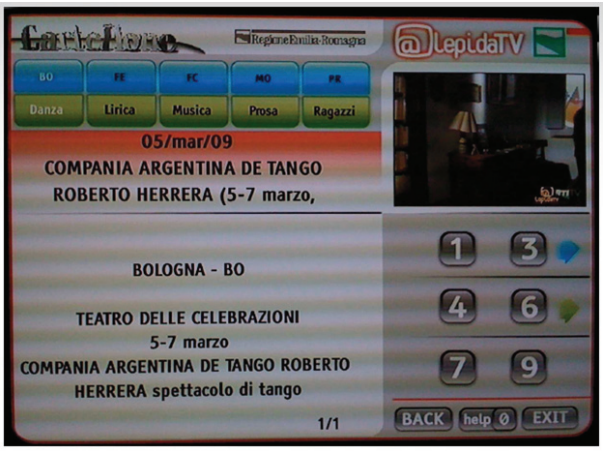


Figura 3.6: Esempio di servizio informativo su LepidaTV

visualizzato, come mostrato dalla fotografia di un televisore in Figura 3.6 con l' esempio del Cartellone, l' area blu mostra le province che l'utente può selezionare attraverso i tasti 1 e 3; l' area verde in modo similare mostra le

categorie, navigabili con 4 e 6. Nell' area centrale una notizia viene sempre mostrata ed ogni volta che un utente scorre uno dei filtri, la prima notizia corrispondente viene visualizzata ed il numero totale di contenuti disponibili per quel doppio filtraggio (in basso al centro nel layout) viene aggiornato. L' utente può poi navigare le notizie con i tasti 7 e 9 e nel caso in cui il testo sia troppo lungo, la notizia si può scorrere con le frecce direzionali. Questi tasti sono rappresentati da due icone nell' angolo in basso a sinistra, e sono graficamente identici alle frecce presenti sui telecomandi, così che l'utente possa trovarli più semplicemente. Il servizio on air con il più elevato numero di notizie raggiunge, giornalmente, più di 3000 contenuti e, grazie all' indicizzazione tramite file binari che abbiamo descritto precedentemente, queste possono essere sempre navigate in modo istantaneo.

3.3.3 Visita guidata

All' interno dello stesso layout, spiegato nel paragrafo precedente, è stato richiesto dal Comune di Argenta di inserire tra i filtri dell' area blu una *galleria*, come mostrato nella fotografia della Figura 3.7, che proponga la virtualizzazione di una visita guidata. Questo servizio è stato quindi creato al fine di permettere alle organizzazioni locali che possiedono percorsi turistici e culturali, o ospitino importanti mostre od eventi che vorrebbero sponsorizzare, di mostrarne un' anteprima attraverso la televisione digitale terrestre.

Ogni notte un altro programma Java, in cron sul server, scarica tutte le immagini lasciate in una cartella concordata sul server dell' organizzazione, poi le ridimensiona per ottimizzarne la loro visualizzazione a monitor, anche quando il televisore è di grandi dimensioni, mantenendo una buona qualità pur crichiedendo poca banda. Il feed RSS utilizza in questo caso alcuni tag definiti appositamente: <*text*> e <*image*>. La visita guidata si compone di una serie di questi due tag. Nel corrispondente file binario sarà memorizzato, per primo, il numero di tag presenti sul feed, dal momento che ciascuno di essi, indipendentemente dalla sua tipologia, verrà mostrato in una pagina diversa. A seguire un intero che specifica la tipologia di tag: 1 se è un' immagine, 0 se è testo descrittivo. Nel primo caso nel file binario verrà scritta la lunghezza del nome dell' immagine ed il suo nome, altrimenti le informazioni saranno memorizzate come precedentemente descritto per il servizio informativo.

Quando il servizio viene aperto, l'utente vedrà nell' area blu la categoria *Gallery*, alla quale corrisponderanno nel secondo filtraggio una o più visite guidate. La selezione di una di queste comporterà la lettura del relativo file binario e all' interno dell' area rossa del layout, dedicata ai contenuti, il numero totale di pagine verrà mostrato ed una serie di immagini e testo alternati inizierà a ciclare. Questo ciclo ha una temporizzazione predefinita e la pagina cambierà dopo 10 secondi, se sta mostrando testo, e 5 secondi

Figura 3.7: Esempio di visita guidata su LepidaTV

per le immagini. Il ciclo finirà quando l'ultima pagina della visita viene visualizzata, ma può essere messo in pausa in ogni momento premendo i tasti 7 e 9, permettendo all' utente di navigare manualmente senza dover rispettare la temporizzazione di default e potendo così focalizzare la propria attenzione sulle pagine di maggiore interesse.

3.3.4 Accesso condizionato

Questo servizio è necessario quando un ente desidera mostrare dati sensibili agli utenti ed una iniziale autenticazione è richiesta per mantenere la privacy. La sperimentazione in questo caso è stata effettuata per FER (Ferrovie dell' Emilia Romagna) che desiderava rendere visibili tramite digitale terrestre i turni del personale; questi ovviamente non dovevano però essere accedibili da tutti gli altri telespettatori di LepidaTV.

Gli orari vengono forniti su fogli di calcolo ed ogni mese un programma Java scarica i nuovi dati, li interpreta e li memorizza su file binari. In questo caso serviranno due file binari: il primo includerà tutti gli username in ordine alfabetico, seguiti da un puntatore che indicizza la sua corrispondente password nel secondo file; in questo file binario sarà memorizzata la password, il numero di informazioni disponibili, seguite dal loro titolo e contenuto, eventualmente suddivise in sottopagine come già illustrato. Quando il servizio viene aperto, come mostrato in Figura 3.8 , per prima cosa viene presentata la richiesta di uno username. Questo inserimento può essere fatto in due modi: se username e password sono composti solo da cifre, l'utente semplicemente userà i tasti numerici del telecomando; se sono richiesti anche i caratteri alfabetici, coerentemente con il layout standard, all' interno dell' area blu compariranno le lettere dell' alfabeto ordinate, che potranno essere

scorse tramite i tasti 1 e 3, il tasto 5 invece verrà utilizzato per confermare la lettera selezionata, e 8 per cancellare l'ultimo carattere inserito. E' stata effettuata questa scelta perchè l' utilizzo di una tastiera qwerty grafica, navigabile attraverso le frecce direzionali, o la pressione multipla di un tasto numerico che produce diverse lettere emulando le funzionalità di un telefono cellulare, potrebbero essere due alternative destabilizzanti per gli utenti affetti da *knowledge divide*.

Figura 3.8: Esempio di servizio con accesso condizionato su LepidaTV

Lo step successivo è il controllo dello username, effettuando una ricerca dicotomica nel primo file al fine di avere una veloce risposta. Se lo username esiste viene richiesta la password. L' Xlet confronterà quella inserita dall' utente con quella puntata nel secondo file e, se corrispondono, verranno effettuate le solite azioni spiegate per il servizio informativo: il numero di contenuti presenti verrà mostrato e titolo e contenuto saranno visualizzati.

Non è previsto il layout in quanto quando l' applicazione viene chiusa, sarà completamente distrutta in ogni sua parte, incluse tutte le variabili usate durante la navigazione, come ad esempio username e password. Oltre alla richiesta specifica di FER è possibile utilizzare la stessa struttura per altri contesti, ad esempio per seguire l'iter di pratiche, visualizzare le multe a proprio carico, visualizzare risultati di analisi mediche, eccetera.

3.3.5 Autovalutazione

Nell' ultimo anno è emersa in particolare una nuova esigenza: l' implementazione di un servizio che possa essere collegato ad un sistema di *t-learning* [46], che permetta quindi ad utenti registrati di verificare il proprio apprendimento tramite un test, reso disponibile dal decoder, successivamente ad una lezione mandata in onda su LepidaTV [47]. Ogni test potrà essere ef-

fettuato solo per un limitato lasso di tempo, solitamente una o due ore dalla fine della lezione.

Molti problemi sono stati affrontati: innanzitutto LepidaTV ha deciso, come spiegato, di non utilizzare un canale di ritorno quindi per trasmettere informazioni dall' utente al Centro Servizi è necessario utilizzare un mezzo alternativo: si è scelto che le risposte vengano inviate tramite SMS, dal momento che la maggior parte delle persone possiedono almeno un telefono cellulare. Per questa ragione è stato settato un SMS gateway sul server che gestisce questi messaggi e memorizza le informazioni ricevute in un database appropriato che verrà descritto a breve. Vogliamo inoltre prevenire il suggerimento delle risposte tra gli utenti durante il test, quindi le domande che verranno di volta in volta mostrate, saranno randomicamente scelte dall' Xlet partendo da un set molto vasto di domande disponibili, tutte fornite dall' organizzazione che realizza anche le lezioni audio/video. Anche la creazione del codice che l' utente deve inviare tramite SMS risulta critico: si deve tener traccia delle domande scelte dal servizio di autovalutazione e delle risposte date e questo dev' essere possibile tramite un codice breve così che l' utente non abbia difficoltà nel digitarlo sul cellulare. Le informazioni verranno quindi compresse e cifrate in una stringa alfanumerica, così da assicurare anche l'impossibilità di un utente di falsificare il suo risultato.

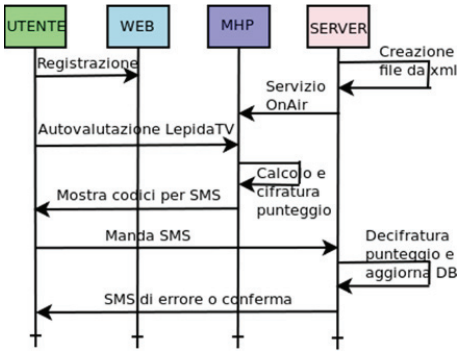


Figura 3.9: Sequence diagram del servizio di autovalutazione

Il *sequence diagram* mostrato in Figura 3.9 mostra come vari elementi coinvolti nel Servizio debbano cooperare l' un l' altro ed in che ordine ciò deve avvenire [48]: per prima cosa l' utente deve iscriversi sul sito web relativo al corso, specificando come numero di cellulare quello che verrà sempre usato per inviare l' SMS al termine di ogni test. Poi può essere seguita la lezione su LepidaTV e al suo termine, tramite un decoder interattivo, si risponderà a un predefinito numero di domande, scelte randomicamente dall' applicazione MHP a partire da un file binario precedentemente creato partendo da un file XML, contenente tutte le domande disponibili. Quando scade il timeout o l' utente termina manualmente il test, l' applicazione cifrerà le domande

chieste e le loro risposte, mostrando infine all' utente la struttura dell' SMS da inviare. Quando il Centro Servizi riceve il messaggio, lo decifra, aggiorna le informazioni memorizzate nel database e manda in risposta all' utente un SMS che conterrà un errore o una conferma in caso di successo. I tre attori principali di questo scenario, che verranno ora analizzati nello specifico, sono: l' interfaccia web, il Centro Servizi e il servizio MHP stesso.

3.3.5.1 Interfaccia Web

Un' interfaccia è già stata prevista su un sito web, che include principalmente due schede: la prima è dedicata alla registrazione dei partecipanti al corso mentre la seconda è una sorta di pannello d' amministrazione.Il server connesso all' SMS gateway include un database con due tabelle di default chiamate *inbox* ed *outbox*, i cui record sono rispettivamente i messaggi in ingresso e in uscita. Una ulteriore tabella dal nome *selfevaluation_inbox* è previst, così da permettere a più di un servizio l'utilizzo dello stesso gateway, dove gli SMS cono poi filtrati attraverso un codice specifico e copiati nell' apposita tabella. Altre sei tabelle sono necessarie per questo servizio specifico, come schematizzato in Figura 3.10. La tabella *user* memorizza informazioni

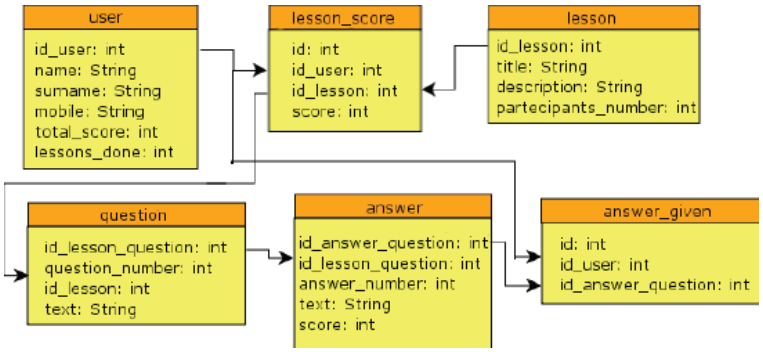


Figura 3.10: Tabelle del database del servizio di autovalutazione

riguardanti ogni partecipante registrato, in particolare: il numero di cellulare che verrà usato per mandare il codice via SMS, il numero totale di test che ha effettuato, per poter successivamente seguire un determinato criterio di ammissione, ed il punteggio totale. La tabella *lesson* memorizza per ogni test il suo codice, titolo ed una breve descrizione degli argomenti trattati. Inoltre tiene traccia di quanti utenti registrati hanno effettuato il test per ciascuna lezione. Le tabelle *question* e *answer* contengono tutte le domande disponibili e le loro risposte complete di punteggi, per ogni test proposto, mentre i record di *given_answers* specificano per ogni utente quali domande sono state scelte dall' applicazione e quali risposte sono state date. Infine, *lesson_scores* memorizza per ogni lezione chi ha terminato il test e quale punteggio ha ottenuto.

Tornando alla struttura dell' interfaccia grafica, vediamo il suo collegamento con il database:

- **User Registration**: l' utente compila il form con i suoi dati personali ed il suo numero di cellulare. Quando i dati vengono salvati, questo numero è controllato e se risulta già presente viene dato un messaggio d'errore all' utente, altrimenti un nuovo record è inserito nella tabella *user* con i campi *total_score* e *lessons_done* impostati ad un valore iniziale pari a zero.

- **Administration Panel**: un login iniziale è ovviamente richiesto. La prima parte della scheda prevede che l' amministratore possa inserire dati riguardanti una nuova lezione che verrà mandata in onda con il relativo test, specificando il suo codice, titolo e descrizione. Al momento del salvataggio dei dati, il sevrer controlla se il codice inserito è già esistente. Se lo è, un messaggio d' errore è mostrato sulla pagina web, in caso contrario un nuovo record è inserito nella tabella *lesson* con il campo *partecipant_number* al valore iniziale zero. La struttura del database è stata decisa anche in previsione di fornire all' amministratore un' ampia serie di informazioni riguardanti il corso. Infatti, la seconda parte di questo pannello è dedicato a molteplici statistiche, ad esempio: selezionando un utente l' amministratore può sapere quanti test ha effettuato, quali e con che punteggio; selezionando una lezione può sapere chi ha effettuato il test correlato, il punteggio per ogni partecipante, il massimo, minimo e medio punteggio ottenuto; quanti utenti hanno risposto a una specifica domanda; quali risposte son state maggiormente scelte per ogni domanda; il punteggio medio relativo ad una specifica domanda; per ogni domanda quanti partecipanti non hanno saputo rispondere.

3.3.5.2 Centro servizi

Il Centro Servizi si occupa di diversi compiti riguardanti la gestione del database, lo scambio di SMS, lo scaricamento e la manipolazione dei dati relativi al test e, inoltre, pacchettizza il servizio in un TS così che possa essere multiplexato con gli altri flussi e mandato in onda. I dati relativi ad uno specifico test sono memorizzati in un file XML così strutturato:

```
<lesson>
<title>Lesson code composed by five cyphers</title>
<question>
<text>Question 1</text>
<answer score=" score 1">Answer 1_1</answer>
<answer score=" score 2">Answer 1_2</answer>
<answer score=" score 3">Answer 1_3</answer>
```

```
</question>
...
</lesson>
```

il tag root della struttura è <*lesson*>, mentre <*title*> specifica il codice della lezione, i tag *question* includono il testo delle domande e tutti i relativi tag *answer* delle risposte con l'attributo *score* del punteggio. Questo file XML è compilato ogni settimana dall' ente che gestisce il corso ed in seguito un comando bash nel cron del server chiama un eseguibile java che legge il file e memorizza i suoi dati in due file binari in accordo alla struttura mostrata in Figura 3.11 Il primo dei due, *question.txt*, include il codice della lezione,

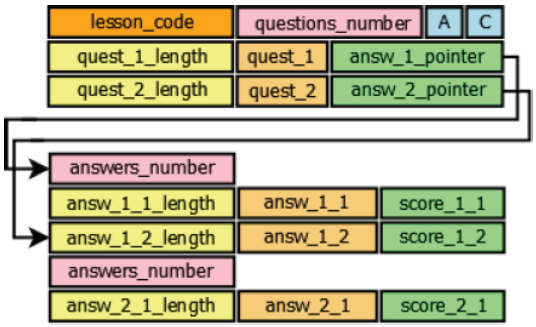


Figura 3.11: Struttura dei file binari del servizio di autovalutazione

il numero totale di domande disponibili, A e C, che sono due parametri del Generatore Congruenziale Lineare (Linear Congruential Generator, LCG) che verrà spiegato a breve, e per ogni domando è specificata la sua lunghezza, il suo testo e un puntatore al secondo file che specifica il byte di inizio delle relative risposte. Nel secondo file, *answers.txt*, per ogni domanda viene per prima cosa specificato il numero totale di risposte, seguito, per ogni risposta, dalla sua lunghezza, il testo ed il punteggio associato.

Notiamo che se, come esempio, assumiamo di avere un numero massimo di domande disponibili composto di 4 cifre (quindi fino a 9999) e al massimo 1 cifra per le risposte (da 0 a 9, vincolo che risulta sensato in quanto un numero maggiore di risposte farebbe perdere troppo tempo all' utente e creerebbe inoltre confusione dando troppe opportunità di scelta), se decidiamo di mostrare 7 domande all' interno del test, dovremmo scrivere un codice di $4 * 7 = 28$ cifre per specificare quali sono state scelte randomicamente e altre $2 * 7 = 14$ cifre per le risposte, per un totale di 42 cifre totali, stringa decisamente troppo lunga. L' LCG è stato infatti introdotto al fine di limitare la lunghezza di questo codice: permette infatti al server di dover conoscere solo 3 parametri, A, C e il seme iniziale, per scoprire quali domande sono state scelte dallo stesso LCG posto all' interno dell' Xlet. Così avremmo il seme iniziale di 4 cifre che con le 14 cifre delle risposte crea un codice di

lunghezza totale di 18. Per l' implementazione dell' LCG è necessario eseguire i seguenti passi: innanzitutto dato il numero totale di domande, che è anche il modulo M usato nella formula finale, una applicazione Java calcola il moltiplicatore A e l' incremento C, in modo tale da permettere all' LCG di avere un periodo lungo M e generare una serie di M interi pseudorandomici, tra zero ed M, senza ripetizioni. Perchè questo sia possibile devono essere verificate le seguenti condizioni:

1. C e M sono relativamente primi;
2. $A - 1$ è divisibile da tutti i fattori primi di M;
3. $A - 1$ dev' essere un multiplo di 4 se anche M lo è;
4. Se $M = 10^q$, C non deve essere divisibile per 2 o 5 e $A(mod 20) = 1$;
5. Se $M = 2^q$, $C(mod 2) = 1$, cioè C dev' essere un numero dispari e $A(mod 4) = 1$;
6. Se $C = 0$, il seme iniziale X_0 dev' essere diverso da zero.

L' LCG è quindi definito dalla seguente formula ricorsiva:

$$X_{n+1} = (A * X_n + C)(mod M) \tag{3.1}$$

Dove X_n è la sequenza di valori pseudorandomici e X_0, il seme iniziale, è calcolato semplicemente dal metodo Java *Math.random(M)*. I due file binari creati vengono copiati nella cartella del servizio di autovalutazione, assieme ai file .png necessari al' interfaccia grafica e la classe dell' applicazione MHP. Questa cartella viene poi pacchettizzata in un appropriato TS ed infine multiplexato con audio, video, altri servizi in un unico carosello trasmesso in broadcast su LepidaTV.

Continuando l' esame delle azioni svolte dal Centro Servizi, in aggiunta alle funzionalità già descritte e connesse all' interfaccia web, quando un nuovo SMS arriva nella tabella *inbox* del database, viene subito controllato se risulta valido corrispondentemente al formato *<Service Code>.<Lesson Code>.<Encrypted Answers>*. Quindi viene verificato che non sia già esistente un record relativo alla specificata coppia di numero di cellulare e codice lezione all' interno della tabella *lesson_score*, se non è presente allora il server decifra il codice seguendo lo schema mostrato in Figura 3.12, parte *(a)*

I primi tre caratteri sono trasformati da base 60, stringa che può includere tutte le 10 cifre e tutte le lettere alfabetiche esclusa la O sia maiuscola che minuscola, a base 10 e la stessa trasformazione è effettuata per il secondo gruppo di quattro caratteri. Questi due interi sono poi affiancati creando un intero di 11 cifre. In seguito questo numero è diviso di nuovo in due gruppi dove le prime 4 cifre rappresentano il seme iniziale e le altre 7 sono

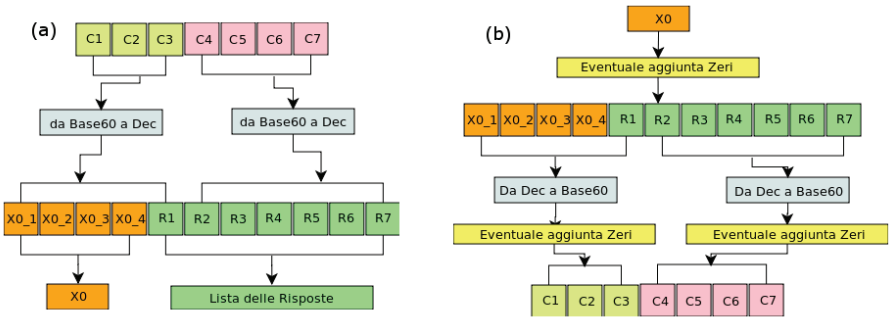


Figura 3.12: Decifratura (a) e cifratura (B) del codice

le risposte date alle 7 domande. Quando termina il processo di decodifica: viene calcolato il punteggio totale ottenuto dall' utente; un nuovo record è inserito nella tabella *lesson_scores* indicando che quel partecipante ha effettuato il test relativo a quella particolare lezione ottenendo il punteggio specificato; il campo *partecipant_number* della tabella *lessons* è aggiornato in relazione al codice di lezione specificato nell' SMS e vengono aggiornati anche i campi *total_score* e *lessons_done* della tabella *user*; per ogni risposta è inserito un nuovo record nella tabella *answer_given* dove viene specificato l' id dell' utente e dove *id_answer_question* descrive univocamente la tripletta *lesson_code*, *question_number*, *answer_number*.

