UNIVERSITÁ DI PADOVA

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

Sviluppo di un access gateway residenziale per la condivisione multimediale in una rete domestica

Laureando: Andrea Nasato 541683-IF

Relatore: Francesco Bombi

Correlatore: Massimo Pegorer

26/03/2010

A.A. 2009-2010

Vorrei ringraziare innanzitutto i miei genitori,
che mi hanno permesso di arrivare
fino questo punto della mia vita.
Ringrazio inoltre il mio relatore, il prof. Franco Bombi,
l'azienda Telsey,
in particolare l'ing. Massimo Pegorer,
che mi ha seguito durante e dopo il tirocinio.
Infine, ma non in ordine di importanza,
ringrazio Sara,
tutti i miei amici e parenti.

INDICE

INDICE DELLE ILLUSTRAZIONI

INDICE DELLE TABELLE

INDICE DEI TESTI

1 INTRODUZIONE

L'oggetto di questa relazione di tirocinio è lo sviluppo del software per un access gateway residenziale con cui implementare funzionalità di condivisione multimediale in una rete domestica, realizzando quindi un'infrastruttura di distribuzione di contenuti come immagini, clip audio e video in un ambiente ad estensione limitata, com'è tipicamente un'abitazione:

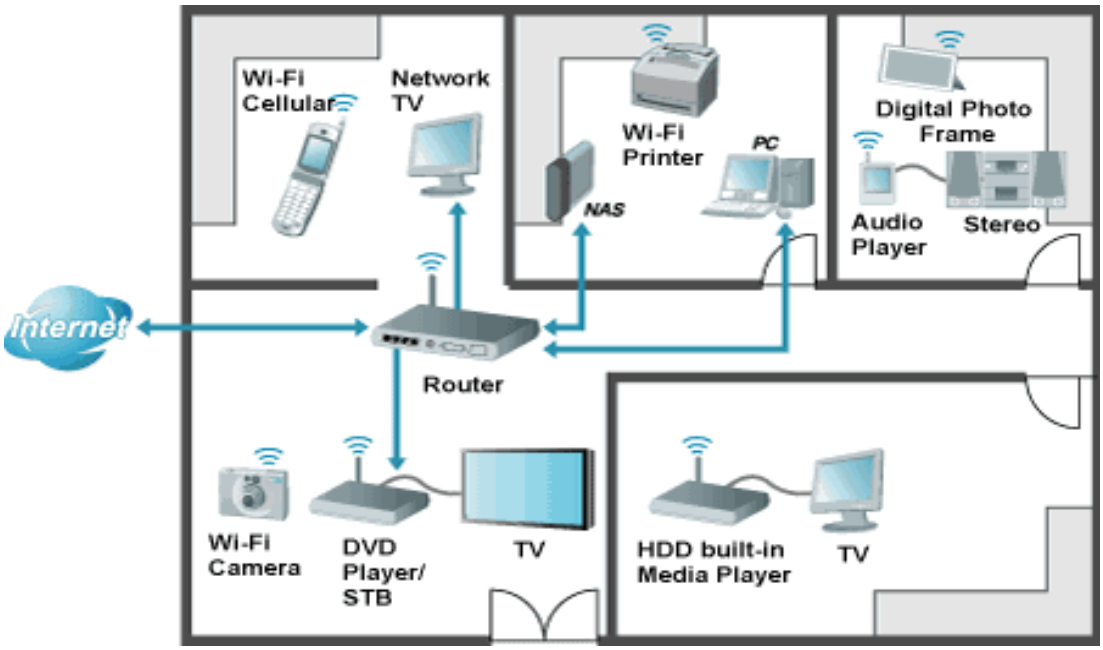

Illustrazione 1: Esempio di applicazione dell'architettura UPnP/DLNA

L'obiettivo è quindi quello di realizzare un sistema di condivisione multimediale utilizzando mezzi di interconnessione di tipo *wired* oppure *wireless* a seconda della tipologia dei dispositivi. In questo modo si intende semplificare l'interoperabilità tra le varie apparecchiature, senza ricorrere a periferiche di memorizzazione per il trasporto dei contenuti ed evitare le incompatibilità che possono presentarsi durante l'interazione di dispositivi diversi. Per la realizzazione di questa infrastruttura di distribuzione multimediale è necessaria l'implementazione di uno standard che possa essere di pubblico dominio ed abbastanza diffuso nell'ambiente delle apparecchiature elettroniche per l'intrattenimento, in modo da avere un discreto numero di categorie di dispositivi e quindi una grande varietà di configurazioni possibili. Lo standard in questione è stato concepito da un'organizzazione composta da diversi produttori, nota con l'acronimo **DLNA** (*Digital Living Network Alliance*), e vanta già un numero rilevante di dispositivi supportati. Tale architettura, oltre a stabilire quali interfacce di connessione e che tipi di contenuti multimediali utilizzare, si basa sui mezzi di comunicazione e sulle direttive specificate da un'altra organizzazione, nota con il nome di **forum UPnP** (*Universal Plug 'n' Play*), la quale si occupa di sviluppare delle interconnessioni semplici e al contempo affidabili tra dispositivi eterogenei di produttori diversi.

Il tirocinio si è svolto presso l'azienda Telsey S.P.A., la cui sede e principale reparto R&D si trovano a Quinto di Treviso. Altre sedi sono, in Italia, a Benevento e Milano, e all'estero, in Austria, Russia e Regno Unito. In particolare, il tirocinio si è tenuto nel contesto delle attività del reparto R&D della Access Device Business Unit. L'azienda si occupa di progettare, sviluppare e commercializzare dispositivi e soluzioni per la modernizzazione delle reti d'accesso a banda larga, come access gateway, wireless access point e switch; vengono inoltre sviluppati e prodotti dispositivi per l'intrattenimento multimediale come decoder per il digitale terrestre e Set Top Box (STB).

2 GLI STANDARD UPNP E DLNA

In questo capitolo verranno descritti gli standard UPnP e DLNA, utilizzati per lo sviluppo dell'access gateway multimediale.

2.1 UPnP

Lo standard UPnP è stato concepito dall'omonimo forum per realizzare un'architettura in grado di offrire una connettività di tipo *peer-to-peer* tra PC, apparecchiature elettroniche e dispositivi *wireless*, utilizzando protocolli impiegati soprattutto per le connessioni ad Internet, quali IP, TCP, UDP, HTTP e XML. Questa architettura offre l'indipendenza tra dispositivi, piattaforme e *media format*, il rilevamento automatico delle apparecchiature elettroniche, e la semplicità con cui è possibile instaurare una connessione tra diverse categorie di device, realizzando una rete a "configurazione nulla" (*zero configuration*). La documentazione UPnP indica due principali tipologie di dispositivi: i punti di controllo (*control point*) e dispositivi controllati, o semplicemente dispositivi (*device*); questi ultimi hanno il ruolo di server, rispondendo alle richieste generate dai punti di controllo.

L'architettura UPnP è rappresentata nella seguente figura:

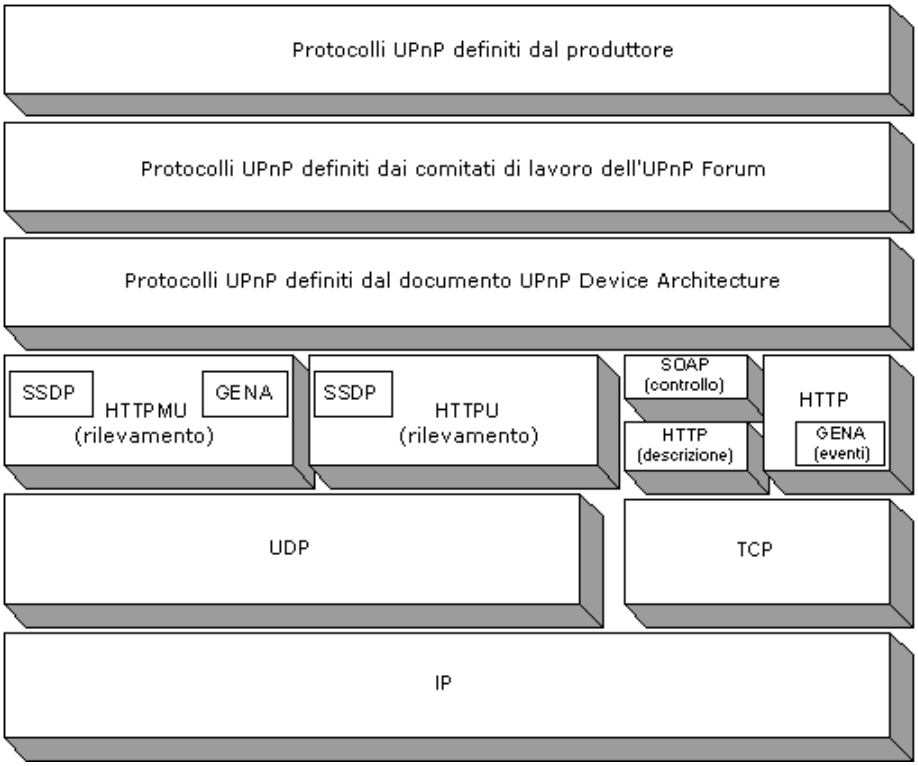

Illustrazione 2: Architettura UPnP

Il protocollo di rete utilizzato è il diffuso IP (*Internet Protocol*), mentre per il livello di trasporto vengono utilizzati sia UDP che TCP, a seconda della funzionalità che si implementano. In particolare, l'architettura UPnP prevede una serie di sei fasi per la realizzazione di una rete domestica:

- *Indirizzamento*: ogni dispositivo deve integrare un client DHCP, in modo da ottenere automaticamente un indirizzo IP quando esso si connette alla rete. Se al momento non vi sono server DHCP attivi, va utilizzato un protocollo denominato *Auto IP* con il quale, attraverso un'interrogazione della rete in *multicast* e un'analisi della tabella di *ARP*, viene assegnato un indirizzo IP temporaneo ed ovviamente univoco. Tale indirizzo deve però essere sostituito con quello fornito da un eventuale server DHCP attivo.
- *Rilevamento*: una volta che è stato assegnato un indirizzo di rete a tutti i dispositivi, si può procedere al rilevamento degli stessi per mezzo del protocollo SSDP (*Simple Service Discovery Protocol*); quest'ultimo utilizza delle versioni modificate di HTTP, denominate HTTPU (dove la U sta per UDP) e HTTPMU (dove M sta per *Multicast*, mentre U sempre per UDP). Con questi protocolli i pacchetti HTTP non vengono trasportati mediante TCP, come avviene normalmente, bensì attraverso UDP. Quando un dispositivo si connette alla rete invia una richiesta di ricerca SSDP *multicast* (tramite HTTPMU). Le altre periferiche UPnP, quando ricevono questa richiesta, esaminano i criteri di ricerca per stabilire se soddisfa tali criteri; in caso affermativo, al dispositivo viene inviata una risposta SSDP *unicast* (attraverso HTTPU). Il protocollo SSDP fornisce inoltre le funzionalità necessarie per la corretta disconnessione di una periferica UPnP.
- *Descrizione*: la fase successiva nella connettività di rete UPnP è rappresentata dalla descrizione, che serve ai punti di controllo per ottenere ulteriori informazioni sui dispositivi presenti nella rete. La descrizione UPnP di una periferica è in formato XML e include informazioni specifiche del produttore, quali il nome e il numero del modello, il numero di serie, il nome del produttore, ecc. Inoltre è incluso l'elenco di eventuali periferiche o servizi incorporati, nonché URL per il controllo, la generazione di eventi e la presentazione.
- *Controllo*: ottenuta la descrizione UPnP dei dispositivi, i punti di controllo possono ottenere in un egual modo la descrizione UPnP dei servizi offerti dalle periferiche, anch'essa in formato XML e comprendente l'elenco dei comandi (o azioni) ai quali il servizio risponde e dei parametri (o argomenti) per ogni azione. Per controllare una periferica, un punto di controllo invia una richiesta di azione ad un servizio della periferica, utilizzando l'URL ricavata durante la fase di descrizione. I messaggi di controllo si basano sul protocollo SOAP (*Simple Object Access Protocol*), il quale definisce l'utilizzo degli standard XML e HTTP per l'esecuzione di chiamate a procedure remote (RPC, *Remote Procedure Call*). Analogamente a una chiamata a una procedura remota, la tecnologia UPnP utilizza il protocollo SOAP per recapitare messaggi di controllo alle periferiche e restituire risultati o errori ai punti di controllo. Ogni richiesta di controllo UPnP è un messaggio SOAP

contenente l'azione da richiamare più un insieme di parametri. Anche la risposta è un messaggio SOAP, che contiene lo stato, il valore restituito ed eventuali parametri restituiti.

- *Gestione degli eventi*: nella descrizione UPnP di un servizio è incluso l'elenco delle azioni alle quali il servizio risponde e l'elenco delle variabili che modellano lo stato del servizio in fase di esecuzione. Quando il valore di queste variabili cambia, il servizio pubblica degli aggiornamenti; un punto di controllo può sottoscrivere la ricezione di queste informazioni. Il servizio pubblica gli aggiornamenti inviando messaggi di notifica degli eventi che contengono i nomi di una o più variabili di stato e il valore corrente di tali variabili. Anche questi messaggi sono espressi in XML e formattati utilizzando GENA (*Generic Event Notification Architecture*), un protocollo realizzato per consentire l'invio e la ricezione di notifiche attraverso gli *stack* HTTP su TCP e UDP *multicast*.
- *Presentazione*: se una periferica dispone di un URL per la presentazione, il punto di controllo può richiamare una pagina da tale URL, caricarla in un browser e, in base alle capacità della pagina, consentire a un utente di controllare la periferica e di visualizzarne lo stato.

L'architettura UPnP è perciò basata su standard esistenti, consentendo di realizzare con facilità l'interoperabilità e mantenendo al tempo stesso una flessibilità tale da soddisfare le esigenze dei dispositivi delle reti attuali e probabilmente future.

Le specifiche UPnP sono organizzate in aree funzionali: ogni gruppo di lavoro all'interno del forum definisce delle specifiche applicabili a dispositivi e servizi con funzionalità peculiari. Ad esempio esistono le specifiche UPnP-AV (dove AV sta per Audio/Video) per le funzionalità multimediali, UPnP-IGD (*Internet Gateway Device*) per il networking e la connettività broadband, UPnP-QoS (*Quality of Service*) per le funzionalità di qualità del servizio e così via.

2.2 UPnP-AV

L'architettura UPnP-AV definisce le specifiche di interazione tra punti di controllo e dispositivi UPnP con funzionalità audio/video, quali televisori, Set Top Box, hard disk multimediali, impianti Hi-Fi e così via. Vengono perciò applicati i concetti dello standard UPnP ai contenuti multimediali, consentendo una condivisione trasparente per l'utente finale.

Le specifiche UPnP-AV indicano tre tipologie di elementi che andranno ad interagire tra loro:

- *Media Server*: fanno parte di questa categoria quei dispositivi che sono in grado di rendere disponibili nella rete UPnP delle risorse multimediali (più comunemente file, ma anche *streaming*).
- *Media Renderer*: sono i dispositivi in grado di ricevere dei file multimediali, decodificarli ed effettuarne la riproduzione; indicano quindi la destinazione del flusso generato dallo *streaming* dei contenuti.
- *Control Point*: sono quei dispositivi o funzioni con i quali è possibile sfogliare i contenuti memorizzati in un Media Server, avviarne e controllarne la riproduzione.

Nella seguente figura viene riportato il modello che rappresenta l'interazione tra questi tre elementi:

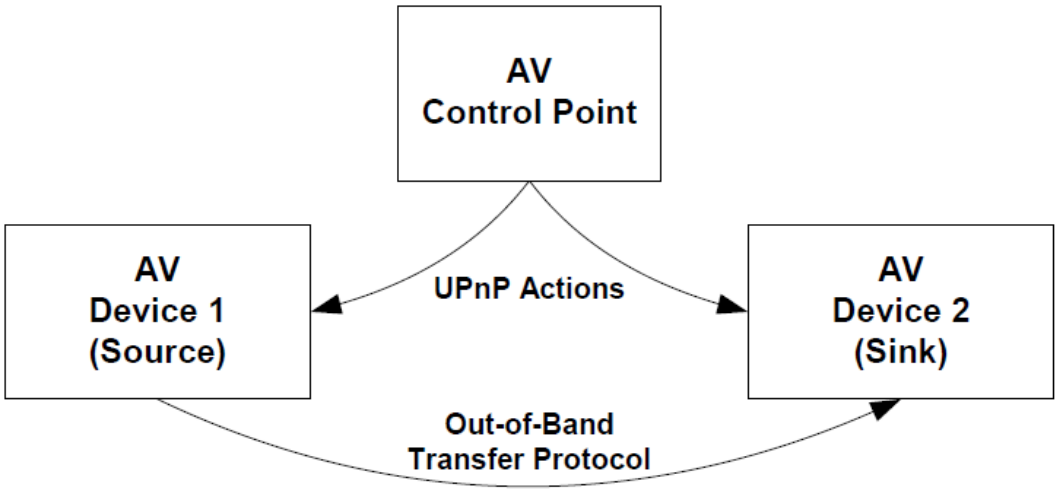

Illustrazione 3: Modello di interazione tra dispositivi UPnP-AV

Il punto di controllo utilizza i servizi e le azioni offerte dai dispositivi, come avviene per le normali reti UPnP, mentre il trasferimento dei contenuti avviene attraverso un protocollo di comunicazione non UPnP. In questo modo, una volta che il punto di controllo ha inizializzato e configurato entrambi i dispositivi, lo *streaming* dei contenuti può avvenire anche senza la presenza di questo elemento nella rete.

I dispositivi Media Server e Media Renderer, essendo elementi UPnP, implementano dei servizi per mezzo dei quali un punto di controllo può gestire e configurare la rete UPnP; entrambi i dispositivi offrono un servizio denominato *ConnectionManager*, il quale è caratterizzato da funzioni di gestione della connessione associata al dispositivo stesso. Essi offrono inoltre il servizio *AVTransport*, che viene utilizzato dal punto di controllo per gestire la riproduzione di un determinato contenuto multimediale; sono quindi implementate azioni di *Play*, *Pause*, *Stop*, *Seek*, ecc. Nell'elemento Media Server è realizzato anche il servizio *ContentDirectory*, che è caratterizzato dalle azioni necessarie per sfogliare e cercare i contenuti, nonché per ottenerne ulteriori informazioni quali durata, *bitrate*, ecc. Il Media Renderer implementa inoltre il servizio *RenderingControl*, il quale offre delle azioni utili a gestire la renderizzazione dei file, come la regolazione della luminosità, del contrasto, del volume e così via.

La figura seguente riporta un esempio di interazione tra i servizi offerti dai dispositivi UPnP-AV:

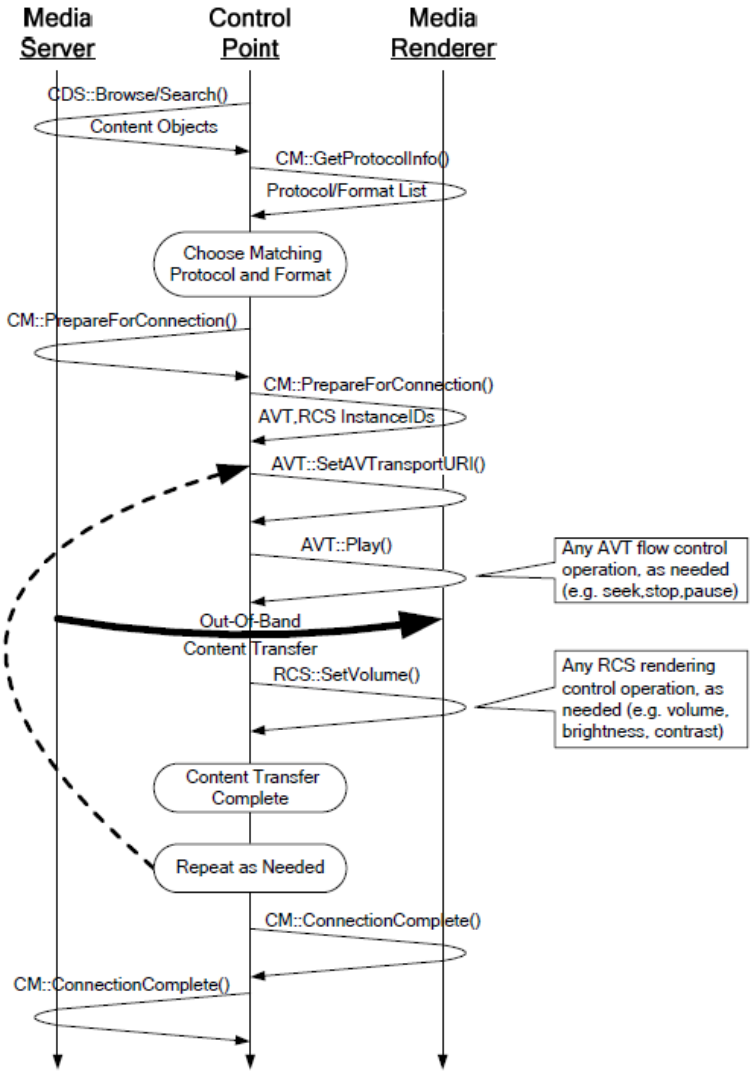

Illustrazione 4: Esempio di interazione tra servizi UPnP-AV

Il punto di controllo, per mezzo dell'azione *Browse()* o *Search()* implementate nel servizio *ContentDirectory*, sfoglia i contenuti multimediali del Media Server. Per verificare la compatibilità dei file a disposizione con il Media Renderer, viene eseguita l'azione *GetProtocolInfo()*, implementata nel servizio *ConnectionManager*. Dopo aver verificato la compatibilità di protocolli di trasporto e *media format* utilizzati, il punto di controllo configura la connessione dei due dispositivi con l'esecuzione del metodo *PrepareForConnection()*. A questo punto, con l'azione *SetAVTransportURI()* del servizio *AVTransport*, viene indicata al Media Renderer la posizione del contenuto multimediale memorizzato nel Media Server e può essere avviata la riproduzione con il metodo *Play()*; una volta finita, il punto di controllo comunica ai dispositivi la fine della trasmissione con l'azione *ConnectionComplete()*. Nel corso della riproduzione è possibile, qualora siano state implementate le funzioni necessarie, gestire lo *streaming* dei contenuti multimediali con i metodi offerti dai servizi *AVTransport* e *RenderingControl*.

