Domotics Lab

Una soluzione 'wireless' per il controllo dell'ambiente domestico

Fabio Testa

Vittorio Miori

CONSIGLIO NAZIONALE
DELLE RICERCHE

Indice

1 Contesto dell'applicativo

1.1 La Domotica

La domotica è la disciplina che si occupa dell'integrazione delle tecnologie che consentono di automatizzare una serie di operazioni all'interno della casa. Si occupa anche dell'integrazione dei dispositivi elettrici ed elettronici, degli elettrodomestici, dei sistemi di comunicazione, di controllo e sorveglianza presenti nelle abitazioni. Il termine domotica deriva dall'importazione del neologismo francese *domotique*, a sua volta contrazione della parola latina domus (casa, edificio) e di automatique (automatica, o secondo alcuni informatique, informatica), quindi letteralmente casa automatica.

La domotica ha come oggetto di studio privilegiato l'*home automation* (automazione della casa), una scienza che studia particolari sistemi per automatizzare l'abitazione e facilitare l'adempimento di molte azioni che di solito si svolgono in casa. Con casa intelligente si indica un ambiente domestico opportunamente progettato e tecnologicamente attrezzato al fine di rendere più agevoli le attività all'interno dell'abitazione (quali accensione luci, attivazione e comando elettrodomestici, gestione climatizzazione, apertura di porte e finestre, ecc.) di aumentarne la sicurezza (controllo anti-intrusione, fughe di gas, incendi, allagamenti, ecc.) e di consentire la connessione a distanza con servizi di assistenza (telesoccorso, teleassistenza, telemonitoraggio, risparmio energetico ecc.).

Obiettivo della domotica è aiutare le persone ad abitare in case più sicure e confortevoli, dotate di un sistema di automazione semplice, affidabile, flessibile ed economico; un sistema (teoricamente) alla portata di tutti, con un

comfort nettamente superiore a quello dei sistemi tradizionali e possibilmente a costi simili o inferiori.

Nel caso di grandi edifici (ospedali, aeroporti, ecc.) si parla di *building automation* o automazione degli edifici. L'edificio intelligente, con il supporto delle nuove tecnologie, permette la gestione coordinata, integrata e computerizzata degli impianti tecnologici (climatizzazione, distribuzione acqua, gas ed energia, impianti di sicurezza), delle reti informatiche e delle reti di comunicazione, allo scopo di migliorare la flessibilità di gestione, il comfort, la sicurezza, il risparmio energetico degli immobili e per migliorare la qualità dell'abitare e del lavorare all'interno degli edifici. A differenza della home automation, settore in fase di piena espansione, la building automation è già consolidata da diversi anni, e, anche per questo, è opinione diffusa che prodotti di questo settore adattati su scala ridotta possano essere applicati nella domotica.

Con il termine casa intelligente si definisce l'integrazione di diversi dispositivi per il controllo automatizzato di apparati domestici, di sensori di misurazione dello stato dell'ambiente, di funzioni intelligenti di supporto e di sistemi telecomunicativi per l'accesso alle funzioni da remoto o per l'assistenza a distanza. Un'abitazione così integrata può essere controllata dall'utilizzatore tramite opportune interfacce utente (come pulsanti, telecomando, touch screen, tastiere, riconoscimento vocale), che realizzano il contatto (invio di comandi e ricezione informazioni) con il sistema intelligente di controllo, basato su un'unità computerizzata centrale oppure basato su un sistema a intelligenza distribuita. I diversi componenti del sistema di home automation sono connessi tra di loro e con il sistema di controllo tramite

vari tipi di interconnessione (ad esempio rete locale, onde convogliate, onde radio, BUS dedicato, ecc..). Il sistema di controllo centralizzato, oppure l'insieme delle periferiche in un sistema ad intelligenza distribuita, provvede a svolgere i comandi impartiti dall'utente (ad esempio accensione luce cucina oppure apertura tapparella sala), a monitorare continuamente i parametri ambientali (come allagamento oppure presenza di gas), a gestire in maniera autonoma alcune regolazioni (ad esempio temperatura) e a generare eventuali segnalazioni all'utente o ai servizi di teleassistenza. I sistemi di automazione sono di solito predisposti affinché ogniqualvolta venga azionato un comando, all'utente ne giunga comunicazione attraverso un segnale visivo di avviso/conferma dell'operazione effettuata (ad esempio LED colorati negli interruttori, cambiamenti nella grafica del touch screen) oppure, nei casi di sistemi per disabili, con altri tipi di segnalazione (ad esempio sonora).

Un sistema domotico si completa, di solito, attraverso uno o più sistemi di comunicazione con il mondo esterno (ad esempio messaggi telefonici preregistrati, sms, generazione automatica di pagine web o email) per permetterne il controllo e la visualizzazione dello stato anche da remoto. Sistemi comunicativi di questo tipo, chiamati *gateway* o *residential gateway* svolgono la funzione di avanzati router, permettono la connessione di tutta la rete domestica al mondo esterno, e quindi alle reti di pubblico dominio. Le soluzioni tecnologiche che possono essere adottate per la realizzazione di un sistema domotico sono caratterizzate da peculiarità d'uso proprie degli oggetti casalinghi:

- semplicità:

il sistema domotico è diretto ad un pubblico vasto e non professionale, per questo deve essere semplice da usare secondo modalità naturali, univoche e universalmente riconosciute attraverso un'interfaccia *user friendly*, deve inoltre essere sicuro e non deve presentare pericoli per chi non ne conosce o comprende le potenzialità;

- continuità di funzionamento:

  il sistema deve essere costruito pensando al fatto che dovrà offrire un servizio continuativo e per questo praticamente immune da guasti o semplice da riparare anche per personale non esperto o, nel caso, necessitare di tempi brevi per la rimessa in funzione;

- affidabilità:

  il sistema funziona sempre, senza richiedere particolari attenzioni; anche in caso di guasti esso deve essere in grado di fornire il servizio per il quale è stato progettato o uno simile in caso di funzionamento ridotto, deve essere inoltre in grado di segnalarne il mancato funzionamento e di generare un report delle eventuali anomalie;

- basso costo:

  affinché un sistema domotico sia alla portata di tutti deve avere un costo contenuto, inteso come economicità delle periferiche (sensori,attuatori, ecc.) e della rete di interconnessione tra i diversi moduli funzionali.