La gestione degli SMS si basa su due file PHP sempre attivi in loop, che lavorano su tre tabelle del database: *inbox*, *outbox* and *selfevaluation_outbox*.

- **SMS_services.php**: questo primo file legge tutti i record memorizzati nella tabella *inbox* che hanno il campo *processed* posto a zero, parsa il testo leggendo la prima stringa che descrive il codice del servizio e se il codice corrisponde a quello dell' autovalutazione copia il messaggio nella tabella dedicata *selfevaluation_inbox*, settando il campo *processed* a uno.

- **SelfEvaluationManager.php**: questo secondo file legge direttamente dalla tabella *selfevaluation_inbox* tutti i record che non sono ancora stati processati e controlla se il codice lezione e il numero di cellulare sono corretti ed esistenti nelle loro tabelle dedicate, se così non è un SMS di errore è inviato all' utente notificando le specifiche dell' errore avvenuto. Altrimenti, se tutto è corretto, la stringa è decifrata ed avvengono le azioni spiegate precedentemente di aggiornamento di informazioni nel database. Un SMS è sempre comunque inviato all' utente creando un record contenente, come testo, il messaggio di errore o conferma di salvataggio dati, nella tabella di default *outbox* che viene letta da un demone Linux, *smsd* presente sul server.

3.3.5.3 Applicazione MHP

Gli utenti possono accedere al servizio di autovalutazione nello stesso modo utilizzato per tutte le applicazioni MHP: tramite il menu che appare premendo il tasto rosso. Quando il servizio viene aperto è mostrato il solito layout standard: infatti, anche in questo caso abbiamo deciso di mantenere le stesse funzionalità, ad eccezione di qualche piccola differenza dovuta alla più complessa interazione tra utente e interfaccia. L' area blu relativa al primo filtraggio non è utilizzata e viene invece mostrato un countdown mentre i tasti 1 e 3 vengono coerentemente segnalati come non attivi; le domande possono essere navigate nell' area verde usando i soliti tasti 4 e 6 del telecomando; le risposte si possono scorrere con 7 e 9 e il numero totale di risposte disponibili per ogni domanda è sempre mostrato in basso al centro.

Vengono introdotti altri due tasti: il tasto 5 è usato per selezionare la risposta scelta. Fino alla fine del test l' utente può cambiare le proprie risposte un numero illimitato di volte. Dopo il testo della domanda è sempre specificato se è stata selezionata o meno una risposta e quale è stata scelta. Premendo il tasto 8, infine, l' utente può terminare il test prima dello scadere del countdown. Se non viene premuto l' 8 invece, al termine del tempo a disposizione le risposte sono memorizzate in un array. Se alcune risposte non sono state date viene posto uno zero in corrispondenza della domanda relativa e l'array viene infine codificato e compresso.

Per la cifratura abbiamo notato che tutti i più conosciuti algoritmi di cifratura, come ad esempio il Data Encription Standard (DES), che esistono già implementati in ogni linguaggio di programmazione, si basano sulla suddivisione del messaggio in blocchi di bytes. Questo è un problema nel nostro caso in quanto l' Xlet può mostrare solo caratteri ASCII standard e quindi dobbiamo essere sicuri di ottenere una stringa finale di questo tipo. Per cifrare il messaggio è stata quindi implementata una semplice formula reversibile, basata sui seguenti passi, schematizzati anche in Figura 3.12 parte *(b)*:

1. il seme iniziale, generato dall' applicazione MHP con il metodo *Math.random* viene scritto usando 4 cifre, aggiungendo zeri se necessario;

2. queste 4 cifre sono affiancate alle 7 che indicano le risposte date;

3. il numero di 11 cifre ottenuto è suddiviso in un gruppo di 5 e uno di 6 cifre e questi 2 interi vengono convertiti in base 60;

4. degli zeri sono eventualmente aggiunti a queste due stringhe al fine di ottenere una prima stringa di 3 caratteri e una seconda di 4;

5. Queste due stringhe affiancate compongono il codice finale di 7 caratteri.

Il risultato è mostrato all' utente al termine del test, spiegando come l' SMS debba essere scritto ed indicando il numero dell' SMS Gateway del Centro Servizi, come mostrato dalla foto di Figura 3.13. Ovviamente, durante que-

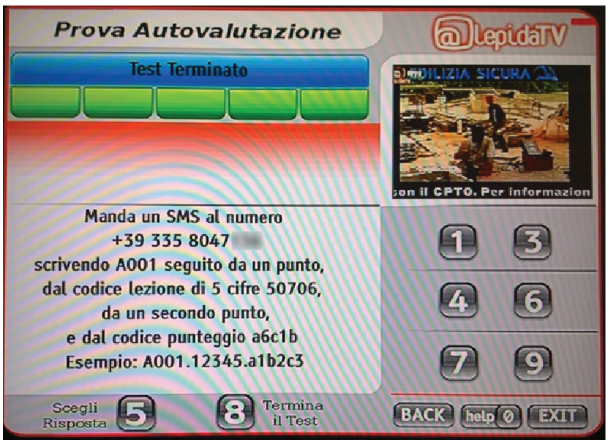


Figura 3.13: Esempio di Servizio di Autovalutazione

sta fase finale tutti i tasti del telecomando, eccetto BACK e EXIT, sono disabilitati. Infine notiamo che anche in questo servizio è stata inckusa una pagina di help, mostrata alla pressione del tasto zero, dove vengono spiegate nello specifico le funzionalità legate ad ogni tasto del telecomando utilizzato.

3.4 Analisi delle prestazioni

Lo scopo è confrontare le tempistiche necessarie per leggere una specifica notizia di interesse in un servizio MHP e nello stesso servizio con tecnologia Teletext. Innanzitutto è necessario definire alcuni parametri:

- T_{menu}: Tempo di apertura del menu (MHP)
- $T_{launcher}$: Tempo di apertura del Launcher (MHP)
- T_{serv}: Tempo di apertura del servizio di interesse (MHP). Questo è calcolato a seconda del variare della dimensione del servizio all' aumento del numero di notizie, tramite l' ottimizzatore di bitrate.
- T_{click}: Tempo impiegato per leggere, premere il tasto idoneo del telecomando e avere la risposta dal televisore.
- T_{sub}: Tempo che intercorre tra una sottopagina del teletext e quella successiva in rotazione automatica (TXT). Questo è pari a T_{wait} se $T_{wait} > 6$, altrimenti è posto pari a 6, effettuando copie di alcune pagine così da assicurare questo tempo, per dare abbastanza secondi

all' utente per leggere la notizia prima di passare alla sottopagina successiva.

- T_{wait}: Tempo di attesa per il caricamento di una pagina (TXT). Questo è dato da:
$$\frac{N_{news} + N_{pagStandard}}{N_{pagSec}} \tag{3.2}$$
- F_1: Numero di opzioni per il primo filtraggio
- F_2: Numero di opzioni per il secondo filtraggio
- N_{news}: Totale delle notizie presenti
- N_{serv}: Numero dei servizi presenti, compreso quello in esame
- $N_{pagStandard}$: Numero di pagine di Teletext escluse quelle del servizio in esame
- N_{pagSec}: Numero di pagine caricate per secondo, ovvero:
$$\frac{bitrate\_in\_Kb\_assegnata\_al\_Teletext}{peso\_in\_Kb\_di\_una\_pagina\_di\_Teletext} = \frac{300,5}{1,3} \tag{3.3}$$

Possiamo quindi definire le seguenti formule per calcolare il tempo necessario:

- **MHP tempo medio**: La sequenza di tempi è data dal tempo di caricamento del launcher, pressione tasto rosso, caricamento menu, pressione frecce fino al servizio giusto (si suppone qui a metà della lista dei servizi), pressione tasto ok, apertura servizio, numero di click pari al primo filtraggio (qui supponiamo che il filtro desiderato sia l' opzione a metà dei filtri possibili), numero di click pari al secondo filtraggio (qui supponiamo che il filtro desiderato sia l' opzione a metà dei filtri possibili), numero di click per raggiungere la notizia di interesse (qui si suppone che le notizie siano uniformemente distribuite per ogni doppio filtraggio e che la notizia di interesse si a metà del gruppo di notizie del doppio filtraggio selezionato). Semplificando:
$$T_{tot} = T_{launcher} + T_{menu} + T_{serv} + T_{click} * (\frac{N_{serv} + F_1 + F_2}{2} + \frac{N_{news}}{F_1 * F_2 * 2}) \tag{3.4}$$
- **MHP caso migliore**: Come il caso precedente ma il numero di click pari al primo filtraggio, il numero di click pari al secondo filtraggio e il numero di click per raggiungere la notizia di interesse son tutti pari a zero in quanto a tutti gli effetti supponiamo che all' apertura del servizio si visualizzi già la notizia cercata. Semplificando:
$$T_{tot} = T_{launcher} + T_{menu} + T_{serv} + (2 * T_{click}) \tag{3.5}$$

- **MHP caso peggiore**: La sequenza di tempi è data dal tempo di caricamento del launcher, pressione tasto rosso, caricamento menu, pressione frecce fino al servizio giusto (si suppone qui pari al numero totale dei servizi in quanto il servizio scelto è l' ultimo della lista), pressione tasto ok, apertura servizio, numero di click pari al primo filtraggio (qui pari al numero dei filtri disponibili in quanto supponiamo che il filtro desiderato sia l' ultimo), numero di click pari al secondo filtraggio (qui pari al numero dei filtri disponibili in quanto supponiamo che il filtro desiderato sia l' ultimo), numero di click per raggiungere la notizia di interesse (qui pari al numero totale di notizie in quanto supponiamo che tutte le notizie siano parte di questo doppio filtraggio e quella desiderata sia l' ultima). Semplificando:

$$T_{tot} = T_{launcher} + T_{menu} + T_{serv} + T_{click} * (N_{serv} + F_1 + F_2 + N_{news}) \quad (3.6)$$

- **TXT tempo medio**: La sequenza dei tempi è data dalla pressione del tasto Teletext, caricamento della pagina 100, pressione delle 3 cifre per selezionare il servizio, caricamento della lista del primo filtraggio, pressione delle 3 cifre per selezionare il primo filtro, caricamento della lista del secondo filtraggio, pressione delle 3 cifre per selezionare il secondo filtro, attesa dell' arrivo della sottopagina contenente la notizia di interesse (in questo caso supponiamo come per l' MHP che le notizie siano uniformemente distribuite per ogni doppio filtraggio e per il doppio filtraggio selezionato la notizia di interesse sia quella a metà).

$$T_{tot} = 3 * \frac{T_{wait}}{2} + 10 * T_{click} + \frac{T_{sub} * N_{pag}}{F_1 * F_2 * 2} \quad (3.7)$$

- **TXT caso migliore**: Come nel caso precedente ma con tutti i tempi di caricamento azzerati in quanto si suppone che il carousel carichi immediatamente la pagina digitata e la notizia desiderata sia la prima sottopagina.

$$T_{tot} = 10 * T_{click} \quad (3.8)$$

- **TXT caso peggiore**: Come nel primo caso ma l' ultimo tempo viene calcolato considerando che tutte le notizie siano riferite al doppio filtraggio effettuato.

$$T_{tot} = 3 * T_{wait} 10 * T_{click} + T_{sub} * N_{pag} \quad (3.9)$$

Riportandoci al caso reale attuale abbiamo quindi impostato i seguenti valori per le simulazioni, utilizzando anche l' ottimittazione delle bitrates spiegata nel paragrafo 1.1.1:

- $F_1 = 9$ (primo filtraggio relativo al servizio Cartellone)
- $F_1 = 6$ (secondo filtraggio relativo al servizio Cartellone)

- $T_{menu} = 4$ (vincolo impostato nell' ottimizzatore di bitrate)
- $T_{launcher} = 3$ (vincolo impostato nell' ottimizzatore di bitrate)
- $N_{serv} = 8$ (numero di servizi attualmente OnAir)
- N_{news}: varia con i seguenti valori *[10, 50, 100, 250, 500, 1000, 1250, 1500, 2000, 2500, 3000, 4000, 5000, 10000]*
- T_{click}: Viene fatto variare da 0,5 a 3 secondi
- $N_{pagStandard} = 746$. Calcolando la pagina 100, per ogni servizio una media di 5 filtri per primo filtraggio, 4 per il secondo (tranne il servizio in esame che ha rispettivamente 9 e 6 filtri come già spiegato) e 100 notizie.

Partendo dal servizio implementato seguendo gli standard MHP, si può notare sia nel caso medio che nel caso peggiore, mostrati in Figura 3.14 come l' aumento del tempo totale sia influenzato sia dal numero di notizie che dal tempo di click dell' utente. In particolare più sono le notizie più il tempo di click è rilevante, questo perché la navigazione è molto interattiva e soprattutto nel caso peggiore dovranno essere premuti molti tasti per giungere alla notizia di interesse. Nel caso migliore di Figura 3.15 invece, dove l' interazio-

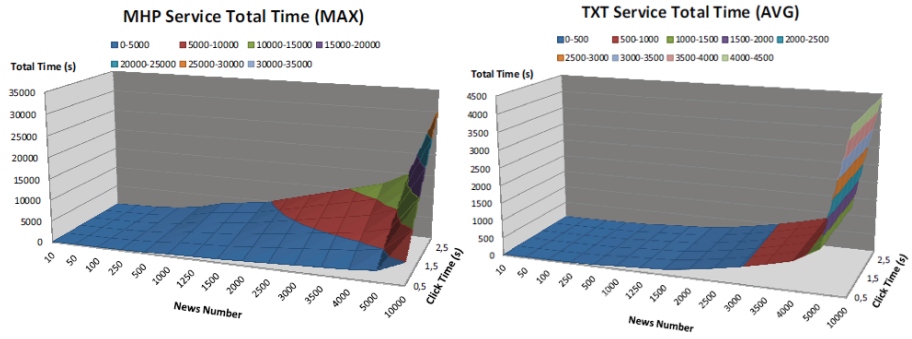


Figura 3.14: Prestazioni MHP nel caso medio e peggiore

ne con l' utente diminuisce in quanto si trova subito la notizia di interesse all' apertura del servizio, vediamo una tempistica abbastanza uniforme e non dipendente dal tempo di click quanto nei due casi precedenti. Nello specifico se fissiamo il numero di notizie ad un plausibile 3000 otteniamo un massimo di 9000 secondi nel caso peggiore (che mai si verificherà perché le notizie sono sempre distribuite in modo abbastanza uniforme tra i vari doppi filtraggi), 17 secondi totali nel caso migliore ed in media 130 secondi, poco più di 2 minuti.

Per quanto riguarda invece il Teletext, nel caso medio e peggiore di Figura 3.16 vediamo come tutto dipenda dal numero di notizie e sia praticamente

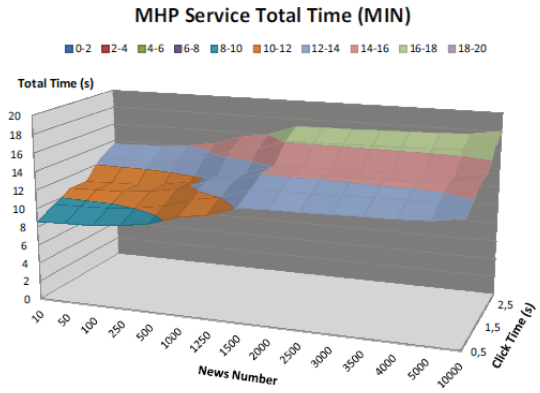


Figura 3.15: Prestazioni MHP nel caso migliore

indipendente dal tempo di click dell' utente. Infatti per il funzionamento del Teletext, l' utente interagisce solo nell' inserire per 3 volte un numero di pagina a 3 cifre (una volta per selezionare il servizio, una volta per il primo filtro, una volta per il secondo filtro) e questo a prescindere dal numero di notizie presenti. Le notizie infatti vengono mostrate in loop automatico ed il tempo che intercorre tra successiva è fissato finchè possibile a 6 secondi, tempo considerato minimo per leggere tutta la notizia, poi questo tempo incrementa all' aumento del numero di pagine. All' opposto nel caso miglio-

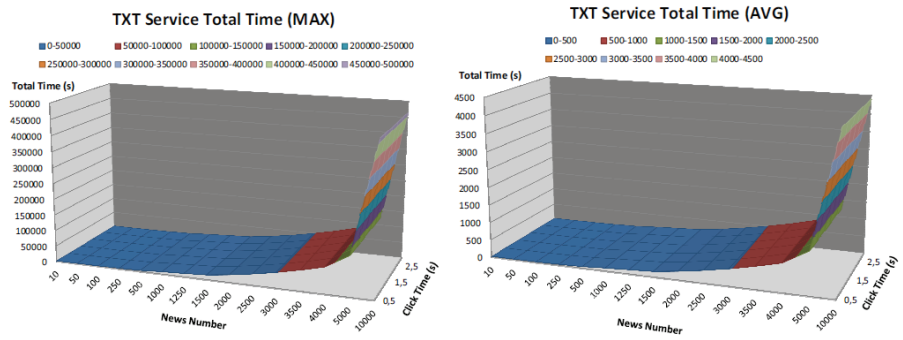


Figura 3.16: Prestazioni Teletext nel caso medio e peggiore

re, mostrato in Figura 3.17, supponiamo che il loop mostri esattamente la pagina voluta per prima, non ci sono quindi tempi di attesa e tutto dipende esclusivamente dal tempo di click dell' utente. Sempre valutando lo stesso numero di 3000 notizie come riferimento abbiamo un ingestibile massimo di 50000 secondi per il caso peggiore, 30 secondi nel caso migliore e circa 500 secondi (quasi 10 minuti) in media. Da ciò di evince quanto sia importante *sfoltire* secondo qualche criterio il numero di notizie del Teletext. Una delle nostre scelte effettuate infatti è quella di agglomerare tutto in un singolo

filtraggio, togliendo il filtro *Tutto* così da evitare notizie doppie e limitarne il numero totale perlomeno dimezzandolo.

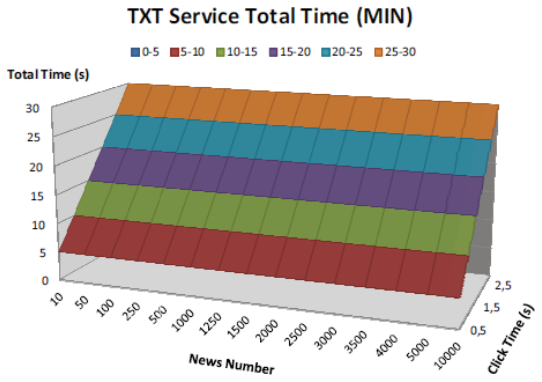


Figura 3.17: Prestazioni Teletext nel caso migliore

Da un confronto tra la soluzione Teletext e MHP, considerando per il tempo di click solo il caso migliore e peggiore, possiamo notare come fino a circa 1000 notizie le due soluzioni abbiano più o meno le stesse prestazioni, da 1000 notizie in poi il teletext è sempre meno performante nel caso migliore e medio. Il caso MHP comincia a risentire sempre di più della velocità dell' utente partendo dalle 3000 notizie in poi, precedentemente il divario è ancor meno evidente. L' unica differenza plateale si trova nel caso migliore in cui, come spiegato, il Teletext è influenzato solo dalle tempistiche dell' utente mentre l' MHP ha anche dei tempi di caricamento dei vari servizi (Launcher, Menu e servizio di interesse) per cui nel caso migliore di tempo di click il Teletext risulta più performante. Mostriamo in Figura 3.18 solo

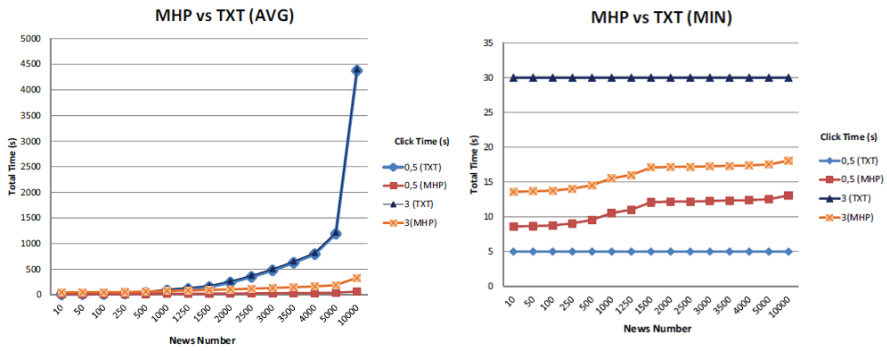


Figura 3.18: Confronto MHP e Teletext nel caso medio e nel caso migliore

il caso migliore e medio, in quanto il caso peggiore dà gli stessi risultati del caso medio ma con valori più alti.