Sebbene la documentazione preveda tre diversi elementi per le reti UPnP-AV, nelle reali implementazioni il punto di controllo può essere integrato nel Media Server o nel Media Renderer; nel primo caso, i contenuti da trasmettere saranno selezionati localmente, mentre sarà necessario indicare un dispositivo remoto in grado di decodificarli e farne la riproduzione. Nel secondo caso, sarà possibile sfogliare i contenuti memorizzati nel Media Server ed avviarne lo *streaming* direttamente nel Media Renderer. In questi due casi il punto di controllo non è quindi un dispositivo a sé stante, bensì un'applicazione residente in una delle due categorie di periferiche e spesso dotata di un'interfaccia grafica.

2.3 UPnP-QoS

Le specifiche UPnP-QoS sono state concepite per l'implementazione di funzionalità di qualità del servizio (*Quality of Service*, in breve *QoS*) in una rete di dimensioni limitate e per qualsiasi tipo di traffico, sebbene siano particolarmente indicate per lo *streaming* dei contenuti distribuiti secondo le specifiche UPnP-Audio/Video. Con l'utilizzo di tecniche di *QoS* è dunque possibile dedicare una parte della banda disponibile ad una determinata tipologia di traffico, così da ottenerne una trasmissione meno vincolata ai meccanismi di gestione delle code di instradamento e delle congestioni presenti in alcuni protocolli a livello di trasporto, come il TCP; questi algoritmi infatti, se da un lato realizzano una connessione stabile ed affidabile, dall'altro interferiscono con le applicazioni che necessitano di una trasmissione il più *real-time* possibile, come può essere ad esempio lo *streaming* di qualche contenuto multimediale. Eventuali ritardi nell'invio di qualche pacchetto, dovuti a possibili congestioni in una rete satura, si tradurrebbero in una riproduzione di scarsa qualità per l'utente finale.

Nella seguente figura viene schematizzata l'architettura UPnP-QoS:

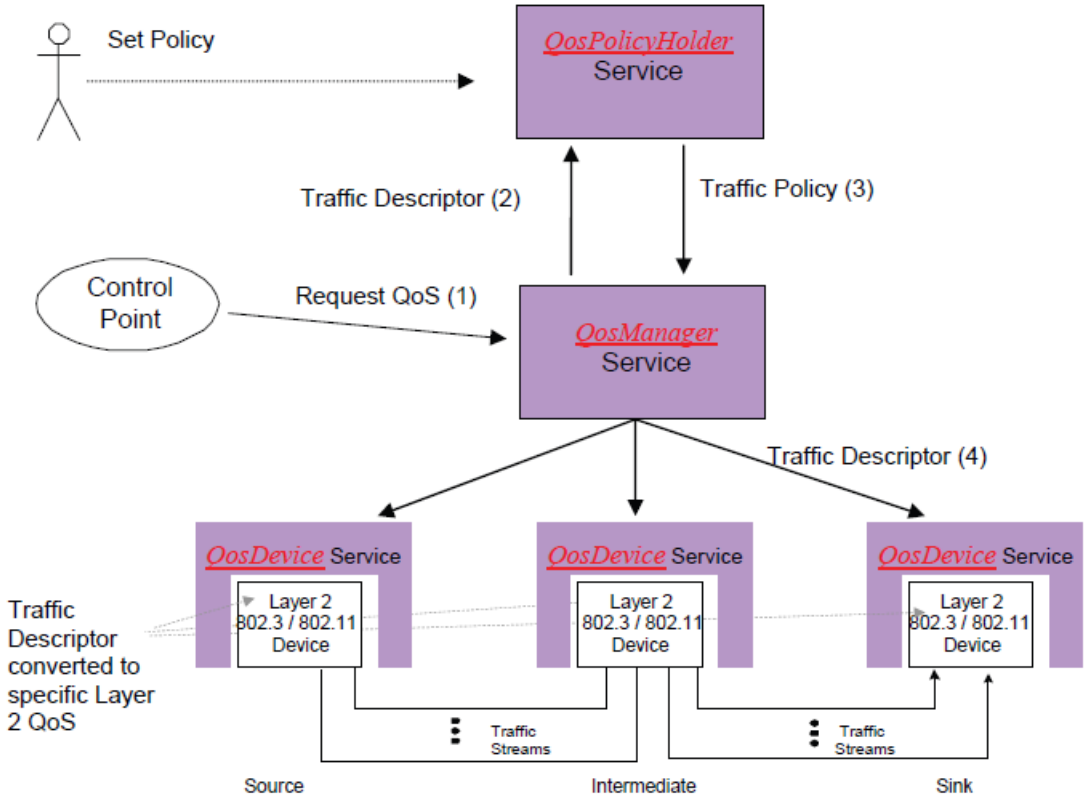

Illustrazione 5: Architettura UPnP-QoS

Nelle specifiche UPnP-QoS vengono definiti tre servizi, ciascuno con le proprie caratteristiche e funzioni:

- *QosPolicyHolder*: fornisce le politiche di gestione del traffico nella rete UPnP di cui fa parte; non è indispensabile qualora si utilizzino funzionalità *QoS* di tipo prioritario.
- *QosManager*: interagisce con il punto di controllo presente nella rete UPnP, compilando il *Traffic Descriptor* che andrà ad istruire i vari dispositivi.
- *QosDevice*: converte le informazioni ricevute dal servizio *QosManager* attraverso il *Traffic Descriptor* in istruzioni specifiche per le tecnologie di secondo livello ISO/OSI (il livello di *data-link*).

Il punto di controllo associato ad un determinato dispositivo fa richiesta *(1)*, attraverso il servizio *QosManager*, di abilitare la funzione di *Quality of Service* su un flusso di dati che intercorre tra diversi dispositivi, ognuno caratterizzato dal proprio indirizzo IP e MAC. Il *QosManager* compila un descrittore di traffico *(Traffic Descriptor),* costituito da un file XML, contenente le informazioni necessarie per poter stabilire la miglior politica di gestione delle connessioni; tali informazioni comprendono la topologia e gli indirizzi di rete dei dispositivi, la priorità che si intende attribuire al flusso in questione, il *data-rate* desiderato, la possibilità di prelazione (*pre-emption*), ecc. Il descrittore di traffico viene inviato al servizio *QosPolicyHolder (2)*, il quale, con tali informazioni, decide la miglior politica di gestione del traffico da applicare tra quelle disponibili e la comunica al servizio *QosManager (3)*; quest'ultimo inoltra la configurazione scelta ai dispositivi interessati *(4)* per mezzo di un altro *Traffic Descriptor*, il quale verrà convertito dai servizi *QosDevice* in istruzioni specifiche per il livello di *data-link* implementato localmente.

La tipologia di *QoS* appena descritta è detta **parametrizzata** (*parameterized QoS*) in quanto viene scelta la politica di gestione del traffico migliore in base ai parametri forniti attraverso il punto di controllo e alla configurazione della rete UPnP.

Esiste un'altra tipologia di *QoS*, denominata **a priorità** (*prioritized QoS*), che non richiede la presenza del servizio *QosPolicyHolder*, in quanto viene attribuito ad ogni flusso di traffico un numero identificativo di priorità da 0 a 7: alle trasmissioni associate alle priorità più alte deve essere data precedenza rispetto a quelle con priorità più basse. Il valore di priorità viene indicato dal punto di controllo e quindi è direttamente il servizio *QosManager* a comunicare ai vari dispositivi, attraverso il *Traffic Descriptor*, le azioni da intraprendere per fornire la qualità del servizio.

Come nell'architettura UPnP-AV, ogni servizio è caratterizzato dalle proprie funzioni, dette azioni, attraverso le quali avviene il trasporto delle informazioni da un servizio ad un altro. Nell'immagine sottostante è riportato un esempio di interazione tra servizi quando si applica una *parameterized QoS*:

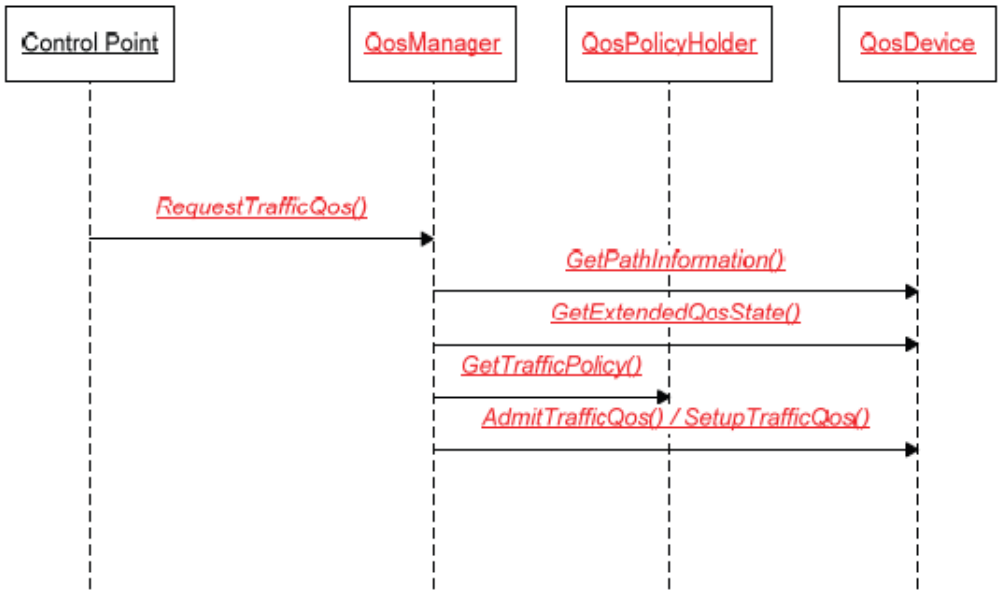

Illustrazione 6: Esempio di interazione tra servizi UPnP-QoS

Il punto di controllo comunica al *QosManager*, per mezzo della funzione *RequestTrafficQos()*, l'intenzione di applicare la qualità del servizio ad una connessione indicandola come parametro insieme alle sue caratteristiche. A questo punto il servizio *QosManager*, attraverso i metodi *GetPathInformation()* e *GetExtendedQosState()*, implementati nel servizio *QosDevice*, ottiene le informazioni relative alla topologia della rete e alle capacità e risorse disponibili in ciascun dispositivo, necessarie per effettuare la scelta della miglior politica di gestione applicabile; tali informazioni vengono in seguito usate come parametri di ingresso per il metodo *GetTrafficPolicy()*, il quale restituisce i valori corrispondenti alla politica scelta dal servizio *QosPolicyHolder* (ad esempio il *TrafficImportanceNumber*). Infine il *QosManager* può comunicare le istruzioni dell'ammissione della *Quality of Service* ai dispositivi con l'azione *AdmitTrafficQos()*.