Le tecnologie per la domotica permettono inoltre di ottenere alcuni vantaggi quali ad esempio:

- risparmio energetico:

  un sistema completamente automatizzato dovrà evitare i costi generati da sprechi energetici dovuti a dimenticanze o ad altre situazioni, monitorando continuativamente i consumi e gestendo le priorità di accensione degli elettrodomestici;

- automatizzazione di azioni quotidiane:

  un sistema di home automation deve semplificare alcune azioni quotidiane, soprattutto quelle ripetitive, non deve in alcun modo complicarle.

Tutte queste caratteristiche, se non sviluppate singolarmente ma nel loro insieme, portano alla creazione di un sistema di home automation integrato che può semplificare la vita all'interno delle abitazioni. La casa diventa intelligente non perché vi sono installati sistemi intelligenti, ma perché il sistema intelligente di cui è dotata è capace di controllare e gestire in modo facile il funzionamento degli impianti presenti. Attualmente le apparecchiature tecnologiche sono poco integrate tra loro e il controllo è ancora ampiamente manuale, nella casa domotica gli apparati sono comandati da un unico sistema automatizzato che ne realizza un controllo intelligente.

1.2 Stato dell'arte

Nel corso degli ultimi dieci anni c'è stato un continuo proliferare di sistemi dedicati all'automazione della casa e per certi versi questo è stato un fatto positivo: è cresciuta l'offerta, la concorrenza ha portato ad un abbassamento

dei prezzi e tutti i soggetti coinvolti nella filiera, in qualche modo, si occupano di domotica.

Nel frattempo però, è diventato sempre più difficile orientarsi in questo settore ed i progettisti e gli installatori faticano a districarsi in un mercato pieno di soluzioni attraenti sul piano dei costi ma poco documentate sul piano della qualità e dell'affidabilità. Certo si può sempre ricorrere ai sistemi definiti *standard* che offrono ampie garanzie sul piano dell'affidabilità e delle prestazioni, ma diventa difficile realizzare soluzioni a costi contenuti e spesso è necessario avere una buona preparazione tecnica per poterli utilizzare. I sistemi dedicati all'automazione domestica possono essere suddivisi in due categorie, quelli di tipo *proprietario* e quelli *aperti*.

I sistemi proprietari sono quelli realizzati da un singolo costruttore che non ha interesse a divulgare le informazioni sul funzionamento dei propri apparati e ne rende impossibile la costruzione a terzi. Spesso si tratta di sistemi che hanno il loro punto di forza nel costo contenuto e di solito permettono di realizzare con facilità impianti medio/piccoli con ottime funzionalità e buone prestazioni. Quando sono realizzati da aziende di un certo livello, offrono un'ampia scelta di dispositivi, linee estetiche assai accattivanti e vantano migliaia di installazioni. Resta l'incognita del prodotto *single-source* che crea una sorta di dipendenza del cliente dal costruttore e che potrebbe risultare fastidiosa in certe situazioni ma anche in questo caso, se ci si orienta verso aziende di un certo prestigio, i rischi saranno limitati.

I sistemi aperti, che a volte vengono anche definiti standard, sono invece quelli dove le specifiche di funzionamento degli apparati vengono rese pubbliche in modo che qualsiasi azienda possa decidere di sviluppare dispositivi

conformi e quindi interoperanti con tutti gli altri. Spesso si tratta di associazioni o consorzi di aziende di grandi dimensioni (a volte multinazionali) che, sfruttando le proprie competenze, creano dei veri e propri riferimenti di mercato, definiti appunto standard. Quasi sempre si tratta di sistemi molto evoluti che permettono di realizzare impianti medio/grandi e che sono in grado di soddisfare qualsiasi esigenza di automazione.

I vantaggi di questi sistemi sono l'affidabilità, l'interoperabilità dei dispositivi, la scalabilità delle soluzioni e la quasi totale indipendenza dal singolo costruttore.

Inoltre, i sistemi domotici possono essere suddivisibili in base alla loro architettura (cioè in base al modo in cui sono collegati e comandati i vari dispositivi) ed in base a dove è localizzata la capacità decisionale del sistema.

Si possono distinguere tre tipi di architettura:

- architettura centralizzata
- architettura distribuita
- architettura mista

Nell' architettura centralizzata esiste un'unica unità di decisione (tipicamente una centralina) che a seconda dei casi, può essere suddivisa in più unità intelligenti, può avere diversi livelli gerarchici e a volte è anche distribuita fisicamente con una logica master/slave. I dispositivi distribuiti in campo possono essere dotati di una propria capacità di autodiagnosi e di autoconfigurazione ma non sono in grado di prendere decisioni, delegando questa funzione alla centralina. In questo scenario, i messaggi provenienti

dai sensori e diretti agli attuatori sono sempre elaborati dall'unità centrale che, in base ad un programma residente nella sua memoria, prende le decisioni del caso. I vantaggi dell'architettura centralizzata sono la facilità di programmazione e di riconfigurazione, il minor costo ed il buon livello di integrazione delle funzioni.

Nell'architettura distribuita invece, tutti i dispositivi sono intelligenti e sono quindi in grado di eseguire una serie di funzioni in modo totalmente autonomo. Ad ogni componente in campo viene prima assegnato un indirizzo univoco (per poterlo identificare nella rete) e poi viene programmato per eseguire determinati comandi e svolgere certe funzioni. In questo tipo di architettura, i dispositivi in campo si parlano tra di loro attraverso una messaggistica standardizzata che permette di creare dei collegamenti (associazioni) di tipo logico tra i componenti senza particolari limiti.

Un'architettura distribuita invece, presenta vantaggi di grande flessibilità, affidabilità (se si guasta infatti un dispositivo si perderanno le funzioni associate a quel dispositivo ma il resto del sistema continuerà a funzionare) e prestazioni che però vanno a scapito dei costi e di una maggiore complessità in fase di programmazione.