3.5 Riadattamento dei servizi per il web

Recenti sondaggi Censis [6] rivelano che la ragione principale per cui la gente non usa internet è la paura di essere tecnicamente incompetente. Inoltre una quantità non così esigua di persone oltre 65 anni, il 20%, ammette di saper usare un computer correttamente ma non utilizza la connessione a internet perché è convinto che il web non proponga nulla di interessante per loro. I tentativi di risolvere questo problema, finora adottate, non hanno funzionato perché erano basati su un difficile e scomodo concetto di *alfabetizzazione*, in cui i giovani, con un forte orientamento tecnico, cercavano di forzare i cittadini anziani a ragionare con prospettive troppo lontane dal loro modo di essere e di pensare.

La nostra proposta invece è quella di fornire, attraverso una modalità multicanale, le stesse informazioni su un supporto diverso in modo che possano essere lette dal più ampio target di utenti possibile. Si tratta di una scelta importante al fine di superare un possibile *knowledge divide*: per chi considera la televisione un mezzo di comunicazione familiare, ad esempio le casalinghe e gli anziani, trovare sul web un servizio identico potrebbe aiutare a sentirsi più sicuri e dare buone motivazioni per usare anche il computer e internet. A causa di questi motivi entrambe le tipologie di servizi sono

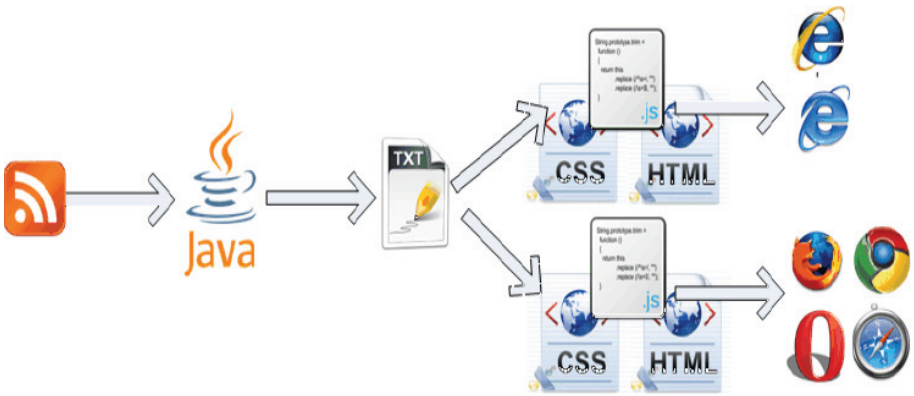


Figura 3.19: Catena relativa a un servizio riadattato per il Web

state duplicate, con lo stesso layout e le stesse funzionalità, anche su un sito web. Nella Figura 3.19 è raffigurata la sequenza di azioni necessarie, dal feed RSS alle applicazioni web finali. Il feed RSS è scaricato nuovamente e un' applicazione java memorizza le informazioni in un singolo o in più file di testo, questa volta in modo sequenziale. Questi file vengono poi tokenizzati da una seconda applicazione java che copia i titoli e contenuti in array e matrici, all'interno di un file JavaScript. In questo file sono definite anche alcune funzioni al fine di gestire la navigazione di quest' applicazione attraverso o un click del mouse sui tasti di comando remoto che sono graficamente

riprodotti all' interno del layout, o direttamente tramite la tastiera del pc. Questo file è duplicato in due cartelle: quella relativa a Internet Explorer e una seconda comune a tutti gli altri browser, in modo che il servizio possa essere correttamente visualizzato su ogni browser superando molti noti problemi di cross-browsing, soprattutto nella interpretazione dei fogli di stile CSS. In entrambe le cartelle sono compresi anche i file specifici CSS e tutte le immagini necessarie.

Sul sito web di lepidaTV sono presenti due aree apposite, una riadattata per i servizi MHP , dove tutti i servizi TV sono disponibili, e una dedicata Teletext, come mostrato nella figura 3.20. Quando un'applicazione viene scelto sul sito web, per prima cosa viene identificato il browser dell' utente, in seguito l' utente viene reindirizzato alla pagina index appartenente alla cartella corretta. I servizi sono navigabile sia attraverso il mouse che tramite la tastiera, utilizzando gli stessi numeri del telecomando per i servizi televisivi. Sempre in nome di un possibile *knowledge divide*, nella sua versione

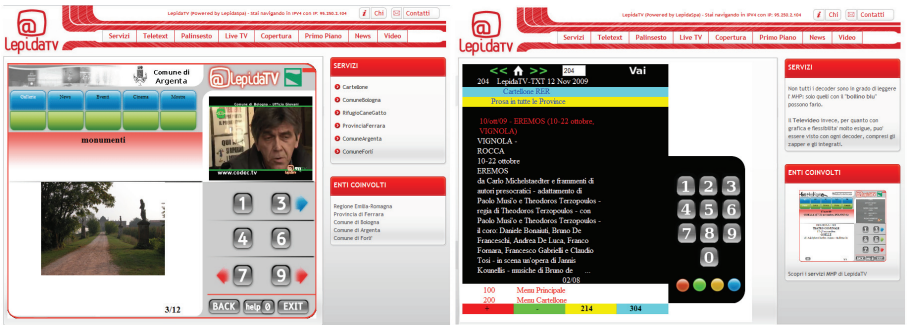


Figura 3.20: Riadattamento dei servizi sul web

web del Teletext ha anche una riproduzione grafica di un telecomando in modo da avere una navigazione identica a quella della televisione. Le tre cifre, che indicano la pagina scelta, possono comunque essere anche scritte direttamente nella label in alto, utilizzando la tastiera.

Capitolo 4

Monitoraggio ambientale

Un primo progetto sviluppato in parallelo, per la raccolta di dati di interesse provenienti dal territtorio, è stato un Centro di Gestione Dati (CGD) per il monitoraggio ambientale [49].

Una grandissima quantità di piccole reti di sensori risulta presente su tutto il territorio regionale, composta da dispositivi spazialmente distribuiti atti al monitoraggio delle condizioni ambientali, quali la temperatura, i suoni, il movimento, gli agenti inquinanti, il traffico. Solitamente sono realizzati come sistemi indipendenti ed autonomi, ognuno con una propria rete di comunicazione per il trasporto dei dati raccolti, ognuno con il suo sistema di elaborazione dei dati e ognuno appartenente ad una specifica Pubblica Amministrazione (PA) locale. Dal momento che questa situazione frammentaria porta spesso la PA locale a gestioni e manutenzioni delle trasmissioni dati inefficienti, abbiamo pensato ad un' architettura centralizzata [50] che, sfruttando la rete ibrida di accesso regionale Lepida, oltre a risolvere le inefficienze appena citate possa anche fornire ulteriori benefici, come ad esempio: evitare la duplicazione dei dati, permettere la generazione di set connessi di dati collegati tra loro e sfruttare queste potenziali correlazioni, dare la disponibilità di supervisione dei dati in tempo reale e così via.

Il modello utilizzato come linea guida per guidare l'integrazione dei sensori delle Pubbliche Amministrazioni in una singola infrastruttura si basa su alcuni elementi fondamentali:

- l' utilizzo di Lepida e ERretre come reti per le trasmissioni dati;
- diverse modalità di interconnessione verso i sensori, dirette o indirette per mezzo di datalogger proprietari;
- creazione di un Centro di Gestione dei Dati (CGD).

Il CGD dovrebbe rispettare i seguenti requisiti, mostrati in Figura 4.1: deve avere la possibilità di ricevere i dati provenienti sia dalla rete a banda larga

che dalla rete Tetra; essere in grado di ricevere dati da sensori che comunicano con protocolli diversi; essere in grado di distinguere i dati provenienti dalle reti di monitoraggio di diverse tipologie, ad esempio, distinguendo particolari fenomeni osservati e collezionati; deve poter gestire un database dove i dati devono essere conservati; elaborare dati provenienti da diversi sistemi di monitoraggio; supportare meccanismi di priorità al fine di essere in grado di elaborare in primo luogo i dati più urgenti; mostrare statistiche riguardanti i dati raccolti, rispetto a diverse variabili (scale temporali, luoghi geografici, eccetera); prevedere un' interfaccia per l' esportazione dei dati (con formati appropriati) nei confronti di altri soggetti; deve realizzare un sistema di allarme, specifico per ogni fenomeno che deve essere monitorato (questo include la definizione di soglie, la definizione dei livelli di riservatezza riferiti a queste soglie, il regolamento del meccanismo di allarme); deve essere in grado di gestire diversi sistemi per generare gli allarmi (sms, mail, SDS ovvero messaggi di testo tramite standard Tetra); deve gestire un sistema di condivisione delle risorse tra i diversi soggetti interessati ai dati, in modo da essere in grado di mostrare i dati agli utenti in base al loro profilo e privilegi assegnati (e questo denota la necessità di un sistema di autenticazione per gli utenti che accedono ai dati e di un sistema di profilazione che differenzi l'accesso ai dati, ad esempio, gratuitamente o con tariffe diverse in base al tipo di utente). Ultimo, ma non meno importante, il CGD deve essere in grado di controllare da remoto tutti i sensori, ad esempio gestendo i periodi di campionamento e la trasmissione dei dati consentendo così in ogni momento una gestione delle risorse ottimale e più armonica, limitando inoltre la necessità di riprogrammazione sul campo.

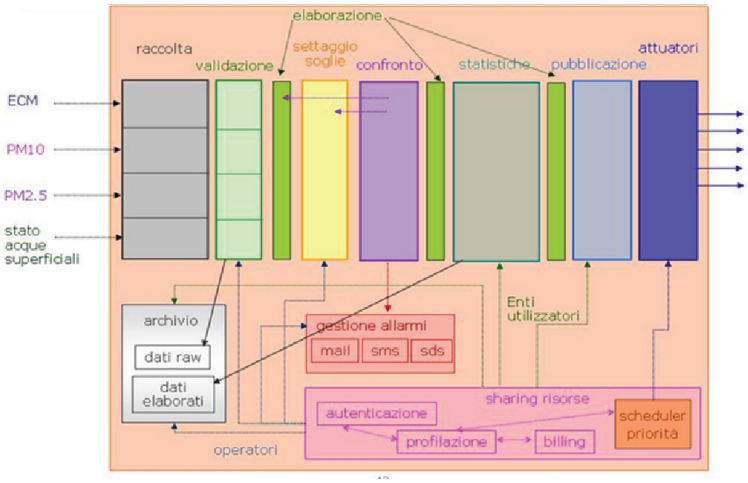


Figura 4.1: Architettura del Centro Gestione Dati

4.1 Prototipo sviluppato internamente

Un prototipo di rete di sensori che segue il modello di integrazione sopra descritto è stato quindi realizzato tramite un' architettura basata sui seguenti componenti base:

1. **BlackBox (BB)**: dispositivo per la raccolta dei dati provenienti da sensori diversi o datalogger.

2. **Driver di comunicazione** con librerie di software idonee a gestire la comunicazione dei dati su diversi canali: messaggi (SDS) e Packet Data tramite la rete ERretre, IP / UDP sulla rete Lepida).

3. **Protocollo di comunicazione**: protocollo per lo scambio dati tra Black Box e CGD.

4. **CGD**: prototipo che rispecchia l' architettura di Figura 4.1.

5. **Server Sensor Network (SN Server)**: server dedicato alla memorizzazione dei dati;

6. **Sensor DataBase Network (SN DB)**: repository per i dati ricevuti, utile per il processamento dei dati effettuato dal CGD.

Partendo da un esame della BB, questa è una semplice scheda che monta un processore ARM con sistema operativo Linux embedded e offrendo un sacco di utili porte di I/O come RS232, Ethernet, USB, GPIO. Tre situazioni diverse possono verificarsi quando si interfacciano le singole reti locali di sensori autonome con l'architettura globale ibrida, e sono schematizzate in Figura 4.2:

- **Driver IP e TETRA (a)**: una stazione di monitoraggio, fornita da terze parti, da un lato si interfaccia ai sensori e dall'altro lato alle infrastrutture telematiche più adatte, scelte tra Lepida ed ERretre, attraverso i driver di gestione idonei;

- **Gateway (b)**: una control board si interfaccia alla stazione di monitoraggio fornita da terze parti tramite un protocollo proprietario o tramite il protocollo standard Modbus. La BB, sul lato della rete di trasporto, fornisce il driver più adatto a seconda del mezzo di trasmissione che verrà scelto;

- **Interfaccia diretta (c)**: la BlackBox può interfacciarsi direttamente ai sensori e sul lato della rete svolgere le funzionalità di gateway come descritto al punto precedente.

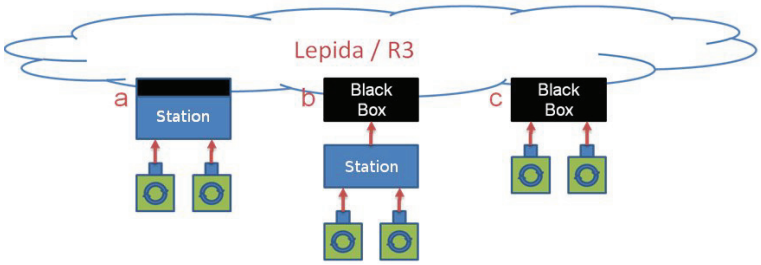


Figura 4.2: Modelli di integrazione delle reti di sensori

Grazie alla flessibilità di questo sistema siamo stati in grado di gestire i dati raccolti da un ampio gruppo eterogeneo di sensori e, come esempio della flessibilità del sistema, verranno descritti nello specifico i testbed ottenuti utilizzando diversi tipi di sensori: fonometri, spire induttive, piezometri ed inclinometri.

4.1.1 Monitoraggio acustico

L' idea di un sistema in grado di monitorare il rumore esterno è nata dalla necessità di risolvere i problemi connessi con aree urbane soggette ad elevati rumori notturni che disturbino la quiete pubblica. Nello specifico ne sono stati installati nel comune di Bologna e a Poggio Berni (RN), in questo secondo caso in un periodo estivo con concerti all' aperto. Il sistema sviluppato si propone di misurare in ogni momento il livello di rumore ma anche di azionare meccanismi predefiniti in corrispondenza del superamento di una soglia di livello massimo di rumore consentito. Il sistema è costituito da un fonometro che misura il livello di rumore e di una scheda di raccolta continua di dati. La scheda si interfaccia con un modem TETRA che invia tutti i dati raccolti al CGD. Questo servizio offre le seguenti caratteristiche interessanti:

- **AutomaticDailyReport**: un report via e-mail viene inviato a un utente specificato tutti i giorni, contenente un grafico e un tabulato che mostrano i dati raccolti dal sensore durante un periodo temporale stabilito precedentemente, relativo al giorno prima. Nel nostro caso essendo interessati ai rumori notturni sarà l' orario 23:00-07:00.

- **UserInterface**: è una pagina web, mostrata in Figura 4.3 che fornisce informazioni aggiuntive: una visualizzazione in tempo reale dei dati raccolti, in cui ogni sensore è associato ad una sorta di icona-semaforo che segnala con colori diversi lo stato del sensore (ad esempio verde se i dati da esso rilevati sono validati dalla DMC, giallo se in stato di warning, rosso in caso di allarme e grigio se risulta disattivato/non

funzionante); visualizzazione dei dati archiviati tramite tabella per con possibilità di selezionare una data di interesse; possibilità di richiedere tramite l'interfaccia un rapporto relativo ai dati archiviati ed il suo invio ad un indirizzo di posta elettronica specificato, scegliendo una tra le tre fasce orarie disponibili (giorno, sera, notte); visualizzazione della dislocazione delle BB su una mappa sviluppata tramite API di Google.

Figura 4.3: Interfaccia web per il monitoraggio acustico

- **Pannello di controllo**: si tratta di una pagina web dedicata agli operatori della PA, collegata ad una profilazione e un sistema di autenticazione, che include le seguenti caratteristiche: la creazione di soglie di allarme, l'inserimento di nuove BB e dei sensori ad esse collegati, la modifica delle impostazioni delle BB e dei sensori che sono già installati, la possibilità di abilitare/disabilitare la raccolta dei dati da remoto, la possibilità di leggere o reimpostare data e ora sulle BB, la gestione in tempo reale di allarmi scatenati da superamenti di soglia che potrebbero essere sia di un singolo sensore sia relativi ad un loro insieme arbitrario sfruttando così eventuali correlazioni tra differenti sensori.

4.1.2 Monitoraggio del traffico

Questo prototipo è stato sviluppato su richiesta del comune di San Giovanni in Persiceto (BO) per specifici incroci in cui, a 100 metri dal centro degli stessi, nell' asfalto di ognuna delle 4 strade, è posizionata una coppia di spire induttive, distanti un metro l' una dall' altra. Queste spire sono in grado di raccogliere le seguenti tipologie di dati:

- **Dati volumetrici**: Il numero di veicoli che sono passati sulla coppia di sensori durante uno slot temporale definito a priori, e la percentuale di occupazione dell' incrocio, sempre calcolato in base allo slot deciso.

- **Dati classificati**: possibile grazie all' accoppiamento delle spire, posiziona tutti i veicoli passati in otto classi di velocità diverse e in 16 classi di lunghezza diversa.

Per questa tipologia di monitoraggio la BB è usata come gateway tra il CGD e il centro di collezione dati, già installato in locale. Questo servizio può offrire le seguenti funzionalità:

- **AutomaticReport**: report realizzato per mezzo di e-mail inviate automaticamente ad uno o più indirizzi precedentemente definiti, con frequenze mensili e settimanali. In entrambi i casi vi è in allegato: un file pdf contenente la lista dei dati volumetrici raccolti dalle quattro bobine durante la settimana, un file pdf con tutti i dati raccolti e classificati da ogni spira, due file PNG con un grafico raffigurante a intervalli regolari (ad esempio ogni ora) i dati raccolti da ogni spira e che rappresentano le informazioni correlate al numero di veicoli e l'occupazione dell' incrocio, due file png con i dati di velocità e lunghezza relative a ogni spira, mostrando tramite istogramma a strati, cioè con una singola barra per ogni giornata, ogni classe con un colore diverso.

- **UserInterface**: una pagina web pubblica, mostrata in Figura 4.4, che include le seguenti funzionalità: visualizzazione in tempo reale dei dati volumetrici, ovvero numero di veicoli e la percentuale di occupazione, aggiornato ogni ora; istogramma che mostra gli ultimi N dati raccolti relativi a ciascun sensore; visualizzazione delle BB su una mappa, attraverso un marker colorato che cambia colore in tempo reale in base alle diverse fasce di valori dei i dati volumetrici (cioè un indicatore verde indica un traffico scorrevole, un marker giallo indica che è possibile trovare auto in coda, rosso significa che il traffico è congestionato); visualizzazione online dei dati archiviati specificando la data di interesse, ed inoltre possono essere generati report su dati archiviati inviati a un indirizzo di posta elettronica specificato, per uno slot temporale indicato dall' utente.

- **ControlPanel**: fornisce le stesse caratteristiche descritte per il servizio di monitoraggio acustico.

Questo servizio permette di valutare l' andamento traffico in base alle diverse fasce orarie giornaliere, ai giorni della settimana e alle stagioni dell' anno in modo da evidenziare le anomalie e le ore di punta così da poter notificare una necessaria riprogrammazione delle tempistiche semaforiche dell' incrocio. Potenzialmente la BB potrebbe attuare, come conseguenza di alcuni

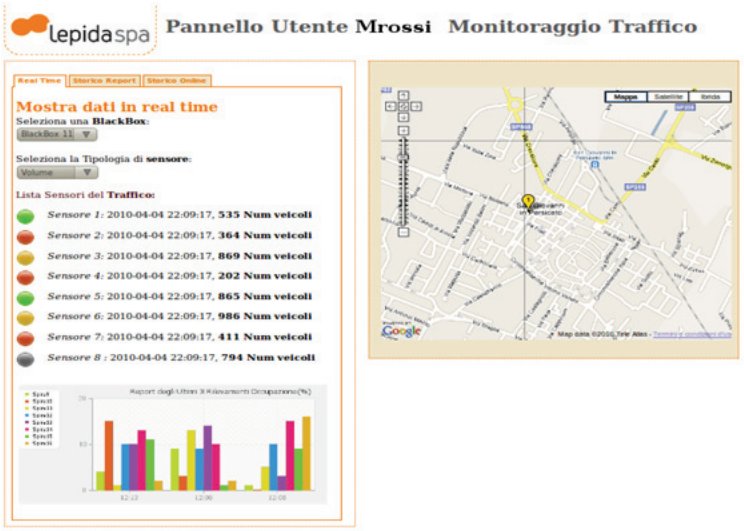


Figura 4.4: Interfaccia web per il monitoraggio del traffico

algoritmi definiti, cambiamenti in tempo reale sulle temporizzazioni di uno specifico semaforo.