Come per gli altri standard UPnP, anche in questo caso le informazioni vengono scambiate per mezzo di pacchetti HTTP contenenti degli schemi basati sul linguaggio XML. A differenza delle azioni dei servizi UPnP-AV, che fanno uso di parametri di tipo semplice, come stringhe e numeri interi, le azioni specificate dall'UPnP-QoS fanno uso di tipi di dati complessi, strutturati secondo degli schemi basati su XML. Ciò che si deve realizzare è quindi una sorta di incapsulamento di una struttura XML di campi e relativi valori all'interno di una macro-struttura XML di trasporto.

2.4 DLNA

Lo standard DLNA offre la possibilità di creare un servizio di condivisione multimediale in una rete domestica composta da elementi eterogenei, i quali hanno le proprie funzioni peculiari: ad esempio, i Media Player sono in grado di cercare i contenuti multimediali supportati messi a disposizione da uno o più Media Server e di farne la riproduzione. Tali dispositivi appartengono a diverse categorie, dagli hard disk esterni con interfaccia di rete ai televisori di ultima generazione, passando per gli smartphone più avanzati.

Per garantire una facile interazione e un'efficiente trasferimento di dati tra le varie tipologie di apparecchiature, l'organizzazione DLNA (Digital Living Network Alliance) ha rilasciato un documento contenente tutte le linee guida da seguire per sviluppare un dispositivi in grado di interagire con altri in modo più trasparente possibile per l'utente finale. In particolare, vengono indicati i tipi di connessione di rete, i protocolli di trasporto, individuazione e gestione dei contenuti e i formati multimediali supportati. Lo schema seguente riassume tutte le caratteristiche:

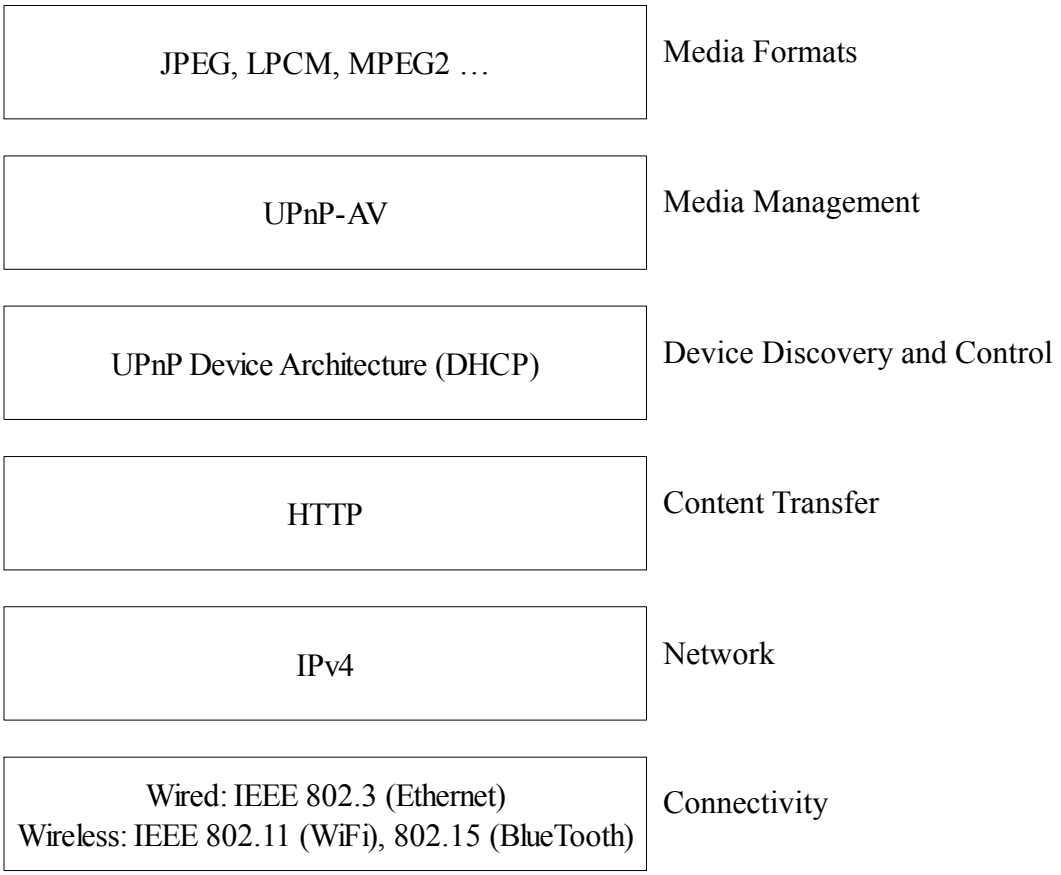

Illustrazione 7: Architettura DLNA

Gli sviluppatori dei dispositivi che potranno avere la certificazione DLNA devono quindi considerare sia l'aspetto hardware, implementando alcune delle più diffuse tecnologie di comunicazione, che quello software, adottando protocolli e formati file ampiamente utilizzati anche in altri ambiti. Si può vedere dallo schema riportato sopra che la rete basata su DLNA (e quindi su UPnP) utilizza lo *stack* TCP/UDP over IPv4, mentre il protocollo HTTP gestisce il trasferimento delle informazioni che i vari dispositivi si scambiano per poter interagire e dei contenuti multimediali.

2.4.1 DLNA Device Classes

Le linee guida dell'organizzazione DLNA stabiliscono una serie di classi di apparecchiature, con le quali si distinguono queste ultime in base alle proprie caratteristiche; in questo modo risultano più evidenti le differenze tra i vari elementi di una rete multimediale domestica.

Vi è innanzitutto una divisione tra le apparecchiature destinate ad un utilizzo prettamente domestico da quelle che sono portatili: ad esempio i televisori e gli impianti Hi-Fi fanno parte della prima categoria, mentre gli smartphone o i lettori multimediali portatili della seconda. In particolare, la prima categoria è chiamata **Home Network Device (HND)**, mentre la seconda **Mobile Handheld Device (MHD)**. All'interno di ciascuna vi sono ulteriori classi, diversificate dalle funzioni proprie del device e sono riportate nel seguente elenco:

Home Network Device (HND):

- Digital Media Server (DMS): raccoglie tutti i contenuti multimediali e li mette a disposizione dei dispositivi che possono farne la riproduzione (es.: computer desktop o portatili, videoregistratori, fotocamere digitali, ecc.);
- Digital Media Player (DMP): si interfaccia con i server disponibili nella rete domestica, ne sfoglia i contenuti multimediali e ne fa la riproduzione (es.: televisori di ultima generazione, cornici digitali, ecc.);
- Digital Media Renderer (DMR): come i DMP riproduce i contenuti multimediali messi a disposizione dei server, ma non può sfogliarne i contenuti (es.: televisori, impianti Hi-Fi, ecc.). Vanno utilizzati con un DMC;
- Digital Media Controller (DMC): sfoglia i contenuti multimediali dei DMS, verifica la compatibilità di riproduzione con il DMR ad esso associato e stabilisce la connessione tra server e renderer (es.: telecomando con funzioni avanzate, palmare, ecc.);
- Digital Media Printer (DMPr): stampa le foto o i documenti messi a disposizione da un DMS (stampanti con interfaccia di rete).

Mobile Handheld Device (MHD):

- Mobile Digital Media Server (M-DMS): come per il DMS, raccoglie e mette a disposizione i contenuti (es.: smartphone, riproduttori multimediali portatili, ecc.);
- Mobile Digital Media Player (M-DMP): sfoglia i contenuti dei M-DMS e ne fa la riproduzione (es.: smartphone, media tablet, ecc.);
- Mobile Digital Media Controller (M-DMC): sfoglia i contenuti dei M-DMS, controlla la compatibilità con un DMR ad esso associato e controlla la riproduzione verso quest'ultimo;
- Mobile Digital Media Uploader (M-DMU): invia dei contenuti multimediali, sotto forma di file vero e proprio, ad un M-DMS (es.: fotocamere digitali);
- Mobile Digital Media Downloader (M-DMD): sfoglia i contenuti, fa il download dei file multimediali e ne fa la riproduzione (es.: riproduttori multimediali portatili); a differenza dei M-DMP, non avviene lo *streaming* dei contenuti, ma la riproduzione in locale.

Al di là delle singole caratteristiche, la differenza principale tra dispositivi domestici e portatili risiede nel tipo di connettività: i primi devono avere interfaccia Ethernet e/o WiFi, mentre gli altri devono supportare il protocollo BlueTooth.

Oltre a quelle sopracitate esiste una terza categoria, detta **Home Infrastructure Device (HID)**, composta da quei dispositivi che si occupano di stabilire un'interfaccia di comunicazione tra le apparecchiature domestiche e portatili; tale categoria è composta da due classi:

- Mobile Network Connectivity Function (M-NCF): implementano funzionalità di *bridging* tra diversi protocolli di comunicazione: ad esempio BlueTooth ↔ WiFi oppure BlueTooth ↔ Ethernet;
- Media Interoperability Unit (MIU): eseguono la conversione dei contenuti tra diversi tipi di *media format*: ad esempio MPEG2 ↔ MPEG4 per i contenuti video oppure LPCM↔MP3 per quelli audio.

Qualora le apparecchiature domestiche e portatili considerate abbiano a loro disposizione una medesima infrastruttura di comunicazione e siano in grado di riprodurre i contenuti multimediali negli stessi *media format*, allora non vi è necessità di utilizzare un dispositivo di quest'ultima categoria.

3 DESCRIZIONE DEL PROGETTO

Il progetto oggetto del tirocinio consisteva nello sviluppo di un'applicazione software in grado di fornire funzionalità di condivisione multimediale in una rete domestica, con l'obiettivo di integrare tale applicazione in un dispositivo hardware di tipo broadband access gateway.

Lo scopo complessivo del progetto era quello di trasformare un comune broadband router residenziale in un dispositivo con ulteriori funzionalità, in particolare per la condivisione dei contenuti, tali da renderlo il punto focale delle attività multimediali domestiche.

Il progetto è stato suddiviso in due parti di maggior rilievo: nella prima, dopo lo studio della documentazione degli standard implementati, si è trattato di eseguire dei test sull'applicazione con funzionalità UPnP/DLNA realizzata dalla divisione di ricerca e sviluppo dell'azienda; i risultati ottenuti facevano parte della documentazione necessaria per ottenere la certificazione rilasciata dall'organizzazione DLNA.