Nell'architettura mista esiste un cablaggio principale a livello di fabbricato che ha caratteristiche simili all'architettura distribuita che poi si collega, attraverso delle interfacce, ad una serie di centraline dislocate nell'edificio. Si tratta di un'architettura molto efficiente e performante che però trova la sua applicazione nell'automazione di grandi edifici con esigenze particolari. Indipendentemente dal tipo di architettura, il collegamento tra i dispositivi in campo viene generalmente effettuato attraverso delle linee dedicate in rame

od in fibra ottica. Il vantaggio del linee in rame è rappresentato dalla facilità di posa in opera e dal basso costo mentre con la fibra ottica si ottengono grandi velocità di comunicazione, un'ottima immunità ai disturbi, ma a discapito del costo e della facilità di montaggio. Esistono poi sistemi che sfruttano l'impianto elettrico esistente per comunicare tra di loro (powerline) ed altri sistemi, sempre più diffusi, che si collegano via radio con tecnologia wireless.

1.3 I middleware domotici

Con il termine inglese *middleware* si intende un insieme di applicativi software che fungono da intermediari tra diverse applicazioni.

I middleware domotici si possono dividere in due grandi classi che differiscono sostanzialmente per il sistema di comunicazione usato: una, più tradizionale, fa uso di un sistema bus; l'altra,più recente, si basa su un insieme di protocolli più evoluto come TCP/IP. Del primo tipo fanno parte standard oramai consolidati nel mondo della domotica, quali ad esempio X10, EIB, BatiBus, EHS, Konnex, LonWorks, CEBus; la seconda tipologia invece include, tra gli altri, standard come UPnP, Jini, OSGi.

A fianco di un particolare sistema di comunicazione, è disponibile un numero sempre maggiore di standard che si basano su mezzi trasmissivi diversi, come IEEE 802.11b (Wi-Fi), Bluetooth, IEEE 1394 (Firewire), IrDA, ZigBee, etc. Andiamo ora ad analizzare due tra i principali standard appartenenti alle due classi di middleware: *Konnex* per quanto riguarda i *sistemi bus*, e *UPnP* per quelli basati su *TCP/IP*.

Andremo a vedere nel dettaglio questi due casi particolari di middleware

domotici, che per molti aspetti si collocano agli antipodi, e proprio per questo motivo sono stati scelti per domostrare la validità di **domoNET** nell'implementare l'interoperabilità; è quindi interessante vederne alcuni aspetti per poter poi capire il ruolo chiave di domoNET, di cui discurremo più avanti, in un panorama variegato come la domotica.

1.3.1 Konnex

Konnex[1] è la risultante di un processo di convergenza tra EIB, BatiBus e EHS. Nel 1997 venne raggiunto un accordo tra i principali produttori di sistemi domotici per la realizzazione di un unico sistema bus, che costituisse l'unico standard Europeo in ambito di automazione degli edifici.

Lo scopo della convergenza era quello di incrementare il mercato della domotica, definendo un protocollo comune per l'automazione e assicurando l'interoperabilità dei tre sistemi. Nel febbraio del 1999, otto aziende (tra cui Siemens, Bosch,Merten, Hager e Schneider) rilasciarono le specifiche di Konnex come singola forza europea per la standardizzazione delle reti domotiche.

Architettura

Il sistema Konnex fornisce le basi per lo sviluppo di applicazioni che si occupano di gestire l'ambiente domestico e, più in generale, quello di qualsiasi edificio.

Una premessa fondamentale per raggiungere questo obiettivo è quella di rappresentare l'ambiente come un sistema in cui ogni dispositivo costituisce

[1] http://www.konnex.org/

una vera e propria applicazione distribuita: per questo motivo si parla di rete Konnex.

Come avviene per gli standard da cui deriva, anche le specifiche della rete Konnex seguono il modello architetturale standard ISO/OSI[2] per definire gli strati, i protocolli e le funzionalità offerte ad ogni strato. In particolare le specifiche definiscono 5 dei complessivi 7 strati del modello ISO/OSI.

Lo standard Konnex specifica molti meccanismi per rendere operativa la rete, consentendo così ai produttori una certa flessibilità nello sviluppo dei propri dispositivi.

Di seguito (figura 1) è riportata una panoramica del sistema Konnex che evidenzia sia le caratteristiche obbligatorie per la conformità allo standard, sia le scelte aperte lasciate al produttore. Lo schema non descrive i protocolli e le scelte tecniche, ma piuttosto individua quegli aspetti necessari per lo sviluppo di dispositivi e componenti.

Alla base del concetto di applicazione proposto da Konnex c'è l'idea dei *datapoint*. Con questo termine vengono indicate tutte le variabili di processazione e controllo dell'intero sistema. I datapoint possono costituire input, output, parametri, dati di tipo diagnostico, ecc... e sono contenuti in *Group Objects* e in *Interface Object Properties*.

Il sistema di comunicazione del modello Konnex deve fornire un insieme di istruzioni ridotto ma completo per la gestione dei datapoint; tipicamente queste istruzioni consentono la lettura e la scrittura dei valori contenuti nei datapoint.

Per garantire l'interfunzionalità all'interno del sistema, i datapoint de-

[2]http://www.iso.org/

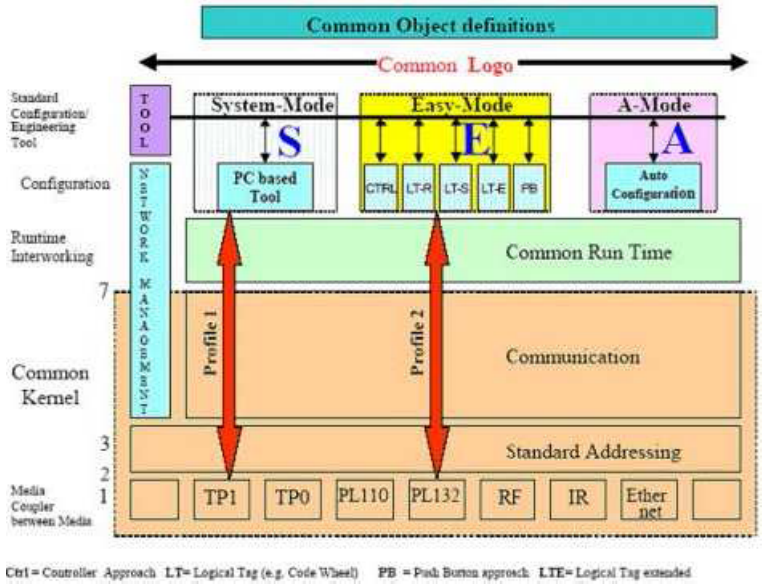

Figura 1: Il sistema Konnex.

vono implementare dei tipi standardizzati di dato, chiamati *datapoint type*, raggruppati in cosiddetti *functional block*(blocchi funzionali). Spesso i functional block e i datapoint type dipendono dal dominio applicativo, ma alcuni di essi possono essere di utilità generale (si pensi, ad esempio, alla codifica della data e del tempo all'interno di un timer).