4.1.3 Monitoraggio frane

Il testbed organizzato da Lepida SpA è stato in questo caso installato presso la frana di Fosso Moranda, nel comune di Polinago, provincia di Modena. Si compone di una stazione di rilevamento proprietaria (datalogger) con collegati ad esso due inclinometri biassiali posti a diverse profondità, che eseguono misure accurate relative a movimenti millimetrici del terreno, e un piezometro, che misura la pressione idrostatica. La BB è collegata ad un modem Tetra per la trasmissione dei dati, seguendo la configurazione in cui la stazione di rilevamento agisce come uno slave e la BB funge sia da master che da gateway verso la rete di trasmissione Tetra.

Il sistema è alimentato da un pannello fotovoltaico ed è normalmente spento. Ad una frequenza di campionamento programmata, di solito ogni ora, la stazione di monitoraggio si *sveglia* e controlla l'alimentazione di tutto il sistema: sia del modem Tetra che della Blackbox. La BB richiede quindi i dati provenienti dai sensori della stazione ed invia il messaggio di risposta verso il CGD e comanda la stazione proprietaria, che si occupa del controllo dell' alimentazione, di spegnere il sistema. Le comunicazioni tra la stazione proprietaria e la BB avvengono fisicamente attraverso una connessione seriale e logicamente tramite l'esportazione dei dati da entrambe le parti con lo standard di interfaccia Modbus. In aggiunta ai parametri specifici il sistema include anche il monitoraggio del livello di batteria di backup, che è

utile nel controllo del funzionamento dell'intero sistema di misura automatizzato. Tutti i dati elaborati hanno un peso ridotto, che è di circa 20 byte per ogni trasmissione. Nonostante questo il banco di prova è molto significativo perché è legato ad un luogo di installazione vero e proprio, mostrato in Figura 4.5 caratterizzato da condizioni particolarmente ostili, trovandosi in una zona isolata, senza alcuna alimentazione elettrica continua disponibile.

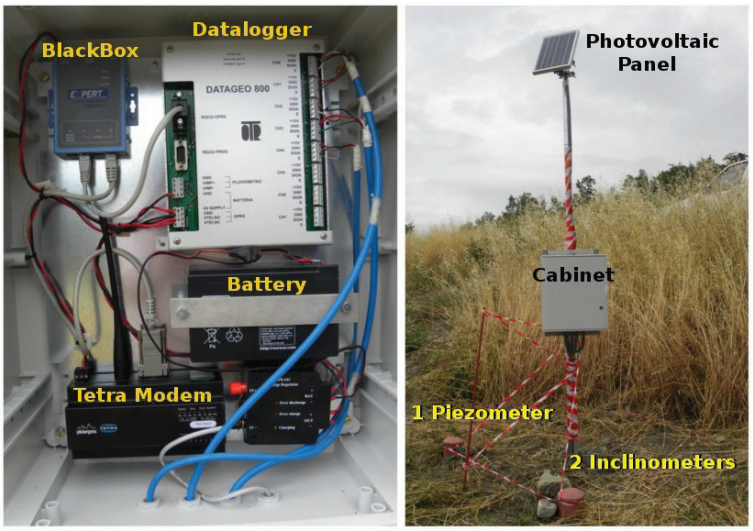


Figura 4.5: Sito di installazione per il monitoraggio frane

Le funzionalità offerte da questo servizio sono analoghe a quelle già spiegate per le altre due tipologie di monitoraggi con l'aggiunta della visualizzazione congiunta, su uno stesso grafico, dei dati rilevati da più sensori, a scelta dell' utente: questo permette di confrontare contemporaneamente le variazioni del piezometro e dei due inclinometri, così da avere più chiaro il movimento della frana.

Come conseguenza dei buoni risultati raggiunti, è stata richiesta una seconda fase di sperimentazione che dovrebbe includere tre nuove installazioni collegate a sensori multipli e un ampliamento delle funzioni della BB, come ad esempio il recupero da remoto dei file di log e la modifica da remoto della frequenza di campionamento.

4.2 Evoluzione tramite LabICT

Anche se questo sistema sviluppato internamente risultava abbastanza completo, era stato implementato con l'obiettivo di mostrare il suo potenziale per il monitoraggio ambientale e alcune funzioni erano quindi in una fase

embrionale di sviluppo. A seguito di un crescente interesse dimostrato dagli enti al termine della fase sperimentale, partendo da questa esperienza il prototipo si è evoluta in una soluzione più complessa ed efficiente sfruttando il LabICT-PA (Laboratorio per le ICT per la Pubblica Amministrazione). Il LabICT-PA, creato nel 2007 dalla RER, fa parte della Rete regionale ad Alta Tecnologia e mira ad accelerare l'innovazione nella PA. Dal 2011 LabICT-PA è anche membro della rete dei Living Labs europei (European Network of Living Labs - ENoLL) [52]. Il modello organizzativo dei LabICT-PA si basa sul concetto dei *laboratori viventi*, in cui i requisiti funzionali e le specifiche sono definite da e con gli utenti, che non sono altro che le pubbliche amministrazioni. Le diverse fasi di progettazione e test saranno effettuate anche attraverso un dialogo continuo con gli utenti finali.

I partner principali e il loro ruolo in questo living lab sono: la Regione Emilia Romagna che determina la polizia attraverso il piano per le ICT; LepidaSpA che coordina le attività e fornisce le competenze tecniche necessarie; quasi 400 azionisti pubblici di LepidaSpA che rappresentano gli utenti finali; quasi 100 partner commerciali, chiamati *club degli stakeholder Lepida*, che sono i *think tank* che creano valore aggiunto per la PA e per il mercato; infine università e istituti di ricerca in veste di partner di ricerca per il laboratorio. In questo contesto delle reti di sensori, il LabICt-PA ha creato un prototipo completamente funzionante, non ingegnerizzato, di un CGD per reti di sensori.

Questo progetto mira a integrare tutte le reti di sensori distribuite nella regione attraverso la realizzazione di una piattaforma condivisa che possa gestire in modo uniforme tutti i tipi di dati ambientali. In primo luogo, il database del prototipo precedente è stato completamente rivisto per migliorare la gestione dei dati, intesi nel nuovo CGD come una singola misura rilevata dal sensore, diventando così generalizzati al massimo. Le principali caratteristiche architetturali, sono:

- **Modularità**: ogni blocco è indipendente e comunica attraverso lo scambio di file XML.
- **Scalabilità**: ogni modulo può essere implementato su differenti macchine fisiche.
- **Configurabilità**: i principali parametri di funzionamento possono essere definiti in un database, compresa la definizione di nuove tipologie di sensori, soglie, allarmi e così via.

Tutti i sensori sono stati schematizzati in modo gerarchico in modo che più sensori possano dipendere, o meno, da una BB, che può essere a sua volta collegata ad un'altra unità, ad esempio stazioni proprietarie. Ognuno di questi elementi è classificato come un sensore, questo perché sono tutti in grado di inviare e ricevere segnali, inoltre è previsto che ogni singolo

sensore sia in grado di eseguire diversi tipi di misura con tempi diversi di acquisizione. Infine, le misure possono essere puntuali, aggregate o calcolate su una media relativa a diversi intervalli di tempo. Oltre alle tabelle dedicate alla gestione dei sensori, il database comprende anche tabelle aggiuntive necessarie per fornire indirizzi di enti e referenti, gestire ticketing, allarmi, profili, logging. Il CGD proposto risulta costituito dai seguenti moduli Open Source, schematizzati in Figura 4.6:

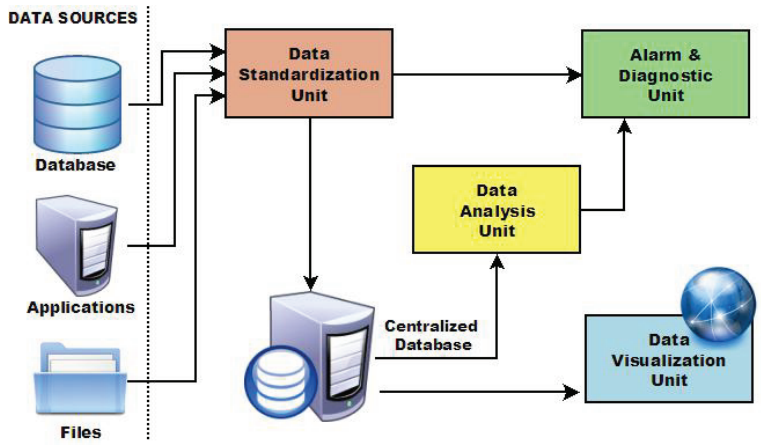


Figura 4.6: Architettura del CGD proposto dai LabICT-PA

- **Data standardization unit**: si occupa di raccogliere i dati dalle sorgenti eterogenee in realtime (sensori di differenti tipologie), interagire eventualmente con repository esterne per caricare basi dati già esistenti sulla nuova piattaforma, incapsulare il dato in un formato standard e salvarlo nel nuovo database riprogettato.
- **Data analysis unit**: deriva da quello già esistente ed implementato da LepidaSpA, aggiornato tenendo conto delle modifiche apportate ai database, e si occupa di validare/elaborare i dati provenienti da ciascun sensore, controllando se si verificano *avarie*(evento in cui si è in grado di riconosce un malfunzionamento del sensore o della centralina), *allerte* (evento in cui si è in grado di riconoscere uno stato significativo del sensore) o si raggiungono le condizioni specificate dal *simplesso* (evento che viene scatenato a seguito del raggiungimento di più condizioni di allerta come ad esempio la verifica contemporanea di più soglie, una correlazione di dati e sensori, eccetera), comunicandole poi tramite XML all' *Alarm and diagnostic unit*;
- **Alarm and diagnostic unit**: oltre agli allarmi generati dalla *Data analysis unit*, tutte le unità parte dell'architettura di sistema possono inviare messaggi di errore nel caso ci sia un malfunzionamento generico nel CGD come errori di connessione al database, query non

riuscita, modulo non funzionante. L'unità diagnostica è implementata utilizzando un web service SOAP e gestisce tutte le richieste XML in arrivo archiviandole in modo corretto. Se sono associati a una o più procedure di allarme l'unità invia un messaggio di avviso ad uno o più utenti predefiniti, tramite e-mail, SMS o SDS su terminale Tetra. Infine, l'unità gestisce generici eventi, eventualmente programmati su determinate fasce orarie o collegati a determinati allarmi, che comportano l' invio di un report e-mail contenente il grafico relativo ai sensori coinvolti, al fine di avere un feedback visivo della situazione che ha scatenato l' allarme stesso.

- **Data visualization unit**: mostrata in Figura 4.7, è basata su un sito web composto da vari form che consentono all'utente di interrogare e monitorare le strutture dati eterogenee incluse nel CGD. Tutti i form sono stati integrati in un unico portale e sono costituiti da diverse schede, disponibili nella schermata principale del sito. Una vista ad albero nella parte sinistra del sito web rappresenta tutte le stazioni di controllo ed i sensori ad esse collegati, ad ogni sensore corrisponderanno inoltre uno o più tipi di misure. Questo albero è generato in base al login iniziale: infatti è possibile definire una associazione tra un profilo e un utente, che specifichi quali sensori sia possibile visualizzare. Le icone della struttura hanno mantenuto gli stessi colori definiti nel prototipo precedente per fornire indicazioni visive sullo stato di ogni sensore. La visualizzazione ad albero consente la selezione di più componenti contemporaneamente. Una mappa geo-referenziata della regione è mostrata inoltre nella home page ed i marker indicano le stazioni installate utilizzando colori in accordo con quelli definiti per le icone visualizzare la struttura. Cliccando su un marker una descrizione della stazione e dei sensori ad essa collegati compare in un pop-up.

  Le schede ereditate dal primo prototipo sono quelle riguardanti il monitoraggio real-time, l' analisi di dati storicizzati e la piattaforma di amministrazione; è stata aggiunta invece una scheda di visualizzazione dei log.

Oltre ai sensori già inclusi nel primo prototipo, in questo caso sono stati integrati dati forniti da ARPA riguardanti sensori per il rilevamento delle polveri sottili e sensori idropluviometrici.

4.3 Integrazione con il sistema multicanale

I dati raccolti tramite questa piattaforma possono essere facilmente inseriti nel sistema multicanale sfruttando le catene di funzionamento spiegate ai paragrafi 3.1 e 3.2. Elenchiamo alcune di esse:

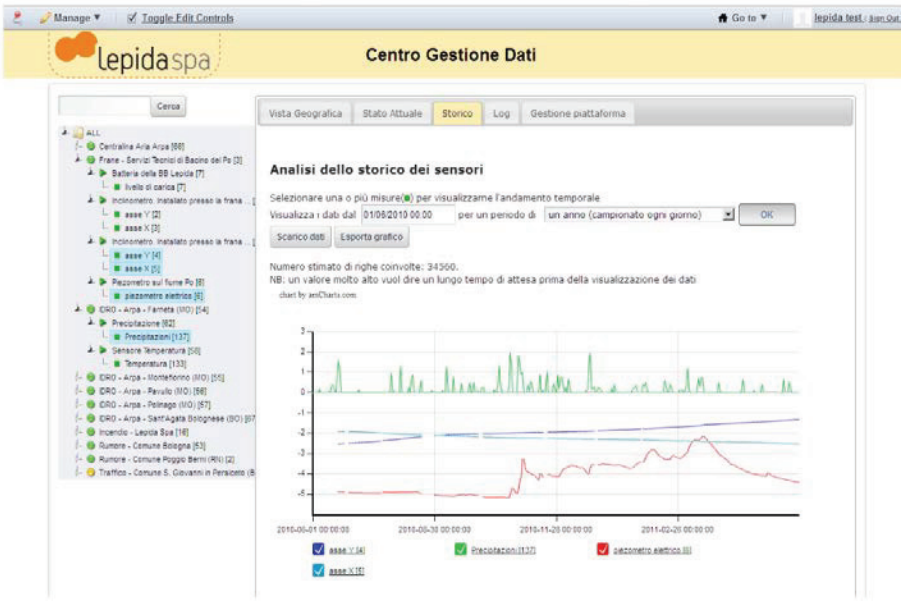


Figura 4.7: Interfaccia Web del CGD LabICT-PA

- **Servizio ad accesso condizionato**: i due file binari proposti nell' architettura generale possono facilmente essere estesi a tre file, come mostrato in Figura 4.8. La fase di login rimane la stessa, ovvero viene effettuata una ricerca dicotomica sul primo file contenente gli username seguiti dai puntatori al secondo file, dove compaiono invece le password per prime. Se il controllo della password va a buon fine si legge il numero di sensori assegnati all' utente (ricavabile dal sistema di profilazione del CGD) e di seguito l' id univoco delle misure associate seguite dal puntatore al terzo file. Nel terzo file troveremo infine il numero di informazioni associate al sensore, seguite da lunghezza e testo di ognuna di esse. Si può pensare di memorizzare in questo terzo file una lista degli allarmi avvenuti durante l' ultima settimana oppure mostrare i ticketing aperti ed il loro stato attuale per tenerne sotto controllo le evoluzioni. Tutte queste informazioni sono semplicemente reperibili in modo automatizzato da un eseguibile java che viene posto in cron con scadenze predefinite, a seconda dell' aggiornamento dei dati che si vogliono mostrare, ed esegue query verso il database del CGD creando di conseguenza i file coinvolti.

- **Servizio informativo**: Data la mole di dati a disposizione si può pensare che per gli altri utenti possa essere più importante una sorta di report settimanale o mensile su informazioni che possano essere interessanti per il cittadino. Ad esempio sarebbe utile fornire dati correlati del traffico e dei valori rilevati delle polveri sottili per comune,

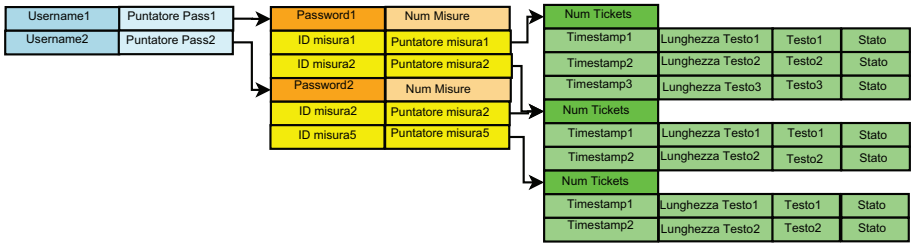


Figura 4.8: File binari di un servizio MHP collegato alle Reti di Sensori

oppure dati riguardanti le precipitazioni del mese, confrontate con le piogge dello stesso mese negli anni precedenti, e così via.

- **Teletext**: Come spiegato nel paragrafo 3.4 è importante non inserire un numero elevato di pagine all' interno del Teletext per evitare che le sottopagine abbiano un tempo di refresh eccessivamente prolungato. Si può pensare quindi di riportare su Teletext informazioni schematiche e riassuntive come il numero di sensori presenti per ogni tipologia oppure, selezionando un comune, avere la lista di sensori installati all' interno di esso.

Informazioni relative a soglie di allerta o malfunzionamenti verrebbero mostrate solo su accesso condizionato in quanto si ritene che i cittadini, non avendo la conoscenza necessaria per interpretare la reale importanza della situazione occorsa, potrebbero dar vita ad allarmismi inutili e dannosi.

Capitolo 5

Monitoraggio rete Lepida

Data una struttura così complessa ed eterogenea come quella della rete Lepida si è rivelato di grande importanza uno strumento [53] in grado di garantire una elevata flessibilità ed integrazione, assicurando un monitoraggio congiunto di reti e servizi [54] [55].

Un primo importante passo è stato quello dell'analisi dei principali problemi che un sistema di monitoraggio potrebbe causare in una così ampia area di interesse. Prima di tutto è necessario implementare un sistema che possa raccogliere informazioni eterogenee da diversi sistemi e fonti diverse, (ad esempio database, file di testo, fogli di calcolo) per poi estrapolare i dati di interesse mostrandoli in modo veloce e semplice, dal momento che deve essere in grado di fornire in tempo reale l'aggiornamento simultaneo, ad esempio con un refresh ogni minuto, di più grafici tramite una interfaccia web. Deve anche riuscire supportare un elevato numero di accessi e richieste concorrenti senza sovraccaricare il server che effettua queste elaborazioni. I dati devono essere resi compatibili rispetto alla granularità del tempo di acquisizione, in modo che sia resa possibile una loro aggregazione e l' allineamento dei rispettivi timestamp sia assicurato. In caso di dati che vengano forniti dalle loro sorgenti tramite diverse unità di misura, dovrebbe essere possibile per l'utente decidere come standardizzarli tramite la scelta di un' unica unità di misura.

Infine, mentre i parametri di interesse riguardanti la rete sono evidenti e vengono quindi mostrati il traffico in arrivo e il traffico in uscita, è stata invece necessaria una valutazione su quali parametri relativi ai servizi potrebbero risultare interessanti per i cittadini e gli enti che accedono all'interfaccia web [56] [57]. Ad oggi abbiamo individuato questi parametri come utili, rendendoli disponibili sul web:

- **Portali**: il numero di accessi utente;
- **Servizi con autenticazione**: il numero di login effettuati;

- **I servizi che comprendono una sottoscrizione di pratiche**: il numero di istanze presentate.

5.1 Architettura

Una architettura è stata quindi progettata per dotare Lepida di un sistema di monitoraggio in grado di superare i problemi che abbiamo appena elencato. La sua schematizzazione è mostrata in Figura 5.1, ed analizzeremo ora più in dettaglio ogni blocco che la compone.

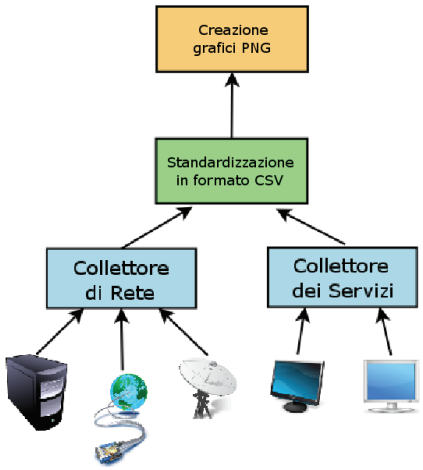


Figura 5.1: Architettura del Monitoraggio Lepida

- **Collettore di rete**: i dati relativi al traffico di rete vengono gestiti attraverso *RRDTool*: il suo scopo è quello di raccogliere dati in una serie temporale e di memorizzarli in un file RRD(round-robin database), che è semplicemente un buffer circolare che permette una mole massima di dati memorizzati costante nel tempo [58]. RRDtool assume che i dati siano variabili nel tempo all' interno di un intervallo di una certa lunghezza. Questo intervallo, di solito chiamato *passo (step)*, viene specificato al momento della creazione di un file RRD e non può essere modificato in seguito. Siccome i dati potrebbero non essere sempre disponibili al momento giusto, RRDtool provvede automaticamente ad interpolare i dati ricevuti adattandoli ai suoi step temporali interni.

  Nel caso di un' interfaccia di rete quindi, il nostro collettore legge i dati dal file RRD tramite un comando di *fetch* appropriato, specificando, a seconda dell'intervallo di tempo richiesto dall'utente, non solo il lasso di tempo, ma anche la sua risoluzione, cioè il numero di

timestamp mostrati: quindi avremo in risposta una serie di dati con una granularità appropriata. Questa query al file rrd restituisce una lista di timestamp e misure associate, scritta in un file txt che verrà ulteriormente manipolato, come spiegato in seguito.

I dati raccolti sono relativi alle diverse parti della rete Lepida che sono state illustrate nel Capitolo 2: il traffico Internet attraverso i nodi principali della rete, il traffico sui collegamenti HDSL e satellitari, il traffico wireless e il traffico specifico riguardante il Sistema Pubblico di Connettività. Inoltre sono disponibili altri dati rilevanti come il traffico IPv6, il peering verso diversi operatori, e così via.