La seconda parte del progetto è stata dedicata allo sviluppo di funzionalità di Quality of Service (QoS) applicate alla distribuzione multimediale, in modo da garantire una fruibilità ottimale dei contenuti in qualsiasi condizione di rete, e quindi un'ottima "user-experience". A differenza della prima parte del progetto, che consisteva sostanzialmente nell'eseguire dei test su applicazioni già realizzate, nel corso della seconda metà si è trattato di sviluppare del software e di testarne la corretta esecuzione, attuando infine una piattaforma di dimostrazione. Anche in questo caso sono state utilizzate le specifiche definite nella documentazione del forum UPnP (UPnP-QoS Architecture), sebbene non ne esistano ancora molte implementazioni; per questo motivo, grazie alla collaborazione tra l'azienda e l'Università di Ghent (Belgio), è stato possibile utilizzare un'applicazione in grado di interagire con il software realizzato nel corso del progetto. Tale applicazione, sviluppata in Java da un team di ricercatori dell'Università, è stata indispensabile per testare il corretto funzionamento dell'implementazione dei servizi *QoS* e per realizzare la piattaforma di dimostrazione finale. In questa seconda parte del progetto si è utilizzato il linguaggio C e l'ambiente di sviluppo integrato *Eclipse*.

Il dispositivo in questione è un access gateway residenziale con uplink ADSL2+, 4 interfacce LAN Ethernet 10/100, due porte FXS per servizi di telefonia VoIP, access point Wi-Fi 802.11b/g e due porte USB. Per quanto riguarda l'hardware, l'access gateway ha una CPU di tipo MIPS dual core (di cui uno interamente dedicato alle funzionalità VoIP), un switch Ethernet (che permette il bridging wire-speed del traffico LAN-to-LAN senza caricare la CPU), e dispone di 64 MB di memoria RAM. Il sistema operativo è basato su kernel Linux e le applicazioni poggiano su librerie uClibc.

Per l'access gateway si intendeva realizzare le funzionalità di un *Digital Media Server (DMS)*, in grado quindi di raccogliere, gestire e distribuire contenuti multimediali come immagini, audio e video digitali. Sfruttando le interfacce USB, si possono connettere dispositivi di memorizzazione di massa come hard disk o memorie fisiche portatili (chiavette, memory card, ecc.) e condividerne il contenuto attraverso la rete domestica. Per velocizzare la ricerca di un determinato file attraverso l'azione di *browsing*, al momento della connessione della periferica di memorizzazione viene eseguita una routine in grado di creare un database, di tipo SQL, e memorizzato in un file in locale, contenente gli indici e le proprietà fondamentali dei contenuti (media format, grandezza, *bitrate*, ecc.). La disponibilità delle connessioni Ethernet e Wi-Fi permette infine una rilevante flessibilità di utilizzo, sfruttando la diffusione della prima e la comodità della seconda e la possibilità di creare dei *bridge* di rete con eventuali access point presenti nella rete UPnP.

4 TEST DI CONFORMITÀ ED INTEROPERABILITÀ

In questo capitolo verranno descritti i test che sono stati eseguiti sul dispositivo per il quale l'azienda intendeva implementare le funzionalità UPnP/DLNA. I test sono stati eseguiti sul dispositivo stesso ed esclusivamente attraverso l'interfaccia Ethernet, in modo da poter verificare il corretto funzionamento e la compatibilità con l'hardware a disposizione.

4.1 Test di conformità

L'organizzazione DLNA rilascia agli sviluppatori che intendono certificare il loro prodotto un software di test, il *Conformance Test Tool (CTT)*, con la quale testare il dispositivo in questione. A seconda delle funzionalità implementate, che si tratti quindi di un Media Server piuttosto che di un Media Player o di un Media Controller, si verifica che l'applicazione rispetti le linee guida DLNA almeno nelle proprietà obbligatorie; nei documenti si trovano infatti sia le specifiche che ogni prodotto deve necessariamente possedere per poter essere certificato, sia le funzionalità opzionali utili a rendere più versatile il dispositivo (ad esempio, si possono implementare formati multimediali più avanzati e/o diffusi).

Per dare inizio al test di conformità si è collegato, attraverso interfaccia Ethernet, l'access gateway ad un PC su cui era in esecuzione il software fornito da DLNA (da ora CTT). Una volta accertata l'avvenuta connessione con lo strumento *ping* si è potuto inizializzare il test vero e proprio; per mezzo dell'interfaccia grafica, il CTT dava istruzioni all'utente su come procedere passo-passo: avviare l'applicazione UPnP/DLNA residente sull'access gateway, fermarne e riprenderne l'esecuzione, segnalare le funzioni opzionali implementate, ecc. In particolare, non avendo un'interfaccia con l'access gateway cui si potesse far riferimento per inizializzare o fermare l'esecuzione dell'applicazione, si sono ottenuti gli stessi effetti collegando o scollegando la periferica di memorizzazione usata per il test.

Il processo di test è stato automatico per la maggior parte della sua attuazione, mentre veniva segnalato a video il risultato di ogni passo eseguito, segnalando eventuali errori o incongruenze con i dati inseriti in precedenza. Alla sua conclusione è stato prodotto un file di *log*, visualizzabile per mezzo dello stesso CTT, con i dettagli dei risultati e un resoconto sulle proprietà obbligatorie ed opzionali implementate e conformi alla documentazione. Il file di *log* è quindi indispensabile per eseguire un eventuale *debug* e per ottenere la certificazione, in quanto il documento viene inoltrato all'organizzazione e considerato come una prova preliminare della conformità del dispositivo.

Una volta concluso con successo il test di conformità si è potuto procedere al più importante e determinante test di interoperabilità.

4.2 Test di interoperabilità

Ad una prima osservazione, per garantire la corretta interoperabilità tra gli elementi che andranno a comporre la rete domestica basata sugli standard UPnP/DLNA, apparirebbe necessario eseguire dei test appositi per ogni configurazione; questo vanificherebbe l'obiettivo di rendere immediata e trasparente all'utente finale la condivisione dei contenuti multimediali. In realtà la documentazione fornita da DLNA stabilisce che, una volta completato con successo il test di conformità con il software CTT, il produttore deve sottoporre il dispositivo ad ulteriori prove, descritte con precisione nelle linee guida per il *Certification Test Plan (CTP)*. Sostanzialmente, il prodotto dev'essere testato da altri due dispositivi facenti parte della categoria con cui l'oggetto andrà ad interagire in un'ipotetica rete domestica. Perciò, trattandosi di un Digital Media Server, il dispositivo oggetto del tirocinio è stato messo a confronto con due diversi Digital Media Player certificati DLNA di diversi produttori ed indicati all'interno delle linee guida per il CTP. Oltre a questo, nella documentazione vengono descritti nel dettaglio i passi da seguire per procedere con il test e vengono indicate le funzioni che devono obbligatoriamente risolversi correttamente, come ad esempio la corretta rilevazione del dispositivo (fase di *discovery*) e l'avvio della riproduzione dei contenuti compatibili con il Media Player. Per permettere una certa uniformità nei test eseguiti dai vai produttori, l'organizzazione DLNA mette a disposizione anche una collezione abbastanza variegata di file multimediali comprendenti immagini, clip audio e video caratterizzati da *media format* standard e conformi alla documentazione DLNA.

Il test del dispositivo è stato avviato dopo aver predisposto la configurazione necessaria, rappresentata nella figura sottostante:

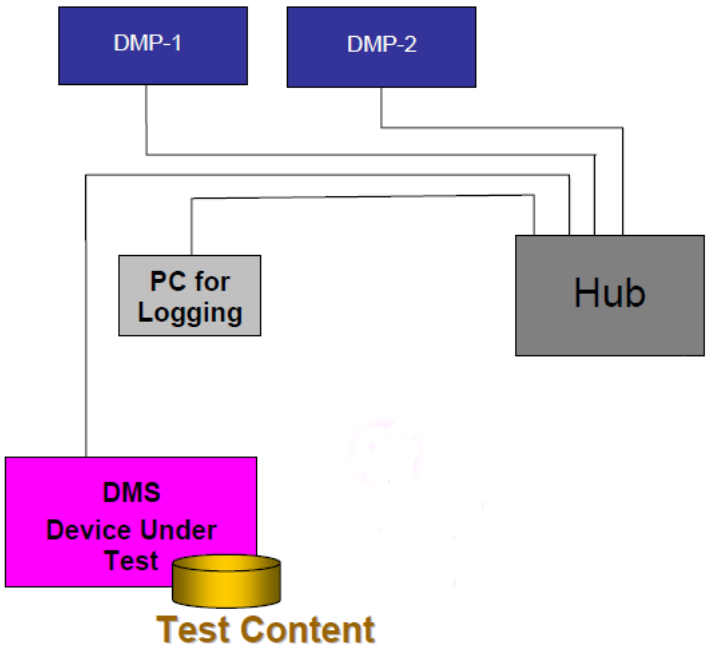

Illustrazione 8: Test bed per il test di interoperabilità

L'access gateway è stato collegato attraverso l'interfaccia Ethernet ad un hub, al quale sono stati connessi due diversi Media Player, specificati all'interno delle linee guida per il CTP. I DMP sono stati collegati infine ad un comune televisore dotato di doppia interfaccia di input, in modo tale da rendere più veloce lo scambio tra un dispositivo e l'altro. E' stata utilizzata inoltre una normale chiavetta USB collegata all'access gateway per memorizzare i contenuti multimediali certificati, i quali comprendevano una serie di immagini in formato JPEG, una raccolta di clip audio codificati in LPCM e MP3 e una collezione di clip video codificati in formato MPEG2-PAL e MPEG2-NTSC.

Per ognuno dei due dispositivi, secondo le linee guida fornite, si sono eseguiti i seguenti passi:

- *Discovery test*: dal Digital Media Player, accedere al menu relativo ai contenuti multimediali in rete e selezionare il Digital Media Server sotto esame.
- *Browse test*: dal DMP, sfogliare la raccolta dei file multimediali contenuti nel DMS; in questo caso, è possibile che non tutti i file a disposizione vengano visualizzati perché incompatibili con il Media Player.
- *Play operation test*: avviare la riproduzione sequenziale e completa di tutti i contenuti compatibili con il DMP; verificare che la riproduzione parta entro un tempo limite e che la qualità sia la stessa riscontrabile alla sorgente.
- *Stop operation test*: se la funzione è disponibile, fermare la riproduzione in corso;
- *Pause/Resume operation test*: se la funzione è disponibile, mettere in pausa la riproduzione per un certo limite prefissato e riprenderne l'esecuzione.
- *Seek operation test*: se disponibili, eseguire le funzioni di *Fast Forwarding* e *Rewinding* in successione da vari punti della riproduzione.