L'accesso ai datapoint può seguire uno schema di tipo *unicast* o *multicast*, mentre il collegamento logico tra i datapoint delle applicazioni distribuite sulla rete, avviene tramite tre cosiddetti modelli di binding: *free*, *structured* e *tagged*.

Descriviamo ora come questi modelli di collegamento possono essere combinati con i vari meccanismi di indirizzamento.

Durante il processo di installazione del sistema Konnex ci sono due fasi che richiedono un'attività di configurazione. La prima fase riguarda la config-

urazione della topologia della rete e dei singoli dispositivi che ne costituiscono i nodi; nella seconda invece si vanno a configurare le applicazioni distribuite sulla rete, comprendendo il collegamento tra i datapoint e il settaggio dei relativi parametri.

Complessivamente, una particolare installazione del sistema Konnex ha i seguenti obiettivi:

- evidenziare un preciso schema di configurazione e collegamento;
- mappare questo schema nella scelta di un adeguato schema di indirizzamento;
- scegliere un insieme di procedure specifiche per la gestione delle risorse del sistema.

Alcune modalità di configurazione richiedono un processo di gestione della rete che coinvolge una comunicazione attiva sul bus (sia di tipo *peer-to-peer* che di tipo *master-slave*), mentre altre si limitano ad un'attività locale al dispositivo stesso.

Affinché sia praticabile un tipo di configurazione attiva, il sistema Konnex è equipaggiato con un potente insieme di strumenti per la gestione delle risorse della rete. Con il termine risorse si intendono sia dati relativamente semplici come indirizzi e parametri per la comunicazione, sia dati più complessi come tabelle di binding o veri e propri programmi applicativi.

Le attività correlate alla gestione della rete costituiscono un insieme di meccanismi per la scoperta, il settaggio e la ricezione, attraverso la comunicazione con il bus, dei dati necessari alla configurazione del sistema. Vengono

impiegate le cosiddette *procedures* (procedure), ossia sequenze di messaggi, che permettono di accedere ad alcune risorse della rete esposte dai dispositivi e garantire così il corretto funzionamento di tutti i componenti della rete Konnex.

Il modulo di gestione della rete sfrutta i servizi offerti dallo strato *Application* dello schema ISO/OSI; ogni dispositivo che implementa una particolare modalità di configurazione deve anche implementare i servizi e le risorse specifiche per un certo profilo.

Lo standard Konnex offre un ampio spettro di scelta ai produttori di componenti per quel che riguarda il supporto dei mezzi trasmissivi.

I mezzi trasmissivi supportati da questo middleware sono TP0 e TP1, PL110 e PL132, RF e in ultimo IR; inoltre Konnex offre delle soluzioni standard ed integrate per gestire il protocollo IP, quali Ethernet (IEEE 802.3), Bluethoot, Wi-Fi/Wireless LAN (IEEE 802.11), FireWire (IEEE 1394).

Sulla base dello strato fisico e *Data-link*, tutti i componenti della rete Konnex condividono un cosiddetto *Common Kernel* (nucleo comune) che si basa sul modello ISO/OSI a 7 strati:

- data-link general :

  posto al di sopra dello strato Data-Link per medium, fornisce i meccanismi per il controllo dell'accesso al mezzo e del canale logico;

- network layer :

  offre un meccanismo di *acknowledgement* dei frames e ne controlla il numero di *hop count* (questo strato risulta particolarmente interessante per quei nodi che, all'interno della rete, hanno funzionalità di routing);

- transport layer :

  specifica 4 modelli di comunicazioni tra le entità della rete:uno-a-molti senza connessione (*multicast*), uno-a-tutti senza connessione (*broadcast*), uno-a-uno senza connessione e uno-a-uno orientato alla connessione;

- application layer :

  fornisce un vasto insieme di servizi, dedicati ai processi applicativi, che variano a seconda del tipo di comunicazione usato allo strato del trasporto.

Come evidente, gli strati *session layer* e *presentation layer* in questo caso sono vuoti. I servizi relativi ai modelli di comunicazione *punto-a-punto* e *broadcast* servono principalmente per la gestione della rete, mentre quelli relativi a comunicazioni multicast supportano le operazioni di runtime.

Come già visto, la gestione della rete consiste in un insieme di procedure per manipolare le risorse, mentre il common kernel fornisce un framework di servizi per adempiere tale proposito. A causa del ruolo centrale che assumono all'interno del sistema, le risorse di rete meritano una descrizione più approfondita.

Si possono distinguere risorse per il controllo delle applicazioni e risorse di sistema (o di configurazione), che comprendono indirizzi e parametri informativi con lo scopo di agevolare le attività di comunicazione.

I cosiddetti Interface Objects forniscono un importante framework, indipendente dall'implementazione, per definire le risorse del sistema, i cui elementi individuali possono essere modellati come datapoint.

Da un punto di vista generico, l'installazione del sistema Konnex consiste semplicemente in un insieme di dispositivi connessi tra loro attraverso il bus. Tutte le funzionalità finora descritte sono realizzate in e per mezzo di questi dispositivi. A questo proposito si possono distinguere vari modelli logico-architetturali che un nodo della rete deve rispettare.

Questi modelli variano in accordo con le principali caratteristiche del nodo, quali la gestione, la modalità di configurazione e il suo ruolo all'interno della rete; per quanto riguarda quest'ultimo aspetto si possono distinguere *application device*, *configuration master*, *router*, *gateway*, ecc.

Il sistema Konnex standardizza anche alcuni modelli di dispositivi che rivestono ruoli più generali all'interno della rete (*general-purpose device*), come Bus Coupling Unit (BCU) o Bus Interface Module (BIM), impiegati principalmente come interfaccia verso il bus.