- **Collettore dei servizi**: i servizi sono normalmente collegati ad un database che memorizza anche i dati che desideriamo visualizzare attraverso il sistema di monitoraggio. Per questo motivo, il collettore si basa su un demone che, secondo alcuni termini fissati, raccoglie i dati di interesse dal database e li scrive in un file. Anche in questo caso la raccolta dei dati viene ripetuta più di una volta, scrivendoli in file diversi, ognuno con una diversa granularità, in accordo con le scelte di periodi ora disponibili su interfaccia web.

- **Standardizzazione**: nel momento in cui un utente seleziona un grafico, che verrà visualizzato attraverso diverse chiamate Ajax [59], se si tratta di un grafico di rete per prima cosa tutti i file rrd coinvolti sono interrogati, poi i file txt generati e contenenti la lista di valori in risposta vengono letti; altrimenti, se si tratta del grafico di un servizio, viene direttamente letto il file con la granularità scelta dall'utente. In entrambi i casi le informazioni memorizzate in questi file sono standardizzate e scritte in un file in formato CSV in cui ogni riga contiene un timestamp e, separati da virgole, i dati misurati per quella data e ora specifica: due valori per le reti, traffico in entrata e in uscita, e un valore unico per i servizi. Questo file CSV standard è necessario per il prossimo passaggio di creazione del grafico.

- **Generazione dei grafici**: Quest' ultima fase di visualizzazione è basata su una libreria PHP chiamata *pChart* [60], che genera le tabelle in formato PNG, a partire da un file CSV composto da un elenco di timestamp seguito da uno o più valori, che abbiamo infatti creato nel passo precedente attraverso la fase di standardizzazione. Questa libreria è stata scelta per la sua velocità nella creazione di grafici, anche nel caso in cui vi sia una mole ingente di dati da visualizzare, e anche perché permette una notevole personalizzazione grafica offrendo numerose tipologie di grafici (lineari, istogrammi, a torta, ad area,eccetera), permettendo di cambiare la granularità degli assi, aggiungere griglie, testi, per scegliere colori e dimensioni. Come si può notare nella Figura 5.2, abbiamo deciso di utilizzare un grafico lineare per visualizzare le reti, l' immagine in alto ne è un esempio, e un istogramma per i

servizi, che permette una più facile lettura dei dati, come mostrato nella parte bassa della stessa immagine. Nel primo caso avremo sull' asse y il numero di byte al secondo come unità di misura mostrata, nel secondo caso il numero di accessi ogni 15 minuti. In entrambi i casi l' asse x è dedicato al timestamp. Vengono inoltre mostrate al di sotto del grafico il valore minimo, massimo e medio registrati nel periodo temporale selezionato dall' utente.

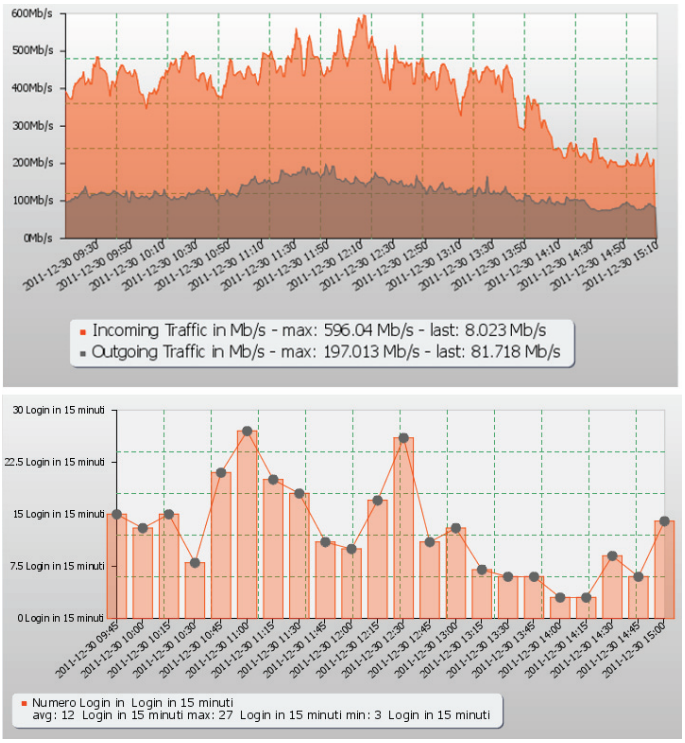


Figura 5.2: Grafici relativi a un' interfaccia di rete e a un sevizio

5.2 Interfaccia Web

L' interfaccia web parte ovviamente con una richiesta di autenticazione tramite un tipico form di login. Questo perché riteniamo opportuno personalizzare l'interfaccia a seconda di una profilazione basata sul login iniziale. In particolare, due profili diversi sono per ora disponibili:

- **Utente tecnico**: gli è consentito visualizzare i dati relativi a tutti i servizi disponibili e le interfacce di rete, singolarmente, e può inoltre aggregarli in un unico grafico;

- **Cittadino**: può visualizzare solo un numero limitato di grafici precedentemente decisi da LepidaSpA, e molti di questi grafici sono l'aggregazione di più reti, perché abbiamo supposto che un cittadino potrebbe essere più interessato al traffico globale su un nodo, rispetto ad una informazione troppo specifica. Può anche creare alcuni grafici di aggregazione, ma partendo da un numero limitato di singoli grafici disponibili.

Le funzionalità dell' interfaccia web si sviluppano su differenti schede, schematizzate in Figura 5.3. La prima scheda, che permette una scelta multipla

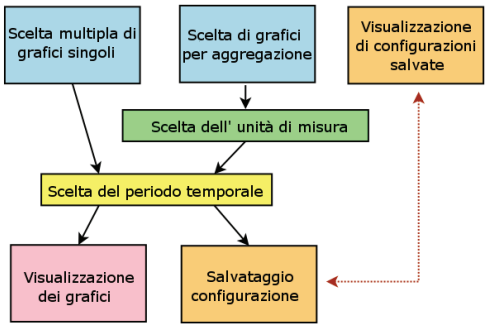


Figura 5.3: Funzionalità dell' interfaccia web del monitoraggio Lepida

di grafici singoli, è suddivisa in diversi frame, a seconda dei diversi tipi di traffico (Internet, peering, HDSL e così via), ed ogni frame comprende tutte le reti disponibili che possono essere monitorate per quella tipologia, sia come grafico singolo che sottoforma di aggregazioni precedentemente impostate. L' utente può anche fare una selezione multipla di reti o servizi, anche provenienti da diversi frame, e visualizzare l' elenco dei grafici richiesti in un' unica pagina.
La seconda scheda, che serve per scegliere i dati che vorremmo sommare in un grafico unico, mostra un elenco di interfacce di rete o servizi singoli, con a fianco relative *checkbox* che possono essere selezionate senza limitazioni nel numero di scelte contemporanee, al fine di aggregarle. In questo caso è necessario che l' utente selezioni l' unità di misura alla quale uniformare tutti i dati.
In entrambi i casi, i grafici visualizzati sono aggiornati automaticamente e l'utente può modificare il periodo di visualizzazione da un minimo di sei ore ad un massimo di un anno. Quando l' utente sta visualizzando una pagina con una serie di grafici singoli, la scelta del periodo può essere fatta sia a livello globale, impostando tutti i grafici con lo stesso periodo, o singolarmente, differenziandoli l' uno dall' altro. Inoltre, nella scheda visualizzazione, l' utente ha la possibilità di salvare la selezione di grafici singoli o aggregati appena creata. Questo viene fatto utilizzando i *cookie*, quindi è un salvataggio legato al browser, che permette però di effettuare salvatag-

gi veloci lato client in modo che il server non venga sovraccaricato da una quantità enorme di configurazioni da salvare e recuperare continuamente. Il cookie viene creato con un testo contenente per prima cosa il codice della scheda da salvare(monitor o aggregato), seguito dalla lista di codici di identificazione univoci assegnati in modo statico, a livello di programmazione, a ogni interfaccia di rete o servizio scelto.
L'ultima scheda di questa interfaccia web è semplicemente utilizzata per mostrare tutte le configurazioni salvate. Quando la scheda viene aperta tutti i cookie presenti sul client vengono controllati, alla ricerca delle parole chiave *monitor* o aggregato nel loro testo. Tutti i cookie trovati rispondenti a queste specifiche sono elencati in un menu a discesa e, selezionandone uno, questo può essere eliminato o caricato, e in caso di caricamento viene recuperata la successiva lista di codici dei grafici coinvolti e la pagina di visualizzazione precedentemente salvata è finalmente ricostruita.

5.3 Caching dei grafici

Dopo una prima fase di pubblicazione del servizio è stato deciso di introdurre un sistema di caching dei grafici. Infatti per rendere visivamente significativi i dati di periodi lunghi come 6 mesi o un anno, devono essere effettuati RRD fetch mensili successivi agganciati l'uno all'altro così da avere una granularità più fine in cui i dati non vengono aggregati in una media temporale troppo ampia mostrando dati inevitabilmente falsati. Questa problematica, unita al fatto che, a prescindere dal periodo selezionato dall' utente, tutti i grafici mostrati venivano ricalcolati ogni minuto, portava un carico di calcolo al server eccessivo, oltre a mostrare rallentamenti lato client per la visualizzazione dei periodi più lunghi. La differenza tra il flusso logico iniziale (senza caching, rappresentato dalle frecce nere) e attuale (con caching, rappresentato dalle frecce rosse) della visualizzazione di un grafico viene mostrato in Figura 5.4. Quando viene richiesto un grafico per la visualizzazione o si cambia il periodo di visualizzazione, per prima cosa viene controllato se già esiste quel grafico per quel periodo, in caso non sia mai stato precedentemente richiesto viene creato, se invece esiste viene controllata la data e ora di ultima modifica e si controlla se il tempo trascorso è maggiore a quelli proporzionalmente così predefiniti:

- **Periodo 6h, 6 ore**: validità del file pari a 5 minuti;
- **Periodo 12h, 12 ore**: validità del file pari a 10 minuti;
- **Periodo 1d, un giorno**: validità del file pari a 20 minuti;
- **Periodo 1w, una settimana**: validità del file pari a 140 minuti, ovvero 2 ore e 20 minuti;

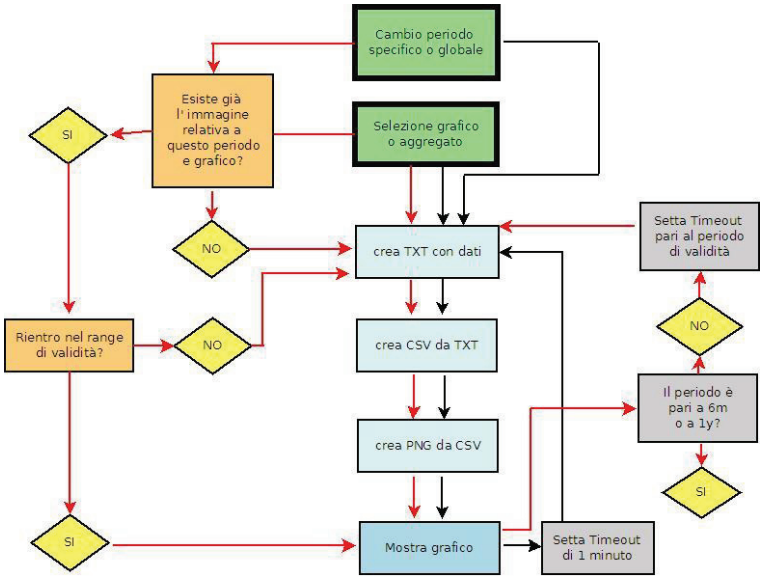


Figura 5.4: Flusso logico del sistema di monitoraggio Lepida

- **Periodo 2w, 2 settimane**: validità del file pari a 280 minuti, ovvero 4 ore e 40 minuti;
- **Periodo 1m, un mese**: validità del file pari a 560 minuti, ovvero 9 ore e 20 minuti;
- **Periodo 6m, 6 mesi**: validità del file pari a 3360 minuti, ovvero 2,33 giorni;
- **Periodo 1y, un anno**: validità del file pari a 6720 minuti, ovvero 4,67 giorni;

Le stesse tempistiche vengono rispettate anche per il refresh, tranne nel caso di periodo pari a 6 mesi o un anno in cui non viene settato un timeout: si lascerebbe un processo appeso per 2 o 4 giorni e al momento non si ritiene che abbia senso che un browser venga lasciato aperto sul grafico così a lungo. Si può in ogni caso facilmente modificare questo settaggio, come anche ovviamente la tabella delle validità.

5.4 Integrazione con il sistema multicanale

Possiamo notare che questo sistema di monitoraggio permette di avere sempre disponibili file CSV contenenti i dati per ognuno dei periodi selezionabili ed inoltre immagini PNG già elaborate in cache. Risulta quindi molto semplice ottenere:

- **Servizio di Visita Guidata**: può facilmente essere riadattato per mostrare due gallerie, una per il traffico di rete, dove vengano visualizzati, per ogni frame del sito di monitoraggio, il grafico dell' aggregato di tutte le interfacce di quella tipologia e una successiva descrizione dei dati che si sono visualizzati, mettendo in evidenza i valori minimo, medio e massimo registrati per il periodo di tempo. L' analogo in una seconda galleria dedicata ai servizi. Anche in questo caso si può seguire la catena già spiegata per questo servizio, scaricando ogni notte le immagini presenti sul server. E' necessario introdurre, dato il sistema di caching, un controllo sulla validità delle immagini che potrebbero risultare non aggiornate se non sono state richieste da interfaccia web ed in quel caso scatenare il processo di creazione del grafico.

- **Teletext**: Pur non potendo mostrare grafici si può mantenere la stessa trasparenza anche attraverso questa tecnologia ricavando dai CSV due pagine di report. In quella dedicata al traffico potrebbero esservi indicati i valori minimo, massimo e medio rilevati per le ultime 24 ore, l' ultima settimana, l'ultimo mese e l' ultimo anno. Corrispondentemente per i servizi verrà mostrato il loro grado di utilizzo nelle stesse fasce temporali.

Capitolo 6

Portale regionale OpenData

L' *open data* è strettamente collegato a sua volta al concetto di *open government*, principio in base al quale la PA dovrebbe essere *aperta* verso i cittadini, tanto in termini di trasparenza quanto di partecipazione diretta al processo decisionale, anche attraverso il ricorso alle nuove tecnologie dell'informazione e della comunicazione. Come anticipato nel Capitolo 2 tra i quattro nuovi diritti di cittadinanza digitale individuati nel PiTER (Piano telematico regionale) 2011-2013 è presente anche il *il diritto di accesso ai dati* inteso come una serie di interventi relativi agli open data per garantire trasparenza e valorizzazione dei dati in possesso, gestiti e mantenuti dalle PA.

I concetti di *dato pubblico* e riuso dell'informazione pubblica vengono sanciti dalla **Direttiva 2003/98/CE** [61] che fornisce la definizione di *documento*, o informazione pubblica come rappresentazione di atti fatti o informazioni raccolti e in possesso di Enti pubblici. Definisce inoltre il *riutilizzo* come l' uso di questi dati da parte di persone fisiche o giuridiche a fini, commerciali o non commerciali, diversi dallo scopo iniziale per il quale il documento che lo rappresenta è stato creato.

La direttiva incoraggia inoltre gli Enti pubblici a rendere disponibili i documenti secondo queste modalità:

- creare indici online dei documenti disponibili per favorire l' accesso e la diffusione;
- definire condizioni eque, adeguate e non discriminatorie da applicare al riutilizzo dei dati.

Tra le attività programmate dal PiTER evidenziamo la presenza del progetto *Open Data Emilia-Romagna*, che ha appunto come obiettivi principali quelli che sono gli elementi caratterizzanti di un insieme di dati *aperto*: i dati aperti devono essere indicizzati dai motori di ricerca; i dati aperti devono

essere disponibili in un formato aperto, standardizzato e leggibile da una applicazione informatica per facilitare la loro consultazione ed incentivare il il loro riutilizzo anche in modo creativo ; i dati aperti devono essere rilasciati attraverso licenze libere che non impediscano la diffusione e il riutilizzo da parte di tutti i soggetti interessati.

6.1 Architettura

Il portale Open Data della RER [62], mostrato in Figura 6.1 è stato realizzato tramite il riuso del software del portale degli Open Data della Regione Piemonte [63], nell'ambito di una collaborazione interregionale per il riuso e l'evoluzione condivisa con un protocollo d'intesa siglato tra le due regioni che prevede, tra gli obiettivi, la *realizzazione di progetti per il trasferimento di soluzioni informatiche realizzate dalle parti mediante il ricorso a modelli di riuso.*Lo stesso accordo prevede anche una evoluzione futura congiunta dei due portali, permettendo quindi anche uno scambio di informazioni ed esperienze su aspetti organizzativi e regolamentari tra le due regioni. Nello

Figura 6.1: Homepage del portale Open Data della Regione Emilia Romagna

specifico CSI-Piemonte ha messo a disposizione di LepidaSpA, che ha realizzato il portale su richiesta RER, le competenze tecniche acquisite nello sviluppo del portale oggetto di riuso. Il portale si basa sull' utilizzo del Content Management System (CMS) Joomla [64], come mostrato dalla loro architettura riportata in Figura 6.2. Al Core di Joomla è stato aggiunto un

componente ad hoc, chiamato RD, che si occupa dell' inserimento dei dati e del loro raggruppamento in categorie, che a breve illustreremo. Entrambi si appoggiano ad un DB MySQL per la memorizzazione dei dati. Sempre da CSI-Piemonte è stato inoltre implementato un motore di ricerca, a se stante, che analizza una volta al giorno i nuovi dati inseriti nel portale e li indicizza per poter mettere a disposizione dell' utente una ricerca avanzata degli stessi. L'ETL è una componente esterna utilizzata da Regione Piemonte per alimentare la base dati dai cataloghi di metadati già presenti: è possibile comunque inserire gli stessi dati in un form disponibile sul portale accedendo all' area riservata o al backend di Joomla. Vengono infine utilizzate due componenti da Google: *Analytics* per le statistiche di visite al portale e *reCAPTCHA* per l' antispam nei commenti e nelle mail. Le

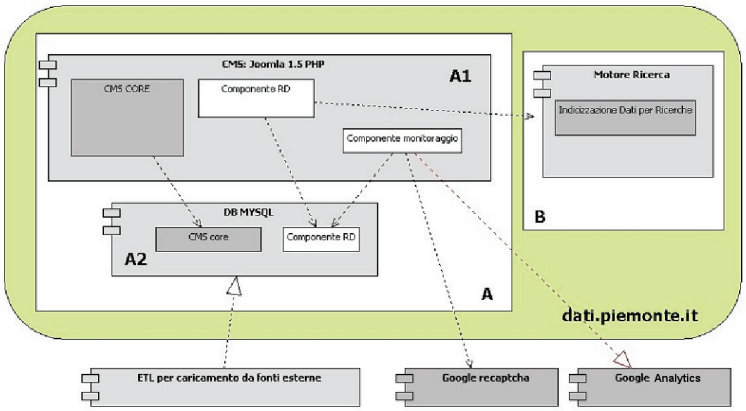


Figura 6.2: Architettura del portale messo al riuso da Regione Piemonte

principali funzionalità del portale possono invece essere così riassunte:

- **Blog**: pagina dedicata alle novità specifiche del portale, come ad esempio la notifica di modifiche o di nuovi set di dati aggiunti, o riguardanti il mondo degli Open Data, dagli aspetti legislativi, alle esperienze provenienti da altre regioni o stati esteri, agli eventi dedicati a questo tema;

- **Faq**: svariate sezioni sono dedicate alle spiegazioni del concetto degli Open Data, di come i dati possano essere riusati a seconda delle licenze con cui sono stati pubblicati, riportando inoltre tutte le normative coinvolte in questi ambiti;

- **Statistiche**: mostra gli accessi al sito tramite Google Analytics e altri dati statistici come i dati maggiormente scaricati, i dati maggiormente votati, la lista dei dati classificati per argomento, i dati classificati in base all'organizzazione o ente responsabile, tutti scaricabili anche in formato standard CSV;

- **Dati**: Il punto focale del portale sono ovviamete i dati open messi a disposizione. Questi vengono indicizzati da un programma esterno, sempre implementato da Regione Piemonte, che li categorizza e classifica a seconda delle informazioni che l' Ente inserisce nell' *Area Riservata* del portale, al momento dell' upload del pacchetto dati. Vediamo, con l' esempio degli Edifici della RER mostrati in Figura 6.3, come vengono presentati i dati da scaricare.

Figura 6.3: Esempio degli Open Data riguardanti gli edifici della RER

Innanzitutto viene mostrata la tipologia di licenza d' uso associata ai dati, tipicamente Creative Commons CC-0 [65] o CC-BY [66], ovvero per un libero riutilizzo con o senza citazione della fonte dei dati, che deve obbligatoriamente essere accettata per permettere lo scaricamento del dato.
Le informazioni sul dato contengono una breve descrizione del contenuto del pacchetto in oggetto, mostrano le parole chiave associate (molto importanti per una corretta indicizzazione e correlazione dei contenuti per future ricerche), l' Ente che ha raccolto le informazioni, gli argomenti coinvolti (suddivisi nelle macrocategorie: Ambiente e meteo, Sociale, Turismo e tempo libero, Risorse naturali, Info scientifica e ricerca, Politica, Cultura, Info geografiche, Economia, Traffico e trasporti, Agricoltura, territorio e pesca, Legali, Formazione, ICT e società dell'informazione), il tipo di dato messo a disposizione (tra le opzioni: Base dati, Carta, Collezione, Dato, Dato alfanumerico, Dato geografico, Documento, Indicatore, Servizio informativo). E' disponibile infine un' area commenti per gli utenti.