Dei test eseguiti, solamente i primi tre sono considerati obbligatori per ottenere la certificazione, in quanto indicano i requisiti minimi che un dispositivo deve soddisfare per poter garantire l'interoperabilità con gli altri elementi presenti in una rete domestica basata sugli standard UPnP/DLNA. Per quanto riguarda il Digital Media Server sviluppato e testato nel corso del tirocinio, sono stati superati con successo i primi cinque punti con entrambi i Digital Media Player e tutti i contenuti multimediali compatibili. Durante il *Seek operation test* tuttavia, le operazioni di avanzamento veloce e di avanzamento all'indietro hanno presentato diversi problemi, specialmente con alcuni clip video che a volte andavano a bloccare la riproduzione senza la possibilità di recuperarne la posizione. In altre occasioni le funzioni di ricerca non sortivano alcun effetto, segno che probabilmente non sono state implementate dal produttore. Con i clip audio, il *Fast Forwarding* non ha presentato alcun problema se non con alcuni file MP3 codificati con un *bitrate* variabile, una tecnica che permette di bilanciare i bit di codifica a seconda della quantità di informazione presente in ogni unità di tempo. Considerando il fatto che anche alcuni dei clip video sono stati realizzati utilizzando un'analoga tecnologia, si presume che i Media Player utilizzati durante il CTP implementassero una funzione di ricerca di

tipo *byte-seek*, basata ovvero sui byte piuttosto che sul tempo; con i file a *bitrate* costante, è sufficiente infatti conoscere il numero di secondi che si vogliono saltare nella riproduzione per poter ricavare la quantità di byte corrispondente. Nel caso di file con *bitrate* variabile invece è più indicata la ricerca di tipo *time-seek*, con la quale si riescono ad effettuare dei salti nella riproduzione utilizzando le informazioni contenute nell'intestazione (*header*) del file multimediale. I problemi riscontrati durante il *Seek operation test* possono perciò essere dovuti più ad una implementazione non ottimale delle funzioni di ricerca nei Media Player che non ad uno scarso *throughput* fornito dal Media Server sotto test.

5 UPNP-QOS E SVILUPPO DEL CODICE

Nella seconda metà del tirocinio il progetto è stato indirizzato verso l'implementazione delle funzionalità di *Quality of Service (QoS)* applicate alla condivisione multimediale, in modo da migliorare la distribuzione dei contenuti disponibili anche in una rete caratterizzata da diverse connessioni concorrenti spesso presenti in un contesto domestico (trasferimento di file, utilizzo di software *peer-to-peer*, ecc.). Con l'adozione delle funzioni di *QoS* si cerca quindi di dedicare una parte della banda disponibile ad uno o più flussi di traffico che vengono generati dallo *streaming* di un contenuto multimediale, garantendo il *throughput* necessario per ottenere una riproduzione priva di interruzioni.

Le funzionalità di *QoS* implementate nell'access gateway descritto nel progetto seguono le direttive del forum UPnP, precisamente le specifiche relative all'UPnP-QoS. In particolare, nel Digital Media Server è stato implementato il servizio *QosDevice* con il supporto della sola *prioritized QoS* (qualità del servizio di tipo a priorità), per limitazioni hardware (in particolare dello switch Ethernet) che non permette di attuare una *QoS* più raffinata, come quella parametrizzata.

5.1 Il servizio QosDevice

Il servizio *QosDevice* si occupa di tradurre i dati ricevuti dal servizio *QosManager*, comprendenti i parametri della politica di gestione del traffico scelta dal *QosPolicyHolder*, e di tradurli in istruzioni specifiche per la tecnologia di livello *data-link* utilizzata nel dispositivo. Oltre ad effettuare questa traduzione, il servizio *QosDevice* ha la funzione di trasmettere al *QosManager* le caratteristiche del dispositivo quali il suo indirizzo IP/MAC e il suo ruolo nella rete, se è cioè fonte, intermediario o destinatario del flusso di traffico multimediale. E' necessario quindi che tutti gli elementi della rete UPnP per i quali si intende abilitare le funzionalità di *QoS* implementino il servizio *QosDevice*.

Nel dettaglio, la documentazione UPnP prevede la realizzazione obbligatoria di quattro azioni per il servizio *QosDevice*:

- *GetPathInformation()*: restituisce la struttura in linguaggio XML denominata *PathInformation*, la quale contiene l'indirizzo MAC del dispositivo e dei device ad esso connessi, nonché la topologia della rete UPnP.
- *GetExtendedQosState()*: restituisce gli indirizzi IP, le porte e il protocollo utilizzati in una determinata trasmissione; vengono inoltre indicate le capacità di *QoS* (come la banda fisica a disposizione) e l'elenco dei flussi di traffico ai quali è stato applicata una politica di qualità del servizio.
- *AdmitTrafficQos()*: è il metodo con cui il servizio *QosManager* richiede l'ammissione della qualità del servizio ad una trasmissione specificata come input; con questa azione si indica inoltre il tipo di *QoS* che si intende applicare.

- *ReleaseAdmittedQos()*: con questa funzione il servizio *QosManager* può disabilitare la qualità del servizio applicata ad un flusso di traffico specificato.

La scelta di implementare una *Quality of Service* di tipo a priorità ha fatto sì che non fosse necessaria la presenza del servizio *QosPolicyHolder* nella rete UPnP, andando quindi ad utilizzare delle politiche di gestione del traffico di tipo standard e descritte nella documentazione:

Traffic Class	**TrafficImportanceNumber**
Data	0
Background	1
Image	3
Audio, Audio/Video	5
Voice, Gaming	6
Network & Streaming Control	7

Tabella 1: Classi di traffico e priorità

Nel momento in cui si va a richiedere l'ammissione della qualità del servizio desiderata per mezzo dell'azione *AdmitTrafficQos()*, si specificherà come valore d'ingresso anche il *TrafficImportanceNumber*. Nella figura sottostante viene riportato un esempio di richiesta di abilitazione di *prioritized QoS*:

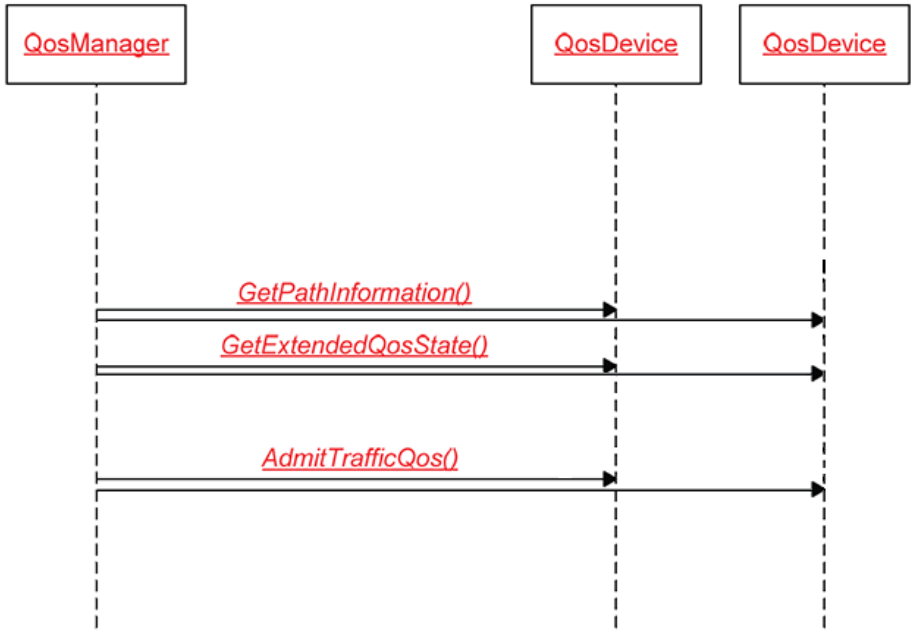

Illustrazione 9: Esempio di ammissione di una prioritized QoS

Il servizio *QosManager*, per mezzo dell'azione *GetPathInformation()*, riceve le informazioni sui dispositivi e sulla topologia della rete organizzate in uno schema XML denominato *PathInformation*, del quale si può trovare un esempio di seguito:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceReachableMacs
xsi:schemaLocation="http://www.upnp.org/schemas/PathInformation.xsd
PathInformation-v3.xsd" xmlns="http://www.upnp.org/schemas/PathInformation.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <LinkReachableMacs>
                <LinkId>eth0</LinkId>
                <MacAddress>112233aabb03</MacAddress>
                <ReachableMac>112233aabb06</ReachableMac>
                <ReachableMac>112233aac206</ReachableMac>
                <ReachableMac>232233aa7126</ReachableMac>
                <ReachableMac>112333aab123</ReachableMac>
                <v2>
                        <v3>
                                <QosSegmentId>174A98172392717512321</QosSegmentId>
                                <ActualLinkId>c8d7aa92c12897</ActualLinkId>
                        </v3>
                </v2>
        </LinkReachableMacs>
</DeviceReachableMacs>
```

Testo 1: Esempio dello schema XML PathInformation

Il campo *LinkId* indica il tipo di interfaccia utilizzato (in questo caso Ethernet), il campo *MacAddress* l'indirizzo MAC del dispositivo e i campi *ReachableMac* indicano gli indirizzi MAC dei device connessi (o raggiungibili). Successivamente, con il metodo *GetExtendedQosState()* il servizio *QosDevice* fornisce le rimanenti informazioni utili al servizio *QosManager* per l'ammissione della qualità del servizio, come ad esempio gli indirizzi IP dei dispositivi interessati, la loro banda fisica disponibile, ecc.

Una volta ricevuti questi dati, il *QosManager* compila uno schema XML denominato *TrafficDescriptor*, nel quale vengono indicate tutte le caratteristiche dei dispositivi coinvolti nella *QoS* acquisite precedentemente; vengono inoltre riportati l'identificatore univoco, detto *TrafficHandle*, associato al flusso di traffico e l'indice di priorità denominato *TrafficImportanceNumber*.

A questo punto, utilizzando lo schema XML *TrafficDescriptor* come parametro d'ingresso per il metodo *AdmitTrafficQos()*, il *QosManager* inoltra la richiesta di ammissione della qualità del servizio ai dispositivi interessati, i quali adatteranno le politiche di gestione del traffico proprie del livello di *data-link* in base alle informazioni ricevute e tradotte in istruzioni dal servizio *QosDevice*.

La qualità del servizio può essere disabilitata con il metodo *RealeaseAdmittedQos()*, utilizzando come parametro d'ingresso l'identificatore *TrafficHandle*; quest'ultimo viene fornito dall'azione *GetExtendedQosState()* che, in presenza di uno o più flussi di traffico dedicati alla qualità del servizio, provvede a compilarne un elenco con le loro caratteristiche principali.

5.2 Sviluppo del codice

L'applicazione che fornisce funzionalità UPnP/DLNA è stata scritta in linguaggio C: questa scelta ha permesso di ottenere un'applicazione in grado di essere compilata ed eseguita sia su un comune PC x86 Linux, sia sull'access gateway, che dispone di capacità di calcolo e risorse di memoria molto più limitate (e non dispone ad esempio di Java Virtual Machine). Utilizzando il compilatore GCC (*Gnu Compiler Collection*) in un ambiente Linux era possibile creare dei file binari eseguibili su un elaboratore comune, con architettura x86, oppure, utilizzando dei cross-compilatori specifici per la CPU dell'access gateway (e le relative librerie e parametri di compilazione) si andavano a creare degli eseguibili che venivano poi caricati nella memoria flash ROM.

La scelta di avere un'applicazione compilabile sia su architettura MIPS che x86 è stata fondamentale per semplificare l'attività di sviluppo, permettendo di utilizzare strumenti di debug più sofisticati, come ad esempio *valgrind*, un utility che permette di rilevare eventuali *memory-leak* durante l'esecuzione del software sotto test.

L'applicazione utilizza librerie OpenSource, come le *libupnp* e *libdlna* per la parte UPnP e DLNA, mentre per il trattamento dei contenuti multimediali sono state utilizzate le librerie *ffmpeg*. Sebbene sia stato realizzato un Digital Media Server, che quindi non esegue la decodifica durante lo *streaming* dei contenuti, essa era comunque necessaria quando si eseguiva l'indicizzazione dei file; venivano così ricavate le informazioni quali durata, *bitrate*, risoluzione, ecc. Per quanto riguarda il database, sono state utilizzate le librerie *sqlite*.

La struttura del codice sviluppato è di tipo modulare, con un blocco principale che implementa il metodo *main* e le funzioni di inizializzazione e terminazione dell'apparato UPnP. Oltre ai moduli ausiliari, tra cui quelli addetti al *buffering* e alla gestione delle proprietà dei file multimediali (ovvero i *meta-dati*), si trovano quelli dedicati all'implementazione dei servizi UPnP-AV (*ContentDirectory* e *ConnectionManager*) e UPnP-QoS (*QosDevice*); precisamente quest'ultimo modulo è stato sviluppato nel corso della seconda metà del tirocinio. La prima attività consisteva nella codifica dei metodi da implementare obbligatoriamente nel servizio *QosDevice*, i quali operano maggiormente con parametri di ingresso e uscita costituiti da schemi XML. Si trattava sostanzialmente di realizzare delle funzioni che, per quanto riguarda i parametri di uscita, compilassero degli schemi XML sotto forma di stringhe con opportuni caratteri di *escape*; utilizzando dei frammenti di stringhe (o *token*) composti da parole chiave proprie della sintassi standard del linguaggio XML e dei metodi messi a disposizione dalle librerie *libupnp*, si è potuto organizzare gli schemi XML in una *struct C* denominata *buffer*. Con l'ausilio del metodo *buffer_append()* si aggiungevano i campi e i relativi valori al *buffer*, il quale veniva successivamente utilizzato come parametro della funzione *upnp_add_response()*, incaricata di inoltrare lo schema XML precedentemente compilato al modulo responsabile della generazione dei pacchetti HTTP da trasmettere nella rete UPnP.

Riguardo i parametri d'ingresso, era necessario realizzare un metodo in grado di estrarre i valori dai campi degli schemi XML ricevuti, così da poter ricavare i dati fondamentali per l'ammissione della qualità del servizio. Utilizzando le librerie *libupnp*, erano disponibili delle funzioni in grado di convertire una stringa opportunamente costruita in un albero di ricerca nel quale ogni nodo costituiva un campo dello schema XML con il rispettivo valore. Venivano inoltre forniti dei metodi attraverso i quali era possibile spostarsi da un nodo dell'albero ad un altro seguendo la struttura gerarchica (perciò funzioni come *getParent()*, *getFirstChild()*, *getNextSibling()*, ecc.). Per eseguire la ricerca, era possibile usare tre approcci: il primo, più semplice concettualmente, era di tipo ricorsivo: per ogni nodo generico, si eseguiva la ricerca tra tutti i suoi figli, dai quali a loro volta veniva iniziata la ricerca in maniera ricorsiva. Per il secondo approccio, di tipo iterativo, era necessario avere il supporto di uno *stack* per memorizzare il livello correntemente visitato, realizzando quindi una sorta di ricorsione esplicita. Il terzo approccio, utilizzato nell'implementazione, utilizzava il metodo *getParent()* per poter tornare indietro una volta arrivati al nodo esterno di un ramo. L'algoritmo usato è riportato di seguito:

```
char *
upnp_get_arg(IXML_Document *doc, const char *key) {

  IXML_Node *root = NULL;
  IXML_Node *node = NULL;

  if (!doc || !key)
  return NULL;

  root = (IXML_Node *) doc;
  if (!root) {
    log_verbose("Invalid argument document\n");
    return NULL;
  }

  root = ixmlNode_getFirstChild(root);
  if (!root) {
    log_verbose("Invalid argument document\n");
    return NULL;
  }

  node = root;
  node = ixmlNode_getFirstChild(node);
  while (node) {
    if (ixmlNode_getNodeName(node) && !strcmp(ixmlNode_getNodeName(node), key)) {
      node = ixmlNode_getFirstChild(node);
      if (!node)
        return strdup("");
      return ixmlNode_getNodeValue(node) ?
        strdup(ixmlNode_getNodeValue(node)) : strdup("");
    }
    if (ixmlNode_hasChildNodes(node))
      node = ixmlNode_getFirstChild(node);
    else {
      while (!ixmlNode_getNextSibling(node) && node != root)
        node = ixmlNode_getParentNode(node);
        node = ixmlNode_getNextSibling(node);
      }
    }

  log_verbose ("Missing argument (%s)\n", key);
  return NULL;
}
```

Testo 2: Algoritmo di ricerca in un albero XML

Partendo dal nodo radice, si percorre il primo ramo fino alla sue foglie; se non è stato trovato il campo cercato, si risale fino a trovare il primo nodo che abbia un fratello, del quale si percorre tutto il ramo fino alle sue foglie. Se ancora non è stato trovato il campo cercato si ripete il procedimento, altrimenti si restituisce il valore corrispondente. Se il campo non esiste, si restituisce una stringa vuota. La complessità temporale di computazione dell'algoritmo, nel caso peggiore, è dell'ordine del numero di nodi dell'albero, il che rappresenta le migliori prestazioni ottenibili da un algoritmo di ricerca senza strutture ausiliarie come uno *heap*.

Per il modulo sviluppato, dovendo memorizzare i dati relativi ai flussi di traffico per cui era stata abilitata la qualità del servizio, è stata necessaria la realizzazione di una serie di strutture di dati; perciò è stata dedicata un'area di memoria a gestione statica, evitando quindi le funzioni *malloc()* e *free()*. In particolare, ogni *struct C* rispecchiava la struttura dello schema XML che si intendeva memorizzare: ad esempio, l'argomento *PathInformation* si componeva di stringhe per quanto riguarda i campi *LinkId*, *MacAddress*, *QosSegmentId* e così via, mentre per i *ReachableMac* è stato utilizzato un vettore di stringhe di dimensione statica.

La seconda attività consisteva nell'implementazione delle istruzioni attraverso le quali il dispositivo potesse predisporre i livelli di rete e di *data-link* all'ammissione della qualità del servizio. Per il Digital Media Server si è scelto di adottare la tecnica della marcatura dei pacchetti (*packet tagging*) delle connessioni abilitate alla *QoS*; in questo modo, realizzando le opportune modifiche ai registri dello switch dedicati alla gestione delle code, è stato possibile attribuire una maggiore priorità alle connessioni caratterizzate da pacchetti marcati.

Quando il servizio *QosManager* invocava il metodo *AdmitTrafficQos()*, veniva richiamata una funzione che andava a configurare opportunamente il *socket* utilizzato per la trasmissione del contenuto multimediale. In particolare la funzione prendeva come argomento l'indice di priorità comunicato dal *QosManager*, e coerentemente a tale valore configurava la *socket option* necessaria per trasmettere pacchetti IP con il *ToS* (il campo *Type of Service* dell'IP header) valorizzato. Quando si rilasciava la *QoS* riservata, la *socket option* veniva resettata, e i pacchetti trasmessi tornavano ad avere *ToS* nullo.

Per quanto riguarda lo switch Ethernet, una opportuna funzione provvedeva a configurare i registri dello switch in modo tale da allocare più code di trasmissione su ciascuna delle porte LAN, e attivare un classificatore del traffico basato sul campo *ToS* dei pacchetti. Questo permetteva ai pacchetti con *ToS* valorizzato di essere accodati nelle code servite a più alta priorità, mentre tutto il resto del traffico finiva nella code a bassa priorità, conferendo al traffico per cui era stata configurata la *QoS* di ottenere tutte le risorse necessarie. Nel momento in cui la *QoS* veniva rilasciata con l'azione *ReleaseAdmittedQos()*, lo switch veniva riconfigurato in modalità base, cioè con unica coda di trasmissione per porta e nessun classificatore attivo, gestendo il traffico in modo indifferenziato.

6 DIMOSTRAZIONE QOS

Come conclusione del progetto sviluppato durante il tirocinio, è stato allestito un banco di prova per verificare il corretto funzionamento del modulo codificato precedentemente; il *test bed* è schematizzato nella seguente figura:

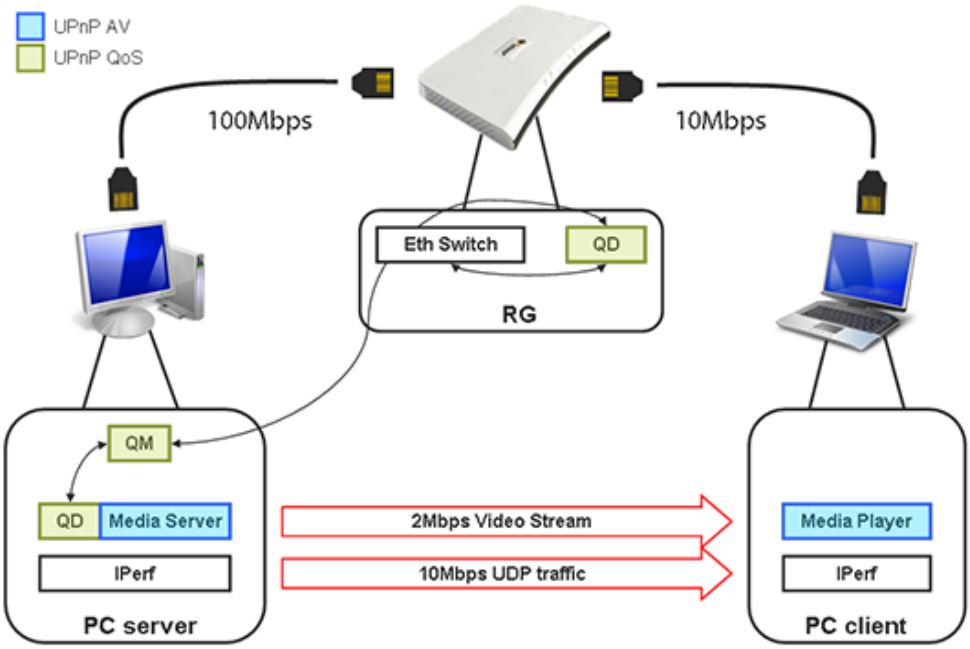

Illustrazione 10: Test bed per la dimostrazione QoS

L'access gateway è stato collegato a due PC, il primo con funzionalità di Digital Media Server e sistema operativo Linux, mentre il secondo fungeva da Digital Media Player, con sistema operativo Windows, per mezzo di *XBMC*, un software media center che include un UPnP-AV media renderer e control point. Nel PC server e nell'access gateway erano in esecuzione l'applicazione UPnP/DLNA sviluppata in Telsey, compreso il modulo del servizio *QosDevice*; oltre a questo, sul PC server girava il software fornito dall'Università di Ghent che implementava il servizio *QosManager*. La banda tra l'access gateway e il PC client è stata limitata a 10Mbps, in modo da creare un collo di bottiglia facilmente saturabile con la trasmissione di contenuti multimediali. Per creare del traffico di disturbo si era inizialmente pensato di utilizzare il trasferimento di un file di grosse dimensioni attraverso una sessione FTP ma, poiché esso utilizza TCP come protocollo di trasporto, venivano attivate le politiche di gestione della congestione proprie del TCP, e quindi l'effetto sul flusso multimediale non era così evidente. Per enfatizzare il problema della congestione e l'impatto che la gestione della *QoS* può avere nel risolvere questo tipo di problemi, si è quindi scelto di utilizzare un flusso UDP come traffico di disturbo, facendo uso di *Iperf*, un programma studiato per misurare le prestazioni massime della banda con i protocolli TCP e UDP.