I modelli di dispositivi, insieme alle caratteristiche delle modalità di configurazione, sono complessivamente rappresentati all'interno dei profiles. Vi possono essere diversi meccanismi di identificazione per accedere ai dispositivi presenti all'interno della rete: attraverso un indirizzo individuale o un numero di serie univoco proprio del dispositivo, in base alla modalità di configurazione; attraverso altri tipi di informazioni direttamente ricavabili dal dispositivo, come l'identificativo del prodotto (dipendente dal produttore) o l'identificativo delle funzionalità offerte (indipendente dal produttore).

L'unicità dei numeri seriali è garantita da Konnex Association Certification Department[3].

[3]http://www.konnex.org/knx-association/introduction/

1.3.2 UPNP

Il middleware UPnP costituisce uno dei più importanti esempi di tecnologie domotiche basate sullo stack dei protocolli TCP/IP.

A differenza dei sistemi bus analizzati finora, UPnP non necessita di componenti hardware particolari, ma sfrutta i comuni meccanismi di accesso alla rete IP per costituire il proprio modello di comunicazione.

La tecnologia UPnP definisce un'architettura per lo sviluppo di reti peer-to-peer tra *dispositivi wireless*, apparecchiature domestiche ed elaboratori generici.

I protocolli utilizzati da UPnP sono quelli standard di Internet come IP, TCP,UDP, HTTP, XML e SOAP; questo consente il supporto di qualsiasi mezzo fisico per cui sia disponibile lo stack TCP/IP, e lo sviluppo di software attraverso molteplici linguaggi di programmazione, su qualunque ambiente operativo implementi TCP/IP. Alla base dell'organizzazione UPnP si trova

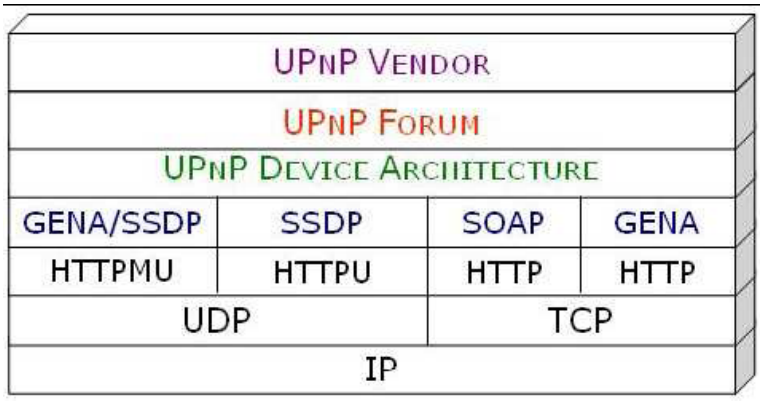

Figura 2: Struttura del framework UPNP.

la cosiddetta UPnP Device Architecture[4], ossia l'interfaccia comune verso i

[4]http://www.upnp.org/

protocolli standard definiti dalle specifiche.

Le attività proposte da UPnP sono regolamentate da UPnP Forum, un'iniziativa guidata da Microsoft e da oltre 700 partner industriali. Le scelte promosse dai singoli produttori vengono gestite da UPnP Vendor. In figura 2 viene mostrato lo stack dei protocolli utilizzati dallo standard UPnP.

UPnP identifica due classi di dispositivi: *controlled device*(o semplicemente device) e *control point*.

Un controlled device svolge il tipico ruolo di un server, rispondendo alle richieste provenienti da uno o più control point, che assume quindi il ruolo di client.

Entrambe le categorie di dispositivi possono essere implementate su varie piattaforme, come PC, PDA o sistemi embedded. L'interazione generale in uno scenario UPnP prevede 6 fasi distinte (*addressing*, *discovery*, *description*, *control*, *eventing* e *presentation*), che andiamo ora a analizzare nel dettaglio.
Fasi di interazione

1. addressing:

   alla base del funzionamento di una rete UPnP c'è l'indirizzamento IP. Ogni dispositivo deve essere provvisto di un client *DHCP* (Dynamic Host Configuration Protocol): quando un dispositivo viene collegato alla rete per la prima volta, deve ricercare un server DHCP che provveda ad assegnargli un indirizzo IP; se la rete non è configurata e non vi sono server DHCP disponibili, il dispositivo utilizzerà il meccanismo di *AutoIP* per ottenere un indirizzo valido;

2. discovery:

quando il dispositivo si è collegato alla rete UPnP ed ha ottenuto un indirizzo IP valido, pubblica i servizi che espone verso tutti i control points presenti. Allo stesso modo, quando un control point viene aggiunto alla rete, potrà ricercare tutti i dispostitivi a cui è interessato. Entrambe le interazioni sono veicolate da *SSPD* (Simple Service Discovery Protocol), il protocollo di discovery sviluppato dallo standard;

3. description:

   dopo che un control point ha scoperto la presenza di un dispositivo, può conoscerne le caratteristiche ottenendo la sua descrizione in formato XML, a partire dalla URL (Uniform Resource Locator) fornita dallo stesso dispositivo durante la fase di discovery. La struttura di un dispositivo è gerarchica: l'elemento principale è il cosiddetto root device che, al suo interno, può contenere cosiddetti *embedded device*(dispositivi logici); ogni embedded device racchiude alcune funzionalità che espone verso la rete sotto forma di *services*(servizi). La descrizione XML del dispositivo è suddivisa in due parti: la prima contiene informazioni relative al produttore (ad es. il marchio, il numero seriale, la URL della fabbrica, etc.); la seconda specifica i dettagli di ogni servizio esposto da quel particolare dispositivo (ad es. la lista dei comandi previsti per quel servizio, il numero ed il tipo di parametri ammessi, etc.);

4. control:

   quando il control point ha ottenuto tutte le informazioni necessarie per un certo dispositivo, può invocare le azioni corrispondenti ai servizi che

questo espone. L'interazione avviene tramite lo scambio di messaggi SOAP (Simple Object Access Protocol), seguendo un modello RPC (Remote Procedure Call);

5. eventing:

   la descrizione di un dispositivo conforme allo standard UPnP contiene, tra l'altro, una lista di variabili che ne modellano lo stato interno. A seguito di particolari e contingenti condizioni, queste variabili possono cambiare valore e il servizio a cui si riferiscono pubblica alla rete questi aggiornamenti. Un control point deve sottoscriversi ad un meccanismo di eventi per ricevere le notifiche dei cambiamenti delle variabili di stato appartenenti ad un certo dispositivo. I messaggi di evento sono anch'essi espressi secondo il formalismo standard XML e veicolati dal protocollo GENA (General Event Notification Architecture);

6. presentation:

   il control point può avere bisogno di ottenere informazioni aggiuntive su un particolare dispositivo; tali informazioni, se presenti, sono contenute in una pagina web indicata in un'opportuna URL nel file XML di descrizione del dispositivo.