L' interesse per i dati aperti è notevole e ne abbiamo avuto riscontro sia dal numero di accessi e download che dai numerosi feedback di enti e utenti ricevuti in questi pochi mesi. Nonostante l' iniziale diffidenza dei detentori dei dati nel diffonderli tramite licenze aperte, abbiamo potuto constatare grazie a questo progetto una graduale apertura verso questa tematica ed una grande collaborazione da parte degli enti.

6.2 Integrazione con il sistema multicanale

Questo portale si è rivelato innanzitutto un collettore anche per i dati derivanti dagli altri progetti. In particolare il monitoraggio Lepida non deve far altro che affidarsi a un demone che raccoglie i CSV mensili ogni primo giorno del mese e crea un archivio incrementale contenente il mese nuovo e messo a disposizione all' url di download segnalata sul portale degli Open Data. I dati vengono diffusi con diverse tipologie di formato, quelli più diffusi sono in SVG per i cartografici ed in CSV per dati statistici. Quelli della seconda tipologia sono ovviamente quelli che possono essere integrati facilmente nel sistema multicanale. Ricordiamo infatti che tramite la libreria pChart illustrata nel Capitolo 5 lo stesso demone in cron che si occupa di effettuare il download del pacchetto può generare facilmente ed in automatico dal CSV il grafico associato. Alcuni esempi di dati che potrebbero essere mostrati sono:

- **Siti della Pubblica Amministrazione**: mostra una analisi dei siti web delle PA locali in Emilia-Romagna, suddivisa secondo differenti fasce di popolosità di un comune, espressa in percentuale su differenti indici interessanti come ad esempio la percentuale di trasparenza, multilinguismo, organizzazione del sito, usabilità, presenza di multicanalità, accessibilità.
- **Open source nella Pubblica Amministrazione**: mostra le percentuali di diffusione ed utilizzo di software open source negli Enti Locali, specificando quale sia la percentuale di utilizzatori consapevoli e quanti inconsapevoli e suddividendo l' indagine in diverse branche di software, ad esempio browser, office automation, posta elettronica, sistema operativo, eccetera.
- **Interattività e completezza servizi on line della Pubblica Amministrazione**: analogamente ai siti della PA sono qui state effettuate indagini sui servizi on-line usando, come indici in percentuali, interattività, personalizzazione, accessibilità e facilità d'uso.

I tutti questi esempi i dati possono essere mostrati in modo testuale, sia tramite servizi informativi che Teletext, oppure seguendo la metodologia delle visite guidate alternando ad esempio istogrammi, ricavati dai CSV in automatico, e descrizione degli stessi.

Ulteriori pacchetti contenenti moli più ingenti di dati, come ad esempio le serie storiche dal 1988 riguardanti la *Popolazione per sesso e anno di età* suddivise per comuni e anni di vita, necessiterebbero inevitabilmente di aggregazioni di dati prima di una loro integrazione multicanale.

Capitolo 7

Contenuti audio/video e multicanalità

Certamente LepidaTV non è una televisione generalista, e non deve competere con la radiodiffusione televisiva italiana,non prevede quindi né pubblicità, né film di intrattenimento commerciale, ma si occupa piuttosto della valorizzazione di contenuti audio/video, prodotti in Emilia-Romagna territorio regionale e relative alla regione stessa [67]. Ad oggi sono presenti più di 1600 contenuti, suddivisi nelle seguenti macrocategorie: arte e spettacolo, scienze e tecnologia, attualità e informazione, cultura, formazione e scuola, storia, sociale, ambiente, economia e sviluppo, sigle e spot, società e costume. Questa valorizzazione aumenta grazie all'importanza che abbiamo dato alla modalità multicanale anche riguardo agli audio/video che sono proposti dal palinsesto televisivo, 7 giorni a settimana, 24 ore al giorno. Non solo i programmi televisivi sono trasmessi in tutto il territorio regionale, ma sono anche in simulcast su un sito web, anche nel caso di streaming di eventi live. Infine, tutti i prodotti video memorizzati nel nostro archivio sono sempre disponibili attraverso una modalità on demand, su un sito web, grazie ad una liberatoria perenne, valida per tutta la Community Network regionale.

7.1 Digitale terrestre

Grazie ad un gruppo di emittenti locali LepidaTV può raggiungere circa il 90% della popolazione RER, tramite la loro trasmissione broadcast. Tutti i contributi audio/video raccolti nel corso di questi ultimi anni, sono localizzati su un server dove uno script di shell controlla ogni ora se sono stati caricati nuovi audio/video e, se questo è vero, questi file vengono elaborati in successione seguendo la catena di processi che ci accingiamo a spiegare.

Per prima cosa è necessario verificare se il file è danneggiato e, se risulta invece corretto, ci assicuriamo che sia il flussi audio che il video siano

presenti al suo interno, procediamo quindi nell' elaborare questi due flussi separatamente.

Molti test sono stati eseguiti, nella prospettiva di utilizzare strategie di compressione mpeg2 che possano assicurare una qualità elevata anche se si ha una larghezza di banda molto ristretta a disposizione, ovvero di 4 Mb, così suddivisa:

- 2.7 Mb per il flusso video,
- 188 Kb per l'audio flusso,
- banda rimanente utilizzata per le tabelle, servizi MHP e Teletext.

La nostra conversione finale in un file con estensione *.m2v* si basa su tre principali comandi bash in pipe tra loro. La prima parte è un ffmpeg [68]:

```
ffmpeg -i $fifoTS -async 1 http://<your IP address>:
<your port>/example.ffm
```

Questo semplicemente imposta la *frame rate* e mantiene inalterato l' *aspect ratio*, in base alle proprietà del file di input.
Il secondo è un altro ffmpeg:

```
ffmpeg -i - -vhook "/usr/local/lib/vhook/
imlib2.so -i $logo" -f yuv4mpegpipe - |
```

che pone il logo di LepidaTV RTITV in sovraimpressione. La terza e ultima parte è invece la più critica:

```
mpeg2enc --verbose 0 --aspect 2 --frame-rate 3
--video-bitrate 2500 --interlace-mode 0
--reduction-4x4 1 --reduction-2x2 1 --video-buffer 500
--force-b-b-p --video-norm p --keep-hf --
sequence-header-every-gop --min-gop-size 6
--max-gop-size 15 --cbr --multi-thread 4
-o $outputfile.m2v
```

Questa volta abbiamo scelto di utilizzare il tool *mpeg2enc* [69] invece di ffmpeg perché ffmpeg non consente una bit rate costante e, soprattutto, i nostri test hanno dimostrato che con l' utilizzo di mpeg2enc, a parità banda, siamo in grado di ottenere risultati di qualità migliore, anche visivamente. Alcune scelte strategiche sono: impostare a 1 il parametro *reduction* in modo che ben pochi blocchi vengano scartati, permettendo una migliore qualità anche se la codifica risulta più lenta; per codificare più informazioni ad alta frequenza viene utilizzato il flag *–keep-hf*; viene forzata la selezione

della GOP size ad assicurare 2 fotogrammi B tra fotogrammi adiacenti I/P, perché diversi decoder comuni non riescono a gestire flussi in cui compaiano meno di 2 fotogrammi B tra I/P frames. Una volta che abbiamo ottenuto il file m2v, questo è infine codificato in un Program Elementary Stream, ovvero un file .pes.

```
esvideompeg2pes video.mp2 > video.pes
```

La codifica audio risulta più semplice, e viene effettuata usando ffmpeg:

```
ffmpeg -i $inputfile -ac 2 -vn -acodec mp2 -f mp2
-ab 128k -ar 48000 -y $outputfile.mp2
```

Quindi anche questo file .mp2 in uscita dall' ffmpeg viene trasformato in .pes tramite i tool di Opencaster.

```
esaudio2pes audio.mp2 [sampler per frame][sample rate]
[es frame size] -1 [first pts] > audio.pes
```

Quando questo processo di conversione si conclude, viene inserito un record in un database specificando la durata del video e il suo codice di identificazione, oltre ad altre informazioni quali titolo, regista, operatore, sinopsi e la categoria. L'elenco dei programmi TV, di solito generato una volta alla settimana, si basa su una lista di numeri di identificazione relativi a contenuti audio/video.

Come mostrato nella parte superiore della Figura 7.1, tre file FIFO sono presenti sul server: attraverso ognuno di essi un file pes viene trasmesso, secondo l'elenco di ID specificato nei programmi televisivi e, infine, questi tre FIFO sono codificati , di volta in volta, in un TS e multiplexati nel carousel finale che viene trasmesso in broadcast. Il primo flusso FIFO è interrotto dopo uno slot temporale pari alla durata del primo video, e il processo di trasmissione inizia così a codificare il secondo file FIFO. Quando comincia la codifica della terza FIFO, il primo viene alimentato dal quarto video della lista, e così via a rotazione. La trasformazione da PES a TS avviene grazie al comando

```
pesvideo2ts [video pid] [frame per second]
[half vbv] [ts bit rate] 0 video.pes
> video.ts
```

```
pesaudio2ts [audio pid] [sampler per frame]
[sample rate] [es frame size] -1 0
audio.pes > audio.ts
```

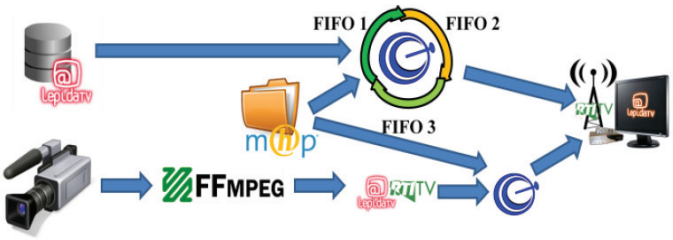


Figura 7.1: Catena dello streaming broadcast di LepidaTV

Questo viene fatto per entrambi i flussi audio e video e questi due flussi, come già spiegato, sono multiplexati con gli altri contenenti il loop delle pagine di Televideo, e tutti i servizi MHP interattivi.

Vengono trasmessi inoltre alcuni eventi dal vivo, come ad esempio il Porretta Soul Festival che si svolge ogni anno nel mese di Luglio. In questo caso si seguirà la catena che viene visualizzata nella parte inferiore della Figura 7.1. Il flusso è pubblicato su http e questo viene agganciato da un processo che lo codifica opportunamente usando ffmpeg, come abbiamo già spiegato in dettaglio, il logo viene sovrapposto e, infine, il flusso viene trasformato in un TS attraverso i tool di OpenCaster, multiplexato con gli altri flussi e mandato in onda. Tutte queste conversioni dei file sono eseguite in tempo reale e rendono possibile, utilizzando esclusivamente software Open Source di avere una trasmissione in diretta di ottima qualità audio/video con un ritardo massimo di un minuto.

7.2 Web

Sia nel caso di una trasmissione standard del palinsesto televisivo sia in caso di live tv, lo stesso flusso viene reso disponibile anche su un sito web, grazie alla catena di processi illustrato nella Figura 7.2. Una volta che il flusso audio/video viene convertito in un flusso di trasporto, questo è manipolato da uno strumento OpenCaster, chiamato *tsdoubleoutput*, che prende come input un file FIFO e semplicemente lo duplica creando due FIFO identiche come output. Successivamente, una FIFO, già multiplexata con servizi e Televideo, andrà in onda in broadcast, come spiegato in precedenza, e l'altra, di cui verranno decodificati solo audio e video, è utilizzata dal *feed* creato da *ffserver*, sempre usando ffmpeg, attraverso il seguente comando:

```
ffmpeg -i $fifoTS -async 1 http://<your IP address>
:<your port>/example.ffm
```

Dove il parametro *async* è utile se vogliamo garantire che il flusso audio sia sincrono rispetto al video, specificando il numero di campioni audio al

secondo che potrebbero essere modificati duplicandoli o scartandoli. l' impostazione di *-async* al valore 1 è un caso particolare in cui viene corretto solo l'inizio del flusso audio senza ulteriori successive correzioni. *Ffserver* è

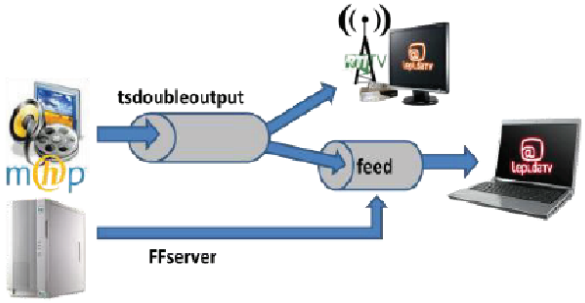


Figura 7.2: Catena dello streaming simulcast di LepidaTV

lo strumento che permette di pubblicare un flusso audio/video in una pagina web. Il passaggio più importante è la modifica del suo file di configurazione *myffserver.conf*, in particolare:

- La descrizione del feed dove vorremmo effettuare lo streaming, in questo caso un flusso di tipo FFM;

```
<Feed example.ffm> File /tmp/example.ffm
 FileMaxSize 200M </Feed>;
```

- La definizione di che tipo di flusso verrà inviato al feed, in questo caso un file FLV che ha *bitrate*, *frame rate* e *size* specificate attraverso i seguenti parametri:

```
<Stream example.flv> Feed example.ffm Format flv
AudioCodec mp3 VideoCodec flv AudioBitRate 24
AudioChannels 1 AudioSampleRate 22050 VideoBitRate 200k
VideoBufferSize 1000 VideoFrameRate 25 VideoSize 320x240
VideoGopSize 30 VideoQMin 3 VideoQMax 31
</Stream>
```

Infine, Flowplayer [70] viene utilizzato come player Open Source di filmati FLV e questo può venire facilmente integrato in pagine web, come mostrato in Figura 7.3, contenente lo screenshot della pagina dedicata al simulcast dei programmi televisivi, sul sito di LepidaTV.

Al fine di avere il doppio di streaming, in onda e su un sito web in simulcast, ci si deve quindi attenere alla seguente procedura:

- far partire il processo ffserver con il comando **ffserver -f** / <**absolute path of the file**> / **myffserver.conf**;

Figura 7.3: Simulcast di LepidaTV attraverso il suo sito web

- avviare il processo standard di OpenCaster che, al termine di tutte le conversioni, dopo la multiplazione in un flusso unico, duplicherà questo flusso in due FIFO;
- mandare uno di loro in input al feed ffserver tramite il comando ffmpeg precedentemente descritto;
- inserire FlowPlayer in un pagina web ed agganciarlo al flusso HTTP generato.

Questa catena di processi permette di avere sempre un perfetto sincronismo tra il flusso in broadcast e quello in simulcast sul web, anche quando i programmi TV vengono interrotti per dare la linea ed un evento dal vivo.

Quando lo script di shell, che abbiamo spiegato nel paragrafo precedente, controlla se nuovi file audio/video sono stati caricati e avvia la conversione per ottenere il flusso di trasporto finale, un processo parallelo trasforma gli stessi file in un altro formato, usando questo comando ffmpeg :

```
ffmpeg -i $inputfile -s 608x336 -vcodec libx264
-g 250 -bf 3 -b_strategy 1 -vb 250k -acodec libfaac
-ar 22050 -ab 56k -f flv -y $outputfile.f4v
```

Un file F4V è stato così creato, specificando la risoluzione finale con *-s* ed anche bit rate e frame rate, forzando una codifica video H264 e una codifica audio AAC. Questi video F4V sono memorizzati su un server e sono sempre disponibili a richiesta sul sito web di LepidaTV, come mostrato nello screenshot di Figura 7.4: gli utenti possono fare ricerche in base a titolo,

descrizione, categorie o leggere i programmi TV nell' apposita sezione contenente il palinsesto settimanale, per poter facilmente scoprire quale video viene trasmesso in questo momento e poterlo vedere di nuovo sul sito web.

Figura 7.4: Esempio di ricerca e visione di un video OnDemand di LepidaTV

7.3 IPTV

L' IPTV è una convergenza di comunicazione basata su IP ed il broadcasting la cui situazione attuale è una sorta di circolo vizioso: l' industria ha bisogno dell' IPTV per giustificare l' investimento nella banda larga, ma non c'è al contempo una buona IPTV senza banda larga [71] [72]. Tuttavia, l'investimento è un modo per sopravvivere per alcune aziende di telecomunicazione. Si sostiene che l' IPTV sarà comunque la killer application per la prossima generazione di Internet e le NGN. L' IPTV non deve essere confusa con la semplice trasmissione di video su una rete IP, ma comprende ulteriori servizi avanzati fortemente orientati verso le decisioni dell' utente e le sue scelte personali tra cui l'accesso al web e alle mail, il video on demand (VoD), memorizzazione di contenuti che permetta di avere funzioni di controllo quali pausa, avanti, indietro durante la loro visione e così via, utilizzando una doppia via di comunicazione interattiva tra operatori ed utilizzatori.

Tra le diverse sfide della IPTV troviamo l'integrazione di operatori diversi con diverse infrastrutture e back-office, la stabilità di lungo termine, e soprattutto risulta fondamentale garantire un' ottima qualità del servizio (QoS), rispetto ai seguenti parametri:

- **QoS per il video**: comprendono jitter, il numero di pacchetti fuori sequenza, probabilità di perdita di pacchetti, probabilità di guasto di rete, tempo di join a un gruppo multicast, ritardo, eccetera.
- **QoS per voce**: comprendono Mean Opinion Score (MOS), jitter, ritardo, tasso di perdita di pacchetti voce, e così via.
- **QoS per servizi IPTV**: includono la disponibilità di un canale, tempo di apertura di un canale, tempo per un cambiamento di canale, il tasso di fallimento nel cambiamento di canale, ed altro.

Prendiamo un esempio di IPTV: con l' attuale tendenza verso la distribuzione di segnali TV ad alta definizione (HDTV), ognuno dei quali richiede circa 1-2Mbps, e il desiderio degli utenti di un elevato numero di canali (tipicamente sui 200-300), ci deve essere un meccanismo efficiente di fornire un segnale di 1-2Gbps congiuntamente ad un gran numero di utenti remoti. Se una sorgente deve consegnare un segnale a 1 Gbps, ad esempio, verso un milione di ricevitori trasmettendo tutta la banda disponibile attraverso la rete principale, sarebbe necessaria una rete che supporta un traffico di un petabit al secondo e questo non è attualmente possibile. Al contrario, se la sorgente potesse inviare l' 1 Gbps di traffico verso 50 punti di distribuzione remoti, ognuno dei quali poi fa uso di una rete di distribuzione locale per raggiungere 20.000 iscritti, la rete principale ha bisogno di solamente 50 Gbps rispetto a prima, e questo è possibile con una corretta progettazione della rete. Per queste ragioni, l' IP multicast è visto come una tecnologia in grado di conservare la banda, e che ottimizza la gestione del traffico tramite la distribuzione simultanea di un flusso informativo ad un vasto gruppo di destinatari, compresi sia gli utenti aziendali che i clienti residenziali.

Un principio importante per l' IP multicast è quello di permettere al ricevitore di aggregarsi, ovvero effettuare un *join*, a nuovi flussi di informazioni; inoltre, un secondo principio è la capacità di supportare una *potatura*, ovvero un *pruning*, ottimale in modo tale che la distribuzione di un contenuto venga semplificata portando la replicazione dei dati il più vicino possibile al ricevitore. Questi principi consentono un utilizzo efficiente della banda delle infrastrutture di rete sottostanti. Nello specifico IGMP (Internet Group Managing Protocol) è il protocollo utilizzato dagli host IPv4 per comunicare ai router multicast gli stati di appartenenza ai gruppi multicast. IGMP viene utilizzato per registrare dinamicamente singoli host/ricevitori su una particolare sottorete locale (ad esempio, LAN) ad un gruppo multicast. Con IGMPv2, inoltre, un host può inviare un messaggio di *leave group* per avvisare il router che non è più parte di un gruppo multicast; questo permette

al router di effettuare un *pruning* all' interno della lista del gruppo per cancellare questo host prima che una richiesta successiva sia programmata, minimizzando così il periodo di tempo durante il quale sono trasmesse in rete trasmissioni non necessarie.

Vediamo ora nello specifico dove e come usare il multicast nelle reti di prossima generazione.