Durante il test *Iperf* è stato utilizzato per produrre un flusso di traffico UDP di 10Mbps in modo da saturare il collo di bottiglia presente e da evitare meccanismi di controllo della congestione: il protocollo UDP non presenta infatti alcun algoritmo di gestione del traffico, ma anzi occupa tutta la banda disponibile.

Utilizzando *Wireshark*, un programma in grado di analizzare i protocolli di rete delle connessioni in corso, è stato possibile rilevare la banda in entrata del PC client; nella figura sottostante viene riportato il grafico generato durante lo *streaming* di un clip video in formato MPEG2-PAL senza funzionalità di *Quality of Service*:

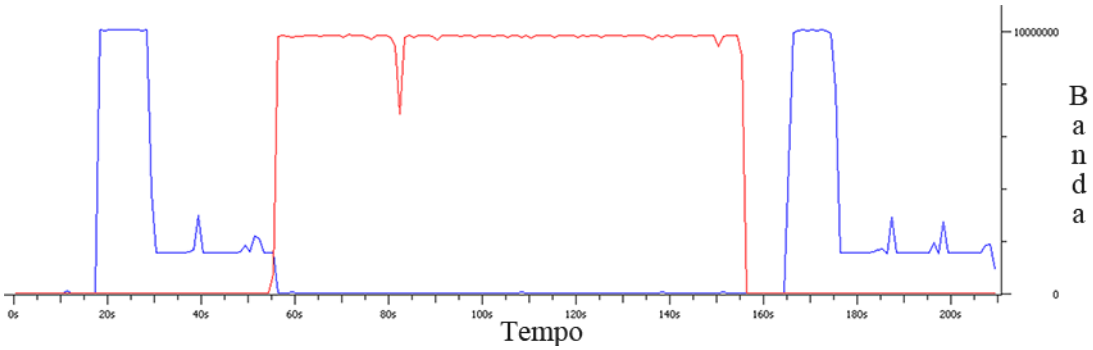

Illustrazione 11: Banda con QoS disabilitata

La linea blu indica il traffico generato dallo *streaming* del filmato (di tipo TCP), mentre la linea rossa rappresenta il traffico di disturbo (di tipo UDP). Come si vede dal grafico, al secondo 17 circa comincia la trasmissione del filmato, la quale va a saturare per un breve tempo tutta la banda a disposizione; questo perché il programma *XBMC* effettua una bufferizzazione iniziale del filmato. Dopodiché la banda utilizzata per lo *streaming* si attesta attorno ai 2Mbps, fino al 55-esimo secondo in cui inizia il traffico generato da *Iperf*: a questo punto la trasmissione TCP collassa a causa del flusso UDP. Ciononostante la riproduzione avanza senza problemi per un certo periodo, cioè fino a quando il buffer del programma *XBMC* si esaurisce: a questo punto viene presentato a video un messaggio che indica una seconda bufferizzazione in corso che di fatto non sta avvenendo. Una volta terminato il traffico generato da *Iperf* si ripresenta la situazione iniziale, con una temporanea saturazione della banda disponibile da parte del flusso TCP ed una successiva stabilizzazione sui 2Mbps.

E' importante osservare che, nonostante si utilizzassero dei PC con diversi sistemi operativi, la rete UPnP era perfettamente utilizzabile e dal programma *XBMC* nel PC client è stato possibile innanzitutto selezionare il Digital Media Server, sfogliarne i contenuti multimediali messi in condivisione e infine avviare e gestire la riproduzione di qualsiasi file presente.

Una volta verificata la saturazione della banda, è stata abilitata la qualità del servizio per mezzo dell'applicazione sviluppata in Java dall'Università di Ghent di cui viene riportata l'interfaccia dalla quale si esegue la richiesta di ammissione per la qualità del servizio:

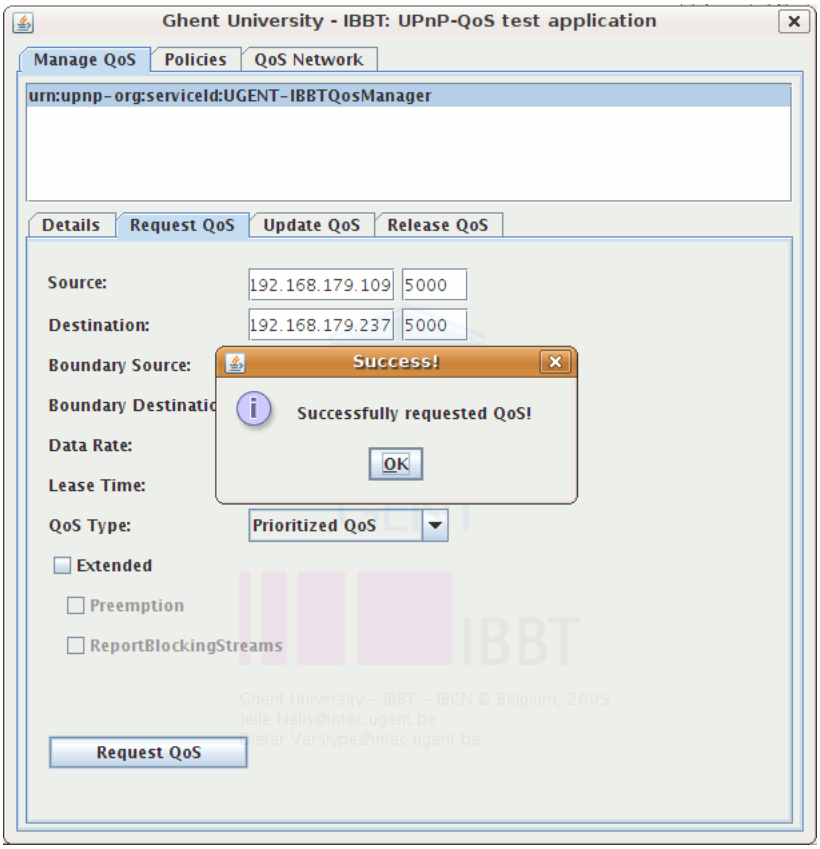

Illustrazione 12: Abilitazione QoS

Nei campi *Source* e *Destination* andavano inseriti gli indirizzi IP di sorgente e destinazione rispettivamente, mentre attraverso il menu *QoS Type* si sceglieva il tipo di qualità di servizio che si intendeva abilitare.

Una volta abilitata la *Quality of Service* è stato rieseguito il test precedente:

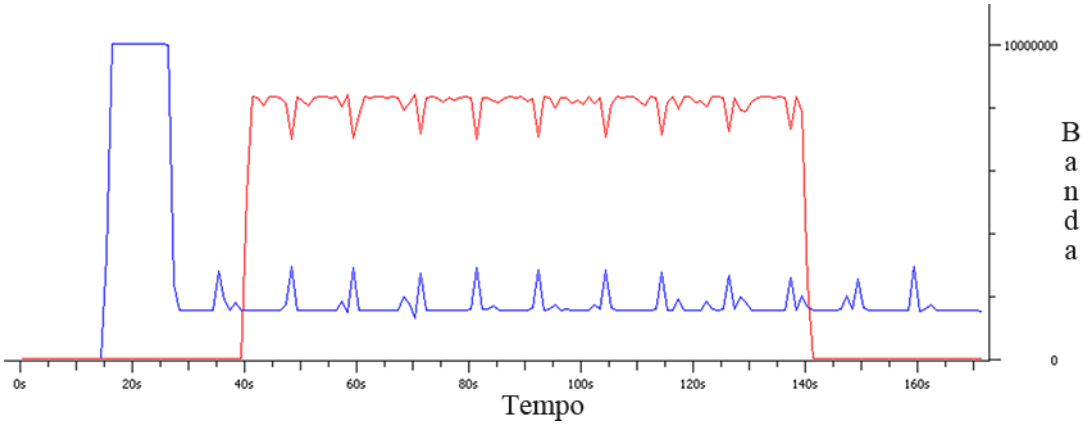

Illustrazione 13: Banda con QoS abilitata

Come nel test precedente, dopo l'iniziale saturazione a causa della bufferizzazione, la banda si stabilizza sui 2Mbps. Quando inizia il traffico generato da *Iperf*, la trasmissione TCP non viene più azzerata ma si conserva in maniera stabile, andando anzi a limitare il flusso UDP; infatti, come si evince dal grafico, la somma istantanea tra la banda dedicata allo *streaming* e quella del traffico di disturbo si attesta sempre a circa 10Mbps, ottenendo così una saturazione più efficiente.

Con questa dimostrazione, seppur limitata dall'utilizzo di prototipi e della sola interfaccia Ethernet, si intendeva provare l'efficacia delle soluzioni fornite dagli standard UPnP e DLNA per la realizzazione di una rete domestica in grado di distribuire in modo semplice una grande varietà di contenuti multimediali, anche in presenza di connessioni concorrenti. Sebbene sia al momento impossibile ottenere un trasferimento veloce di file e al contempo uno *streaming* dei contenuti il più fluido possibile, a meno di non aumentare la banda a disposizione procurandosi dei dispositivi più efficienti, si può ottenere un ottimo compromesso utilizzando le specifiche UPnP-QoS.

Nelle reti domestiche future, caratterizzate da una maggiore disponibilità di banda (Gigabit Ethernet, fibre ottiche polimeriche, ecc.) e da una crescente diffusione di applicazioni *web-based* (o comunque *network-oriented*), la gestione della qualità del servizio potrebbe diventare un punto chiave nella realizzazione di un servizio efficiente.

7 BIBLIOGRAFIA

- UPnP Forum, *http://www.upnp.org/*
- DLNA, Digital Living Network Alliance, *http://www.dlna.org/*
- Microsoft TechNet, *http://technet.microsoft.com*
- Portable SDK for UPnP Devices (libupnp), *http://pupnp.sourceforge.net/*
- DLNA open-source implementation for Linux, *http://libdlna.geexbox.org/*
- Traditional Ways of Tree Traversal, *http://myarch.com/treeiter/traditways*
- Guida al C, *http://programmazione.html.it/guide/leggi/32/guida-c/*
- Il linguaggio C. Principi di programmazione e manuale di riferimento, *Kernighan Brian W., Ritchie Dennis M. - Pearson Education Italia*