Attualmente è in corso di rilascio UPnP Device Architecture 2.0, che si baserà totalmente sui protocolli standard definiti dal framework dei *Web Services*[5].

[5] http://www.w3.org/2002/ws/

1.4 L'applicativo domoNET

La principale problematica della Domotica è quindi senza dubbio l'interoperoperabilità tra gli standard domotici esistenti.

Il Laboratorio di Domotica del CNR di Pisa, ha di recente concepito e sviluppato un'architettura in grado di far comunicare alcuni degli standard piu'diffusi nel campo della domotica. Abbiamo infatti visto nel capitolo precedente le caratteristiche di due importanti middleware domotici, utili non solo per comprendere le enormi differenze tra gli standard esistenti, ma anche perché proprio Konnex e UPNP sono stati scelti per la realizzazione del prototipo di domoNET, che ne ha dimostrato l'interoperabilità.

Il cuore dell'architettura, realizzata tramite l'applicativo domoNET, è il linguaggio *domoML* [6]; questo linguaggio, formalizzato tramite una grammatica XML, permette di astrarre dispositivi e servizi appartenenti ad un determinato standard in modo da poter gestire tutte le reti domotiche connesse a domoNET come fossero uno standard unico.

In breve domoNET, un applicativo freeware rilasciato con licenza GPL che utilizza strumenti e standard assolutamente open-source di cui parleremo brevemente[7], si candida a diventare un punto di incontro tra tutti gli standard esistenti e, viste le premesse di generalità alla base dell'architettura, persino per gli standard avvenire.

Inoltre la soluzione del problema dell'interoperabilità dovrebbe dare finalmente l'impulso giusto alla diffusione su larga scala della domotica nelle

[6]cfr: 'Una grammatica XML per la Domotica' (Fabio Testa, Vittorio Miori)

[7]cfr. Tesi di Laurea di Dario Russo *La domotica e Internet. Una soluzione per l'interoperabilità*

nostre case, vista la possibilità di avere libera scelta (e quindi prezzi più competitivi) nell'acquisto dei prodotti, *conditio sine qua non* il mercato dei dispositivi domotici sarà sempre relegato ad una nicchia.

Di seguito metteremo in luce alcuni aspetti di domoNET utili a comprendere il contesto in cui si collocano gli applicativi realizzati in questa tesi.

1.4.1 Architettura

Per comprendere la necessità di un *framework* che garantisse l'interoperabilità tra i diversi sistemi domotici presenti sul mercato, il Laboratorio di Domotica è partito da uno scenario generico. In un ipotetico ambiente in cui coesistono alcuni tra i principali middleware domotici, è possibile identificare ogni sistema come una particolare *sottorete*, pur senza soffermarsi sui dettagli e sulle infrastrutture necessarie.

Ogni nuvola in figura 3 rappresenta un middleware, ovvero una sottorete all'interno della quale sono presenti sia le infrastrutture hardware e software per il supporto, sia i dispositivi conformi. Tutti i dispositivi all'interno della stessa sottorete colloquiano e cooperano per costituire un sistema funzionante ed indipendente, ma chiuso.

Di fatto è impossibile controllare un dispositivo che si trova in una nuvola tramite un altro presente in una nuvola diversa, poiché non esiste nessun tipo di collegamento tra le due sottoreti.

Per collegamento si intende un'infrastruttura logica che consenta ai dispositivi di qualsiasi sottorete di conoscere tutti i dispositivi presenti nelle altre sottoreti.

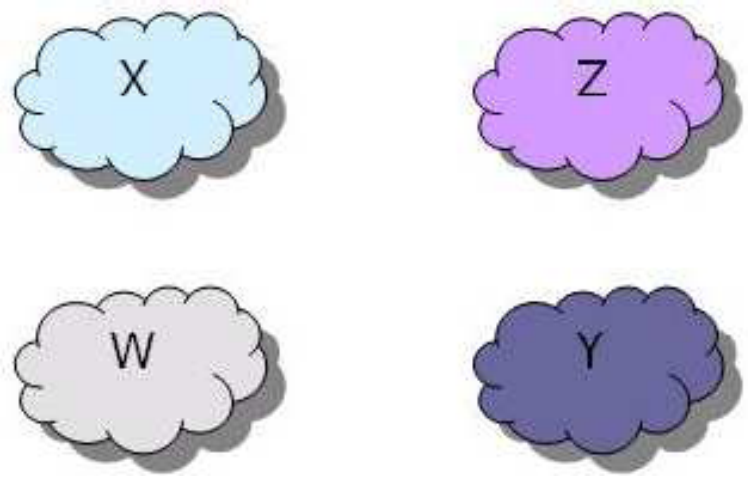

Figura 3: Rappresentazione dei middleware domotici.

Infine, anche qualora fosse possibile avere una visione completa dell'intera rete, sarebbe comunque necessario un meccanismo comune per lo scambio di informazioni tra i dispositivi.

Vi sono quindi due aspetti da risolvere: da un lato riuscire a costruire un'infrastruttura logica di collegamento tra i vari middleware, dall'altro mettere a punto un meccanismo di comunicazione universale veicolato da quest'infrastruttura.

Una possibile soluzione a questi due problemi è quella di introdurre tra ogni coppia di sottoreti un modulo hardware e/o software che sia provvisto di un'interfaccia verso entrambe le nuvole.

Ogni modulo, che nella terminologia generale è detto *gateway*, deve non solo fornire funzionalità di traduzione tra i due protocolli delle sottoreti che collega, ma anche conciliare le differenze tra i paradigmi di comunicazione su cui si basano i due sistemi.