Nella Figura 7.5 vediamo un esempio di un' applicazione che viene erogata secondo una architettura generale che prevede due punti di distribuzione di contenuti, due core network di distribuzione centrali, dall' emittente alle DSLAM, e le reti di distribuzione locali, dalla DSLAM agli utenti, con indicati possibili protocolli multicast da utilizzare nelle varie parti. Partendo dalle *Core Network* i protocolli multicast che possono essere usati

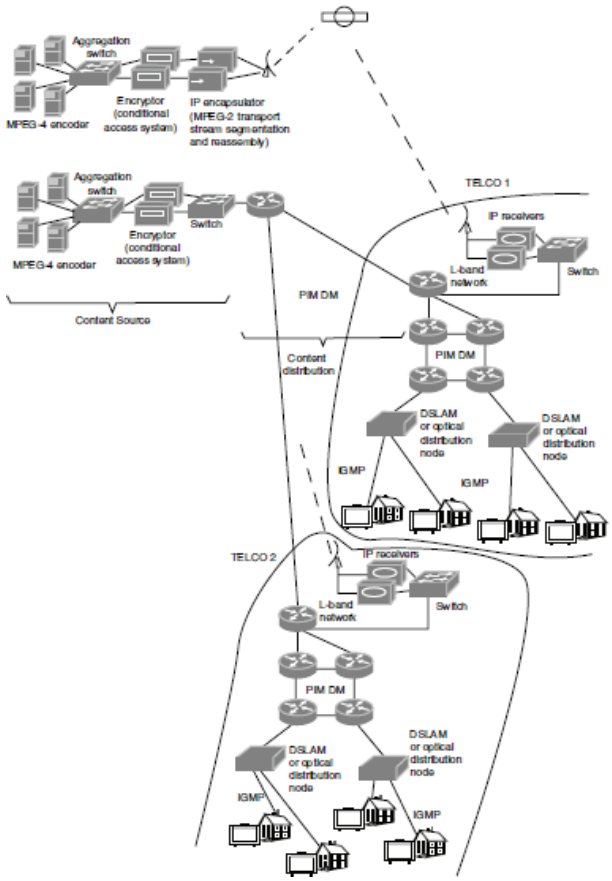


Figura 7.5: Esempio di sistema multicast per IPTV

sono diversi:

- **PIM-SM o PIM-SSM**: usano meccanismi di tipo pull per la distribuzione multicast, i riceventi devono essere attivi ed aver fatto esplicite

richieste IGMP per ricevere il traffico. La differenza tra i due protocolli è che l' SSM necessita che l' edge router, ovvero il router sul confine della rete, conosca l' IP della sorgente multicast, nell' SM non è necessario poichè la corrispondenza viene chiesta a un RP. In caso la popolazione di utenti attivi sia molto minore della popolazione totale questa soluzione è preferibile.

- **PIM-DM**: usa un meccanismo push. Molto efficiente nel caso della IPTV dove si suppone che ci sia almeno un ricevente attivo in ogni sottorete (tipicamente ogni DSLAM necessita di ricevere tutti i pacchetti multi cast). L' efficienza è dovuta al fatto che evita il continuo traffico di richieste IGMP e anche la necessità di un RP che diventerebbe un punto critico in caso di rottura.

- **P2MP LSPs**: Soluzione meno dinamica dei PIM che si basa su MPLS, ovvero utilizza delle label inserite nei pacchetti IP in transito per definire un successivo instradamento a *commutazione di etichetta*. Dal lato utente è sempre invece necessario che i ricevitori supportino l' IGMP per informare la rete che un ricevitore sia o meno membro di un particolare gruppo.

Passando all' *Access Network* distinguiamo invece le principali soluzioni per le due tipologie: Ethernet e fibra ottica.

- **Ethernet**: Si basa semplicemente su VLAN e vi sono due diverse architetture possibili [73]:

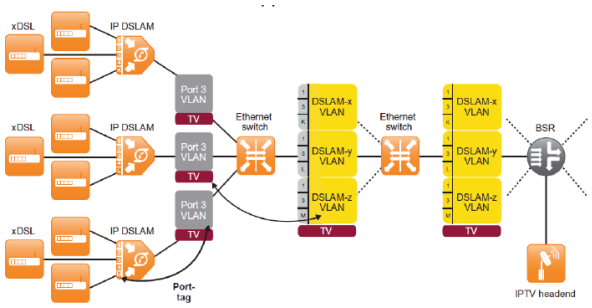


Figura 7.6: Architettura VLAN per subscriber

  - **VLAN per subscriber (1:1)**: Come mostrato in Figura 7.6 il subscriber è in questo caso al centro del sistema. La VLAN prevede una connessione alla rete per distribuire molteplici servizi. L' aggiunta di un nuovo servizio non necessita di riconfigurazione per la VLAN. Per la scalabilità si utilizzano VLAN a stack: quella interna identifica la porta del subscriber sulla DSLAM, quella

esterna identifica la DSLAM. Una ulteriore speciale VLAN multicast viene posta sul Broadband Service Router o presso un altro IP edge router. Sia DSLAM che switch supportano l' IGMP.

– **VLAN per service (N:1)**: L' obiettivo è ottimizzare la distribuzione dei diversi servizi e sono quindi incluse VLAN specifiche per i servizi dati e voce. Solitamente questa VLAN è posta sul link DSL. La schematizzazione è presentata in Figura 7.7.

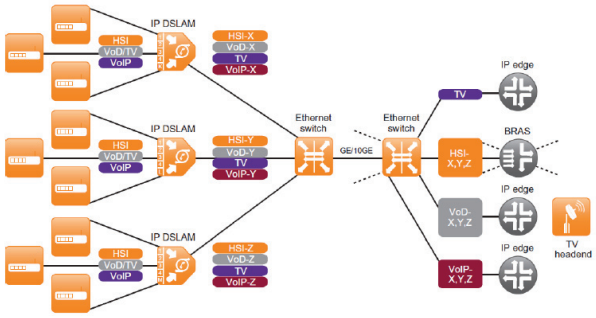


Figura 7.7: Architettura VLAN per service

- **Reti ottiche**: In questo caso il multicast si basa sulla costruzione di più *light trees* utilizzando la tecnologia WDM. Ogni ligth path è un circuito ottico tra due nodi della rete. I vincoli principali da rispettare sono: se i nodi intermedi non hanno la possibilità di convertire le lunghezze d' onda è quindo necessario che tutto il light path sia sulla stessa lunghezza per tutto il cammino (*wavelength continuity*); due o più light paths che attraversano la stessa fibra devono essere su lunghezze d' onda diverse per non interferire l' un l' altra. Il light tree, che verrà di seguito trattato, è una generalizzazione del light path che connette un nodo a più nodi della rete [74].

Gli switch ottici si basano su due tipologie di architetture:

- **Switch opachi**: I bit ottici che arrivano al nodo sono convertiti in dati elettronici, switchati tramite un cross connettore elettronico e poi riconvertiti nel dominio ottico. In Figura 7.8 *(a)* si vede come il segnale D sia convertito in 3 segnali elettronici, uno dei quali è rilasciato in locale e gli altri due convertiti in ottici su link 1 e link 2.

- **Switch trasparenti**: Non vi sono conversioni del segnale. Nella Figura 7.8 *(b)* uno vediamo come il segnale che debba essere replicato venga passato a uno splitter e amplificato prima di essere trasmesso su più link in uscita tramite un secondo switch ottico. Un miglioramento può essere dato dalla Figura 7.8 *(c)* in cui viene utilizzato un unico switch richiedendo però più porte disponibili sullo stesso. Un

vantaggio di questa architettura è che il fanout di un segnale non è più limitato dalla tipologia di splitter in quanto l' uscita di uno splitter può diventare l' ingresso di un secondo splitter riuscendo così ad avere un più alto grado di splitting. Anche in quest' ultima figura il segnale da splittare è passato allo splitter X, amplificato da Y e torna allo switch ottico per essere ritrasmesso su link 1 e link 2.

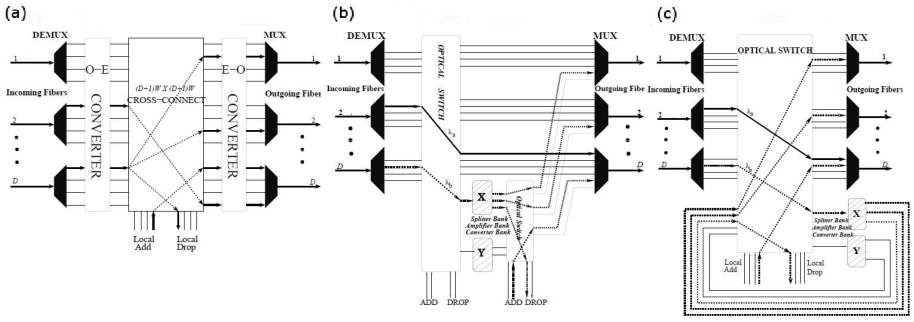


Figura 7.8: Architetture switch ottici

Su queste topologie sono stati formulati modelli matematici sia nel caso siano presenti convertitori di lunghezza d' onda nei nodi, nel caso non siano presenti, nel caso siano previste sessioni multicast che operino a velocita piu basse di quella di un canale a singola lunghezza d' onda e nel caso in cui vi sia un limite nel grado di splitting, mostrandone i limiti e i casi di irrealizzabilità. Sono poi state applicate diverse euristiche in queste 3 topologie applicando una MPH (minimum path heuristic) con una lista dei nodi successivi random, in ordine ascendente/discendente di costo per stabilire la connessione e in ordine ascendente/discendente di numero dei nodi destinazione. Le migliori performance si ottengono nei casi dei liste ordinate in modo discendente.

Risulta inoltre molto importante un' analisi sulla *protezione delle sessioni multicast*: anche in questo caso sono stati valutati piu sistemi ed euristiche, schematizzate nella loro interezza in Figura 7.9 e analizzate nei rami più interessanti:

- **Arc-disjoint light-trees**: Nel caso di fallimento su singola fibra. Si fornisce un light tree di backup che risulti disgiunto per gli archi coinvolti rispetto al primo. Da notare che stessi link possono essere coinvolti nei due tree ma solo se usati in sensi opposti.
- **Self-sharing light-trees**: Nel caso di fallimento su singola connessione. Due soluzioni possibili: o si protegge ogni segmento dell' albero indipendentemente tramite segmenti di backup che possono anche coincidere con segmenti primari o di backup per diversi alberi, o si offre

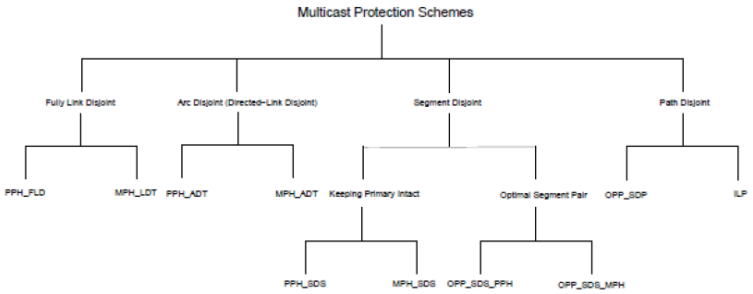


Figura 7.9: Classificazione delle protezioni delle sessioni multicast

una *path-pair protection*, ovvero una coppia di percorsi, dove si trova un cammino di backup disgiunto dal primario. Confrontando diverse euristiche (OPP_SDP Optimal Path-Pair based Shared Disjoint Path, PPH Pruned Prim' s Path Heuristic, MPH Minimum-cost Path Heuristic, ILP Modello Matematico) con LDT per Link Disjoint tree e ADT Arc Disjoint tree, sono stati ottenuti i risultati mostrati nei grafici di Figura 7.10. Si può osservare che MPH_SDS ha un costo consistentemente più basso di PPH_SDS per tutte le dimensioni di gruppo e questo è dovuto al fatto che il costo dell' albero primario è maggiore nel caso MPH (infatti MHP_ADT si può vedere dal grafico di sinistra ha una maggiore probabilità di blocco). Inoltre, OPP_SPD risulta migliore di tutti gli schemi *segment-disjoint* come prestazioni anche se la sua performance è molto legata alla sequenza con cui vengono selezionati nell' algoritmo i nodi di destinazione. OPP_SPD permette di ottenere risultati molto vicini alla soluzione ottima ottenuta dalla risoluzione matematica dell' ILP, per tutte le dimensioni dei gruppi multicast analizzati.

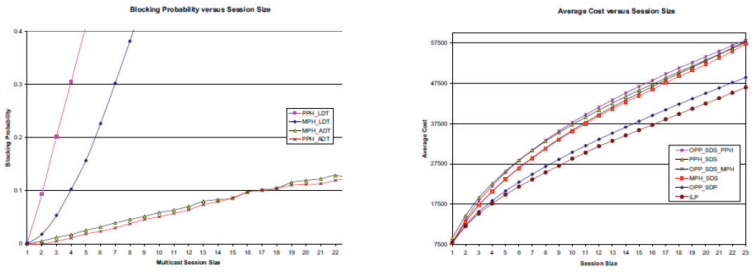


Figura 7.10: Confronto euristiche Self-sharing light-trees

- **Cross-sharing light-trees**: Studio della condivisione ottimale delle risorse di backup (delle due tipologie appena spiegate) per molteplici sessioni multi cast in una rete ottica. Il modello utilizzato per determi-

nare il numero di backup previsti è un link-vector modificato. In conclusione sono stati comparati questi tre approcci, come si può vedere da Figura 7.11. Questo approccio è stato utilizzato sia per reti grandi

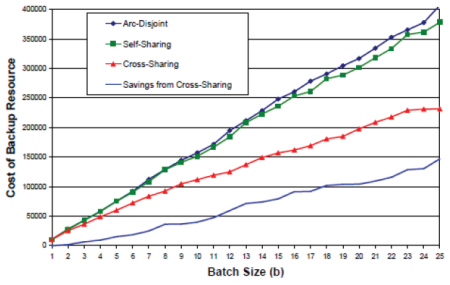


Figura 7.11: Comparazione dei tre approcci di protezione delle sessioni multicast

che piccole ed ha prodotto significativi risparmi nel costo di risorse di backup, rispetto agli approcci self-sharing e arc-disjoint. Nello specificano risultano ingenti, fino al 39%, nel caso di reti a livello nazionale e per gruppi grandi, ad esempio con 25 sessioni contemporanee.

Capitolo 8

OpenBOXware

Il progetto OpenBOXware [75] è stato promosso dall' Associazione culturale NeuNet, in collaborazione con il Corso di Laurea di Informatica Applicata e con la Sezione di Scienze e Tecnologie dell' Informazione del Dipartimento di Matematica Fisica e Informatica dell' Università di Urbino. OpenBOXware è un piattaforma open-source per lo sviluppo di applicazioni multimediali bandwidth-aware la cui caratteristica principale è la capacità di gestire flussi multimediali in streaming provenienti da fonti eterogenee verso dispositivi sia locali che remoti. Questa capacità di gestire pipelines multiple simultanee in concomitanza al loro streaming verso oggetti remoti rendono questa piattaforma particolarmente adatta ad essere installata non solo in corrispondenza del punto finale di ricezione, ma anche su nodi intermedi di una rete di distribuzione di contenuti, come può essere la rete IP. Infatti il progetto è strettamente connesso al concetto di *reti neutrali*, a causa dell' oggettivo ruolo trainante che la televisione via internet ha nello sviluppo delle reti di nuova generazione. La neutralità della rete è un principio secondo cui i fornitori di connettività (ISP) e gli altri operatori della rete devono occuparsi di trasportare le comunicazioni degli utenti fino a destinazione senza discriminarle, cioè senza privilegiare nè filtrare, rallentandole, alcune applicazioni o alcuni contenuti in base alle loro convenienze. La normativa italiana non ha ancora preso una posizione decisa in merito alla neutralità, tuttavia gli utenti hanno diritto alla trasparenza, cioè a sapere se il proprio operatore mette in atto qualche tipo di restrizione al traffico internet.

8.1 Architettura

La piattaforma OpenBOXware è stata testata su sistemi Linux utilizzando i seguenti componenti: Mono 2.6, un'implementazione open source del framework [76] .NET che fornisce l'astrazione richiesta rispetto all'hardware sottostante e ai componenti software; GStreamer 0.10.30, per lo sviluppo

del sottosistema multimediale; Qt 4.5 come libreria .NET necessaria per lo sviluppo di una interfaccia utente basata su Qyoto. Il suo stack software è stato scelto in modo da avere la massima flessibilità. La sua architettura, mostrata in Figura 8.1, è basata sui seguenti elementi: il *kernel* che ha il compito di gestire l' avvio e successivamente delega le varie funzionalità ai moduli esterni; una *interfaccia*, basata su API, che non è altro che un modulo astratto che espone le funzionalità comuni del framework ai moduli esterni; una *skin* che è un particolare modulo che si prende carico di visualizzare l' interfaccia grafica. Essendo un modulo esterno è personalizzabile e modificabile liberamente da utente, sviluppatore o distributore e cambiandolo, è possibile diversificare l' esperienza d'uso, pur mantenendo la compatibilità con tutte le altre applicazioni.

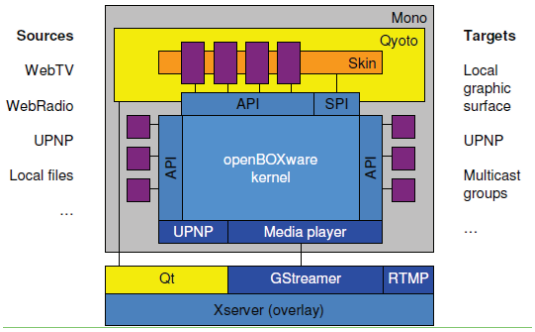


Figura 8.1: Architettura della piattaforma OpenBOXware

Lo stack software è stato scelto in modo tale da massimizzarne la portabilità e la flessibilità, pensando anche a come facilitare al massimo gli sviluppatori software nella creazione di *plugin*, ovvero una serie di moduli che possano le funzionalità di openBOXware e che possano essere liberamente installati dall' utente tramite un *application store* online. Sono tre le principali categorie di plugin che possono essere sviluppate:

- **Applicazioni**: si basano su un'interfaccia grafica tramite cui l'utente può interagire. Possono essere iniziate e terminate a runtime, sia in modalità schermo intero che utilizzando una modalità in versione *barra laterale*. In entrambe le modalità, l' applicazione è in grado di mostrare elementi GUI complessi sfruttando la potenza delle librerie Qt. Gli esempi in questo ambito spaziano dalle utility, ai lettori di notizie online, ai videogiochi. Ogni applicazione può essere eseguita a schermo intero, nella barra laterale, in un widget personalizzato, o in background. Le modalità di esecuzione relative ad una data applicazione vengono dichiarate nel suo *manifest*, che è un file XML che fornisce al framework le informazioni necessarie per presentarlo agli utenti finali e per caricare tutti i componenti necessari. Tutte le applicazioni, incluse quelle in esecuzione in background, possono anche

fare uso delle API per interagire con un utente locale (tramite notifiche o finestre di dialogo) o con un utente remoto o un'applicazione (esponendo servizi web/UPnP).

- **Media Sources**: espongono un albero di risorse multimediali navigabile dall' utente. I contenuti sono forniti al framework possono essere acceduti anche da altre componenti aggiuntivi (ad esempio un'applicazione) o dal framework stesso.
- **Media Targets**: espongono un dispositivo multimediale al resto del framework. Ogni installazione openBOXware ha un *media target* locale che visualizza il video sullo schermo di default e riproduce l' audio sul dispositivo di output predefinito. Ulteriori media target possono collegare al framework ulteriori dispositivi remoti, centri media UPnP in rete locale e così via. Il framework sarà quindi in grado di trasmettere flussi multimediali a qualsiasi media target di destinazione disponibile.

8.2 Integrazione servizi ed audio/video

Prima della la presentazione della piattaforma al pubblico e alla stampa e la pubblicazione dei sorgenti del codice, a Ottobre 2010 un numero limitato di programmatori ha avuto l' opportunità di analizzare il codice di openBOXware [78], testare esempi base delle principali funzionalità supportati degli sviluppatori della piattaforma, al fine di contribuire con nuovi plugins per l' *application store*. In quest' ambito è stato implementato anche un widget contenente due plugin per OpenBOXware collegati a LepidaTV, mostrati in Figura 8.2, al fine di ampliare ulteriormente il concetto di multicanalità anche a questa piattaforma emergente. Per prima cosa nel file *manifest* vengono specificati titolo, icona e descrizione del plugin, indicando inoltre il nome della classe principale dell' applicazione:

```
<addin xmlns:xsi="http://www.w3.org/2001/XMLSchema
-instance"xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <name>LepidaTV</name>
  <description>Simple streaming application with custom
  media pipeline.</description>
  <author>Elisa Benetti</author>
  <version>1.0</version>
  <assemblyName>LepidaTV</assemblyName>
  <iconPath>lepidatv.png</iconPath>
  <application qualifiedName="LepidaTV.PipelineApplication">
   <gui>
   <fullscreen maintainSystemBar="true"/>
   </gui>
```

Figura 8.2: Widget di LepidaTV sviluppato per OpenBOXware

```
  </application>
</addin>
```

In questa classe principale viene gestita la parte di interfaccia grafica, usando Qyoto, e vengono iseriti due plugin all' interno del widget:

```
var w = new PlayerWidget(container, this);
w.Show();
var c = new FeedReader(container, this);
c.Show();
```

Il primo plugin consiste in un player in cui viene creata una pipeline personalizzata, un sub-widget viene specificato come target video e gli altoparlanti locali come target audio, infine viene creato un media target, partendo dai due specifici per audio e video.

```
//subwidget target video
_monitorTV = new QWidget(this);
_monitorTV.SetFixedWidth(6*(parent.Width()/11));
_monitorTV.SetFixedHeight(2*(parent.Height()/3));
_layout.AddWidget(_monitorTV);
//creazione dei target
var videoTarget = context.MediaPlayer.
```

```
EnableWidgetAsVideoMediaTarget(_monitorTV);
var audioTarget = context.MediaPlayer.
  CreateLocalAudioTarget();
var mediaTarget = MediaTarget.Create(videoTarget,audioTarget);
//creazione pipeline
_pipeline.SetMediaTarget(mediaTarget);
```

Quando l'utente clicca sul pulsante play è generato un *media element*, a partire dallo stesso flusso http che viene usato per la visualizzazione simulcast di LepidaTV, come illustrato al Capitolo 7.

```
const string FileUrl = "http://www.lepida.tv:8080/
  lepidatv.flv";
var mediaElement = new MediaElement(
new HttpMediaResource {
Uri = new Uri(FileUrl)
},
MediaElementContent.FlvContainer
);
```

Il flusso può essere arrestato e riavviato ed è inoltre disponibile un controllo di livello audio per l'utente, attraverso un cursore.