Quest'ultima necessità risulta evidente quando i sistemi domotici che si

desidera collegare presentano caratteristiche notevolmente differenti (ad es. middleware a *configurazione manuale(Konnex)* vs. *plug and play(UPNP)*).

L'implementazione di un particolare gateway non può dunque prescindere da una conoscenza approfondita di entrambi i sistemi domotici da collegare.

Questa soluzione, tuttavia, risulterebbe poco scalabile a causa del numero di gateway necessari al crescere dei sottosistemi da collegare: infatti, all'aumentare del numero delle sottoreti, il numero di gateway da sviluppare cresce sensibilmente.A questa prima soluzione ne abbiamo preferita una seconda,che interviene ad un livello superiore.

Per riconoscere la validità di questo approccio e la sua applicabilità, si è partiti da un'analogia che deriva direttamente dal mondo delle reti di calcolatori.

In passato, molti costruttori hanno sviluppato infrastrutture *ad hoc* per la costituzione di reti che permettessero la comunicazione tra piattaforme e periferiche dello stesso marchio. Proprio per questo motivo, vi sono stati tentativi da parte di singole aziende, peraltro falliti, di imporre la propria infrastruttura come unico standard.

Di fatto nessuna infrastruttura ha mai raggiunto un'egemonia sulle altre, principalmente a motivo dello sviluppo di un'infrastruttura innovativa in grado di superare il vincolo della unicità di piattaforma: questa infrastruttura, di ispirazione ISO/OSI, è il noto stack di protocolli TCP/IP.

Seguendo lo stesso approccio in contesto domotico, si è pensato di introdurre un'ulteriore nuvola atta a gestire i diversi sistemi domotici al di là delle differenze fra essi, permettendo così la comunicazione tra nodi appartenenti a sottoreti distinte.

Questa nuova infrastruttura, denominata **domoNET**, si basa sui *Web Service* e fornisce una visione dell'intera topologia della rete secondo un modello SOA[8].

Per garantire l'interoperabilità tra nodi comunicanti attraverso l'infrastruttura domoNET, è necessario sviluppare, oltre ad opportuni gateway, una grammatica XML standard detta *domoML*. Rispetto alla prima soluzione proposta, questa è certamente più scalabile, in quanto richiede solamente un gateway per ogni sistema domotico da collegare a domoNET.

Ogni gateway, che d'ora in avanti chiameremo *techmanager*, dovrà possedere da un lato un'interfaccia verso il particolare sistema domotico a cui si riferisce, e dall'altro un'interfaccia verso domoNET (figura 4).

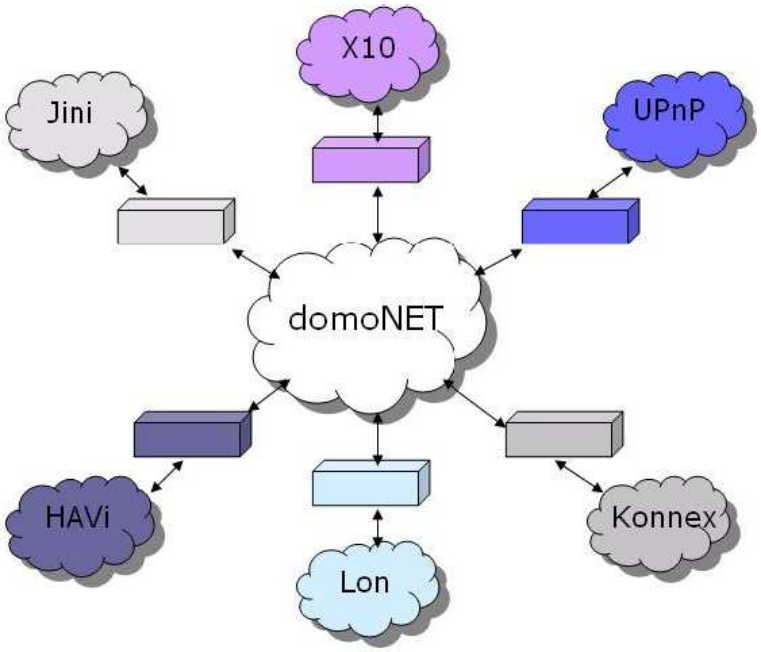

Figura 4: Il framework domoNET.

[8]Service Oriented Application

1.4.2 Un *techmanager* per l'utente

Una volta realizzato il framework domoNET, il Laboratorio di Domotica ha voluto realizzare un applicativo che fornisse un interfaccia per l'utente finale di accesso al framework stesso.

E' infatti logico pensare di rendere fruibile da chiunque il sistema domotico realizzato, il cui successo sarà realmente decretato solo nel caso in cui possa entrare a far parte della nostra vita quotidiana.

Si è pensato quindi di sfruttare *in primis* i dispositivi mobili più diffusi e posseduti da chiunque, dal costo molto basso e con una potenza di calcolo adeguata: i telefoni cellulari.

Per non realizzare però un semplice client per i Web Services di domoNET, che avrebbe ristretto i protocolli di accesso sfruttabili al solo TCP/IP, è stato pensato di realizzare un'entità che si frapponesse tra i Web Services di domoNET e il mondo esterno.

Una sorta di techmanager aggiuntivo per rendere accessibili a chiunque i servizi messi a disposizione da domoNET e da tutti i middleware domotici da esso gestiti.

Ecco come domoNET Mobile Gateway e il client domoNET Mobile Controller si inseriscono nel contesto del framework domoNET (figura 5). Parleremo nel dettaglio di entrambe le applicazioni nella sezione 2.