Il secondo plugin è un semplice lettore di feed RSS: i tag standard vengono interpretati a partire da <*item*> e visualizzando titolo e data di pubblicazione delle notizie in un box collocato al di sopra del player. E' inoltre previsto un pulsante di refresh per forzare il ricaricamento del feed.

```
const string FEEDURL = "http://twitter.com/statuses/
user_timeline/97672934.rss";
_channel = (from c in rssDoc.Descendants("channel")
select c).FirstOrDefault();
var items = (from i in _channel.Elements("item")
let date = i.Element("pubDate")  select i);
foreach(var item in items){
//estraiamo il titolo
var title = item.Element("title").Value;
//estraiamo l'url
var link = item.Element("link").Value;
var newElement = new MyStandardItem(title,link);
```

Tramite questo widget abbiamo quindi la possibilità di diffondere i contenuti di LepidaTV e riadattare le notizie dei servizi informativi anche su questo ulteriore canale comunicativo basato su una piattaforma emergente, tutt'ora in continua evoluzione.

Capitolo 9

Conclusioni e sviluppi futuri

Partendo dalle nuove potenzialità date dalla Televisione Digitale, lo scopo primario di questa tesi è stato lo studio di come adattare contenuti audio video ed informativi per questo strumento che risulta da sempre il più conosciuto e diffuso nelle famiglie italiane. La conseguenza logica è stata l' estensione ad un concetto di multicanalità che portasse gli stessi servizi anche su altri mezzi di comunicazione, uniformandoli per facilitarne l'utilizzo nel passaggio da un media all' altro. Grazie allo switch-off al digitale terrestre avvenuto nella nostra regione a Novembre 2010 è stato possibile effettuare una sperimentazione reale dei servizi implementati attraverso LepidaTV, canale digitale che copre approssimativamente il 90% della Regione Emilia Romagna.

Cominciando dalle reti *wireless*, ovvero nel nostro caso il broadcasting, abbiamo innanzitutto analizzato la struttura del flusso di trasporto per il DVB-T per illustrare quindi gli standard sia per l' MHP che per il Teletext mostrando passo a passo come creare il flusso finale tramite un software open source.

Ritenendo l' IPTV una possibile futura killer application con l'introduzione di canali ad alta definizione e 3D, sono state parallelamente analizzate le reti *wired*. In una prima fase sono state analizzate le differenti architetture possibili per le reti di nuova generazione, giungendo alla conclusione che la scelta che verrà probabilmente effettuata in futuro in Italia sarà quella dell' utilizzo di un sistema GPON con fibra fino all' armadio stradale(FTTC) o direttamente fino all' utente (FTTH). La seconda fase di studio si è focalizzata su una ricerca esaustiva degli esistenti algoritmi di routing e dei protocolli multicast affidabili e, notando le eterogenee descrizioni di questi ultimi, difficilmente comparabili, è stato proposto il linguaggio AUML per una loro schematizzazione omogenea.

Sono state quindi illustrate, secondo modalità che permettano una interattività di tipo debole ovvero senza canale di ritorno, differenti tipologie di

servizi MHP implementate a partire da specifiche richieste di alcuni enti regionali: contenuti informativi navigabili attraverso due livelli di filtraggio, servizi con accesso condizionato per la visualizzazione di dati personali, gallerie di immagini e testo per simulare visite guidate ed infine un test di autovalutazione associato ad un sistema di t-learning per il quale, in futuro, vorremmo analizzare ulteriori metodi di cifratura e compressione comparandoli con quello attuale per ottimizzare la fase di scambio dati finale. I servizi informativi sono stati in seguito sviluppati anche con tecnologia Teletext al fine di essere utilizzabili tramite qualsiasi decoder, vista la scarsa diffusione di set-top-box interattivi sul territorio italiano. Come integrare invece altre tipologie di servizi per poterne usufruire da Teletext sarà un futuro oggetto di studio. Le prestazioni dello stesso servizio sviluppato in MHP e Teletext sono state misurate e comparate per valutare pro e contro di entrambe le tecnologie evidenziando come, tra le due, il Teletext sia maggiormente slegato dalla velocità di interazione dell' utente ma più limitato dal numero di pagine totali da gestire. La totalità dei servizi, compresa la versione Teletext, è stata infine riprodotta, con stessa grafica e funzionalità, anche su sito web. Questo per avvicinare a internet le fasce di popolazione, che risultano tutt' altro che esigue nella nostra nazione, che conoscono e utilizzano la televisione ma non hanno le conoscenze necessarie per approciarsi o trovare interesse verso il computer e il web.

Parallelamente sono stati sviluppati differenti progetti che hanno permesso la raccolta di molteplici dati di pubblico interesse, che possono essere messi a disposizione per il cittadino anche attraverso il sistema multicanale creatosi. Il progetto di un Centro di Gestione Dati per il monitoraggio ambientale raccoglie dati riguardanti sensori eterogenei dislocati sul territorio regionale (fonometri, pluviometri, spire di misurazione di traffico, misurazioni di temperatura, polveri sottili e inclinometri per il movimento del terreno in zone franose) che possono essere messi in relazione e fornire informazioni più o meno complesse. Dati i notevoli riscontri avuti dagli enti regionali il progetto è stato ulteriormente esteso in una forma più evoluta grazie ai LabICT-PA. Un secondo progetto di monitoraggio di reti e servizi è stato invece implementato sfruttando la rete Lepida e raccogliendo, uniformando, aggregando, i dati riguardanti il traffico di rete (Internet, wireless, HDSL, satellitare, ecc) e la frequenza di utilizzo dei servizi erogati sulla rete stessa. Il portale per la raccolta di OpenData, implementato per la Regione Emilia Romagna a seguito degli obiettivi richiesti sia dall' Agenda Digitale Europea che del Piano Telematico Regionale 2011-2013, è infine risultato un punto di convergenza di dati provenienti dai progetti precedenti e da altri enti del territorio fornendo informazioni di ogni tipologia, dal catasto, alla popolazione, ai dati cartografici. Per ognuno di questi progetti è stato illustrato quali dei loro dati possano facilmente inserirsi, e come, all' interno del sistema multicanale tramite i servizi precedentemente illustrati.

La multicanalità è stata mantenuta anche nell' ambito audio/video. Infatti

non solo è stata studiata una filiera di conversione che permetta di avere alta qualità con poca banda a disposizione sia nel caso di normale programmazione da palinsesto che nel caso di dirette, ma è stato sdoppiato il flusso per poterlo visualizzare anche in simulcast sul web. Gli stessi video convertiti in un formato uniforme restano anche disponibili per una visione OnDemand. Il carousel di LepidaTV, contenente sia audio/video che, ad oggi, otto servizi, è stato reso più efficiente fissando la banda per audio e video e ottimizzando la bit-rate da associare a ciascun servizio a seconda di differenti vincoli di importanza assoluta e diversi livelli di priorità tra essi così da assicurarne un tempo di caricamento minimizzato a seconda del loro utilizzo e dimensione. Nell' ambito *wired* sono state evidenziate le criticità dell' IPTV dovute sia all' aspetto infrastrutturale che alle richieste sempre più complesse degli utenti. Data la necessità del multicast per un' ottimizzazione della banda, è stata quindi effettuata un' indagine su quali protocolli possano essere utilizzati nella rete di distribuzione ed in quella di accesso.

La multicanalità è stata estesa ulteriormente tramite lo sviluppo di uno streaming audio/video e di un servizio informativo, anche su una piattaforma open source emergente, chiamata *OpenBOXware* ed in futuro sono previsti ulteriori ampliamenti. Innanzitutto realizzando test effettivi sull' IPTV, sfruttando la rete in fibra di Lepida e analizzando le prestazioni a seconda dei protocolli multicast utilizzati, degli operatori e dei servizi presenti. In seguito includendo anche il mondo dei dispositivi mobili cellulari: ad esempio il DVB-H per la trasmissione di audio-video ed uno studio di come sviluppare mobile application che mostrino le informazioni distribuite ad oggi tramite gli altri mezzi comunicativi illustrati.

Bibliografia

[1] A. Bria, f. Gessler, O. Queseth, R. Stridh, J. Wu, J. Zander, 4-th Generation Wireless Infrastructures: Scenario and Research Challenges, IEEE Personal communications, Dec. 2001, pp. 25-31.

[2] M.J.B.Robshaw, "Block cipher", RSA Laboratories Technical Report TR-601, August 1995.

[3] E.Biham, A.Shamir,"Differential cryptanalysis of DES-like cryptosystems", Journal of Cryptology, Vol 4, No. 1, pp.3-72, 1991.

[4] http://www.lepida.it

[5] Seminar on Tetra Market and Technology Developments. IEE Seminar on Volume , Issue 2000, Page(s):10/1 - 10/20.

[6] http://www.istat.it/it/archivio/banda+larga

[7] http://www.censis.it/

[8] http://www.lepida.tv

[9] http://www.dgtvi.it/switchoff.php

[10] http://www.dvb.org

[11] ETSI TS 101 812 V1.3.1 (2003-06) Technical Specification, Digital Video Broadcasting (DVB) - Multimedia Home Platform (MHP) Specification 1.0.3

[12] http://www.mhp.org

[13] http://java.sun.com/products/JavaTV

[14] http://www.dgtvi.it

[15] http://www.avalpa.com/the-key-values/15-free-software/33-opencaster

[16] http://www.regionedigitale.net/piano-telematico-2011-2013/

[17] http://ec.europa.eu/information_society/digital-agenda/index_en.htm

[18] http://www.campusdemedia.it/public/prima-parte-3.pdf

[19] Lo Presti, G. , "Il problema di Steiner nelle reti", Tesi di Laurea in Ingegneria Informatica, A/A 1999/2000

[20] Gatani, L. Lo Re, G. Gaglio, S. , "An efficient distributed algorithm for generating multicast distribution trees", International Conference Workshops on Parallel Processing, 2005. ICPP 2005 Workshops.

[21] Matsumura, R. Inoue, M. Tsujino, M. Iwashita, M. , "Evaluation of MPLS P2MP Distribution Tree Algorithms ", The 13th International Telecommunications Network Strategy and Planning Symposium, 2008. Networks 2008.

[22] Mochizuki, K. Shimizu, M. Yasukawa, S. , "CAM05-3: Multicast Tree Algorithm Minimizing the Number of Fast Reroute Protection Links for P2MP-TE Networks", IEEE Global Telecommunications Conference, 2006. GLOBECOM '06.

[23] Hui Cheng,Jiannong Cao,Srinivasan Mullai, Xingwei Wang ,"A Heuristic Multicast Algorithm to Support QoS Group Communications in Heterogeneous Network", Proceedings of the Second International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, 2005

[24] Brian Neil Levine, J.J. Garcia-Luna-Aceves, "A comparison of reliable multicast protocols", Journal Multimedia Systems, Volume 6 Issue 5, sept. 1998

[25] Yavatkar, R. Griffioen, J. Sudan,M., "A Reliable Dissemination Protocol for Interactive Collaborative Applications". In Proceedings of the ACM Multimedia '95 Conference, November 1995.

[26] Lin, J.C. Paul, S. "RMTP: a reliable multicast transport protocol" 1996. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE Volume 3, pp:1414-1424, 1996

[27] Whetten, B. Taskale, G. , " An overview of Reliable Multicast Transport Protocol II ", IEEE Network, Volume 14 Issue 1 , pp. 37-47, 2000

[28] Floyd, S., Jacobson, V., Liu, C., McCanne, S., and Zhang, L., "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", IEEE/ACM Transactions on Networking, December 1997, Volume 5, Number 6, pp. 784-803.

[29] Jim Gemmell, Todd Montgomery, Tony Speakman, Nidhi Bhaskar, Jon Crowcroft "The PGM Reliable Multicast Protocol", IEEE Network, Volume 17 , Issue 1, pp. 16-22, 2003

[30] Papadopoulos, C. Parulkar, G. Varghese, G. "Light-weight multicast services (LMS): a router-assisted scheme for reliable multicast",IEEE/ACM Transactions on Networking, Volume 12 Issue 3, pp. 456-468, 2004

[31] Macker, J.P. Adamson, P.B. "The multicast dissemination protocol (MDP) toolkit", Military Communications Conference Proceedings, 1999

[32] Yajian, Z. Xuedong, W. , "A Region-Based Multicast Routing Protocol for Ad Hoc Networks" ICCT '06. International Conference on Communication Technology, 2006

[33] Whetten, B. Montgomery , T. Kaplan, S. M. , "A high performance totally ordered multicast protocol", Selected Papers from the International Workshop on Theory and Practice in Distributed Systems, 1995

[34] Motyckova, L. Carr, D. , "A cluster ring topology for reliable multicasting", The International Journal of Computer and Telecommunications Networking, Volume 49 Issue 6, 2005

[35] Venugopalan, R. C. Venugopalan, R. Birman , K. P., "Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks", Proc. 21st International Conference on Distributed Computing Systems, 2001

[36] Luo, J. Eugster, P. Th. Hubaux, J.-P. "Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks", 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, 2003

[37] Benetti, E. Mazzini, G. , "Agent UML for Reliable Multicast Protocols Design", 17th International Conference on Software, Telecommunications and Computer Networks. SoftCOM 2009. September 24-26, 2009

[38] Booch, G. Rumbaugh, J. Jacobson, I. , "The Unified Modeling Language User Guide", Addison Wesley, 1999

[39] Bauer, B. Odell, J. , "UML 2.0 and agents: how to build agent-based systems with the new UML standard", Journal of Engineering Applications of Artificial Intelligence Volume 18, Issue 2 , March 2005

[40] Odell, J. , "An Extension of UML by Protocols for Multiagent Interaction", Proceedings of the Fourth International Conference on MultiAgent Systems, 2000

[41] Ming-Syan, C. Kun-Lung, W. Philip S., Y. "Optimizing Index Allocation for Sequential Data Broadcasting in Wireless Mobile Computing",

IEEE Transactions on Knowledge and Data Engineering (TKDE), 15(1), pp 161-173, January/February 2003.

[42] Benetti, E. Mazzini, G. , "Weak Interactive Systems Design in Wireless Broadcast Channels" , pp. 1-5, Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th

[43] Alvarez, R.P.M., Rodriguez-Diez-Caballero, J.L., Pousada-Carballo, J.M. Costas-Rodriguez, S. Rodriguez-Hernandez, P.S. Gonzalez-Castano, F.J. Fernandez-Iglesias,M.J. "Automating Content Update for MHP Applications: A Practical Experience", pp. 1-6, Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE.

[44] Kaixiong, S. Yihong, P. "A Method for Teletext Display", International Conference on Computer Graphics, Imaging and Visualization, 2006 Page(s): 231-235,2006

[45] Benetti, E. Mazzini, G. , "Interactive Services for Digital Terrestrial Television" , in Proceedings of the International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2010), 2010

[46] Benetti, E. Mazzini, G. , "MHP Application for a Self Evaluation Service", 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2011), 2011

[47] Baldi, M. De Santis, A. Falcone, D. Gambi, E. Spinsante, S., "A T-Learning Platform based on Digital Terrestrial Television", Softcom, pp. 347-351, 2006 International Conference on Software in Telecommunications and Computer Networks, 2006

[48] Lopez-Nores, M. Blanco-Fernandez, Y. Pazos-Arias, J.J. : "Architecting multimedia-rich collaborative learning services over Interactive Digital TV", pp. 1, 5th Iberian Conference on Information Systems and Technologies (CISTI), 2010

[49] Benetti, E. Salbaroli, E. Taddia, C. Mazzini, G. , "Centralized Management of Data Collection over Hybrid Networks", First International Conferences on Access Networks, Services and Technologies. ACCESS 2010. September 20-25, 2010 - Valencia, Spain

[50] Taddia, C. Nanni, S. Mazzini, G. , "Technology Integration for the Services Offered by the Public Administrations". INTERNET 2009, August 23-29, 2009 - Cannes/La Bocca, French Riviera, France.

[51] Benetti, E. Taddia , C. Mazzini, G., "Environmental Monitoring Supported by the Regional Network Infrastructure", Environmental Monitoring, ISBN 978-953-307-724-6, Edited by: Ema O. Ekundayo, Publisher: InTech, November 2011 (pages 389 - 410)

[52] http://www.openlivinglabs.eu/

[53] Benetti, E. Mazzini, G. , "Network and Services Monitoring System", 19th International Conference on Software, Telecommunications and Computer Networks. (SoftCOM 2011), 2011

[54] Heward, G. Muller, I. Jun Han Schneider, J.-G. Versteeg, S. : "Assessing the Performance Impact of ServiceMonitoring" , pp. 192-201, 2010, 21st Australian Software Engineering Conference (ASWEC), 2010

[55] Ten, D.W.H. Manickam, S. Ramadass, S. Al Bazar, H.A. : "Study on Advanced Visualization Tools In NetworkMonitoring Platform", pp. 445-449, 2009, Third UKSim European Symposium on Computer Modeling and Simulation, 2009. EMS '09

[56] Liu Yucheng, Liu Yubin : "A Monitoring System Design Program Based on B/S Mode" , pp. 184-187, 2010, International Conference on Intelligent Computation Technology and Automation (ICICTA), 2010

[57] Kamoshida, Y. Taura, K.: "Scalable Data Gathering for Real-Time MonitoringSystems on Distributed Computing", pp. 425-432, 2008, 8th IEEE International Symposium on Cluster Computing and the Grid, 2008. CCGRID '08

[58] http://oss.oetiker.ch/rrdtool/

[59] Robson, C.C.W. Silverstein, S. Bohm, C. : "Implementing clients for control and monitoringusing AJAX, pp. 538-539, 2007, Nuclear Science Symposium Conference Record, 2007

[60] http://pchart.sourceforge.net/

[61] http://www.camera.it/parlam/leggi/deleghe/testi/06036dl.htm

[62] http://dati.emilia-romagna.it

[63] http://dati.piemonte.it

[64] http://www.joomla.it/

[65] http://creativecommons.org/publicdomain/zero/1.0/

[66] http://creativecommons.org/licenses/by/2.0/

[67] Chung-yen, O. Jian, S. Changyong, P. Yangang, L. "Technology and standards of digital television terrestrial multimedia broadcasting" Communications Magazine, IEEE p. 119, 2010.

[68] http://ffmpeg.org/

[69] http://linux.die.net/man/1/mpeg2enc

[70] http://flowplayer.org/

[71] Yang, X. Xiaojian, D. Jingyuan, Z. Fei, H. Sghaier, G., "Internet Protocol Television (IPTV): The Killer Application for the Next-Generation Internet", IEEE Communications Magazine, pp 126-134 , November 2007

[72] Minoli, D., "IP multicast with applications to IPTV and mobile DVB-H", ISBN 978-0-470-25815-6, Edited by: John Wiley & sons, inc., 2008

[73] Nokia Siemens Networks Corporation, "High-quality and resilient IPTV multicast architecture", TechnicalWhite Paper, 2008

[74] Singhal, N. K., "Survivable Multicast Communications in Next Generation High-Capacity Networks", Dissertation for the degree of Doctor of Philosophy in Computer Science, University of California, Davis, 2004

[75] http://www.neunet.it/openboxware/

[76] Seraghiti, A. Klopfenstein, L. Bonino, S. Tarasconi, A. Bogliolo, A., "Multicast TV over Wireless Neutral Access Networks", in Proceedings of the International Conference on Access Networks, Services and Technologies (ACCESS2010), 2010.

[77] Klopfenstein, L. Delpriori, S. Valentini, M. Seraghiti, A. Bogliolo, A., "Protected Delivery of Multimedia Contents over Multicast IP Networks: an Open-Source Approach", 12th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WOWMOM 2011, Lucca, Italy, 20-24 June, 2011. pages 1-3, IEEE, 2011

[78] http://www.neunet.it/openboxware/wp-content/uploads/2010-/12/LisbonaTutorial.pdf

Pubblicazioni

Benetti, E. Mazzini, G. , “Weak Interactive Systems Design in Wireless Broadcast Channels” , pp. 1-5, Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th

Benetti, E. Mazzini, G. , “Agent UML for Reliable Multicast Protocols Design”, 17th International Conference on Software, Telecommunications and Computer Networks. (SoftCOM 2009), 2009

Benetti, E. Mazzini, G. , “Interactive Services for Digital Terrestrial Television” , in Proceedings of the International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2010), 2010

Benetti, E. Salbaroli, E. Taddia, C. Mazzini, G. , “Centralized Management of Data Collection over Hybrid Networks”, First International Conferences on Access Networks, Services and Technologies. ACCESS 2010. September 20-25, 2010 - Valencia, Spain

Benetti, E. Mazzini, G. , “MHP Application for a Self Evaluation Service”, 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2011), 2011

Benetti, E. Mazzini, G. , “Network and Services Monitoring System”, 19th International Conference on Software, Telecommunications and Computer Networks. (SoftCOM 2011), 2011

Benetti, E. Taddia , C. Mazzini, G., “Environmental Monitoring Supported by the Regional Network Infrastructure”, Environmental Monitoring, ISBN 978-953-307-724-6, Edited by: Ema O. Ekundayo, Publisher: InTech, November 2011 (pages 389 - 410)