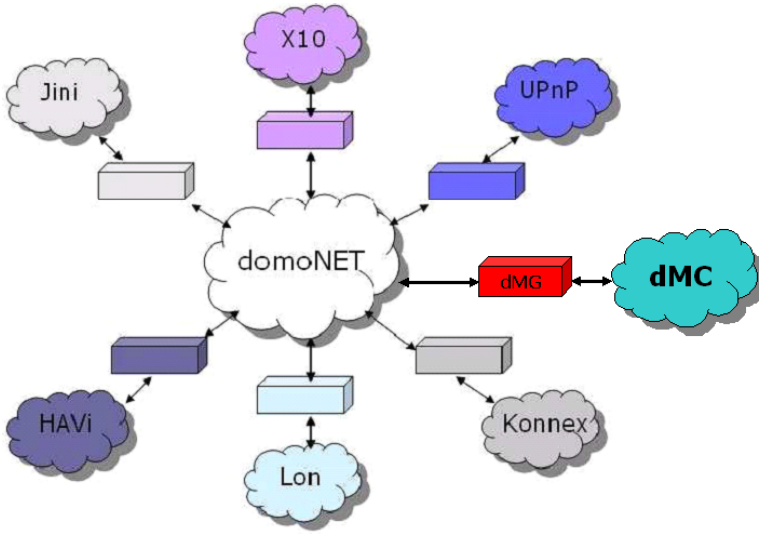

Figura 5: Il *techmanager* domoNET Mobile Gateway (dMG) e i client domoNET Mobile Controller (dMC).

2 Una soluzione 'wireless' per il controllo dell'ambiente domestico

Entrambe le applicazioni dMG[9] e dMC[10] sono state pensate per soddisfare i principi di generalità ed espandibilità che stanno alla base del linguaggio domoML e del software domoNET. Lo scopo stesso di queste due entità è infatti di creare un punto d'incontro tra sistemi domotici esistenti, spesso basati su tecnologie diametralmente opposte, ma anche di contemplare la possibilità di nascita di nuovi standard e di facilitare il lavoro di integrazione con gli standard esistenti; di conseguenza entrambe le applicazioni, dMG e dMC, lasciano la possibilità di integrare facilmente funzionalità per la comunicazione tramite media differenti, come ad esempio il già affermato *Bluetooth* o il nuovo *Zigbee*.

Considerando inoltre il carattere di generalità delle applicazioni, che da una parte presenta enormi vantaggi in uno scenario molto variegato come la domotica, dall'altra si riflette sulla intituivività, per esempio, dell'interfaccia utente di dMC, il codice stesso delle applicazioni è stato scritto prevedendo gli auspicabili potenziamenti del linguaggio domoML [11], progettato sempre dal Laboratorio di Domotica che ha lo specifico intento di porsi come lingua comune tra i vari standard, ed è fondamentale per astrarre le informazioni su dispositivi e servizi, circoscrivendo a poche righe di codice gli eventuali cambiamenti nella creazione delle interfacce utente.

domoNET Mobile Gateway e domoNET Mobile Controller (dMG e dMC),

[9] domoNET Mobile Gateway
[10] domoNET Mobile Controller
[11] cfr: 'Una grammatica XML per la Domotica' (Fabio Testa, Vittorio Miori)

sono state pensate e realizzate, così come domoNET stesso, con tecnologie assolutamente free e standard open-source.

Gli stessi sorgenti della Java Virtual Machine (JVM), piattaforma su cui sono state sviluppate tutte le applicazioni, sono stati da poco rilasciati ufficialmente dal detentore dei diritti, Sun Microsystems; questo evento, negli auspici di tutti, dovrebbe apportare notevoli miglioramenti, in particolare alle prestazioni del linguaggio Java e alle realizzazioni embedded della sua virtual machine, fin'ora vero punto a sfavore di questa tecnologia.

2.1 domoNET Mobile Gateway e domoNET Mobile Controller

2.1.1 Tecnologie adottate

Il linguaggio Java, adottato per lo sviluppo degli applicativi in questione, nasce nell'ambito di un grande progetto per la creazione di una vasta gamma di device di rete e sistemi embedded; proprio in questo campo sono attesi a breve termine gli sviluppi più interessanti, con l'arrivo dei primi chip con la Java Virtual Machine implementata in silicio e quando gli accordi della SUN con grossi produttori di sistemi embedded porteranno ai primi *PLC* programmabili in Java e le JavaCard, telefoniche o personali, diventeranno una realtà quotidiana.

La portabilità del codice è stata sicuramente la peculiarità che maggiormente ha contribuito alla scelta del linguaggio da adottare, vista anche la grande presenza sul mercato di dispositivi che hanno una virtual machine integrata.

Infatti qualsiasi software scritto in questo linguaggio di programmazione, può essere eseguito, una volta compilato, su qualsiasi piattaforma o dispositivo che disponga di una JVM.

L'indipendenza da piattaforme e sistemi operativi commerciali, quindi, è un altro punto fondamentale per mantenere assolutamente *free* l'utilizzo del software realizzato, come detto in precedenza.

Se pensiamo inoltre che nella realizzazione di sistemi domotici personalizzati, tutt'ora legati all'adozione di un unico standard per tutti i dispositivi, spesso richiede la conoscenza dei diversi linguaggi utilizzati, la scelta di un linguaggio come Java, e la peculiarità di domoNET di essere uno standard open-source, può ridurre notevolmente i costi di *training*, di gestione delle versioni e, in generale, del *time to market*, rivelandosi un fattore determinante per un auspicabile successo commerciale.

Inoltre, ed è spesso una regola, lo sviluppo del software avviene parallelamente a quello dei dispositivi fisici: questo richiede sforzi aggiuntivi per la creazione di simulatori e dei successivi adattamenti al sistema definitivo.

Un programma Java, invece, viene eseguito dalla propria JVM sia su un personal computer che su una scheda *custom*, con differenze limitate ai tempi d'esecuzione. Tutte queste doti costituiscono, di per sé, un motivo sufficiente per destare l'interesse di chiunque produca software professionale di qualsiasi tipo.

Quest'attenzione per la facilità che fornisce Java nello sviluppo e nella portabilità del codice su dispositivi integrati è fondamentalmente dettata dalle prospettive auspicate per lo standard domoNET il cui applicativo, come già avviene per alcuni standard domotici, potrà essere integrato direttamente

su un chip esclusivamente preposto alla gestione del sistema domotico casalingo, aumentando le prestazioni e la sicurezza e allo stesso tempo eliminando l'ingombro e i costi energetici di un calcolatore tradizionale, il pc casalingo per intenderci, sul quale domoNET è stato sviluppato e testato con successo, dimostrando la validità delle scelte effettuate a favore di generalità, portabilità ed economicità dello sviluppo.

2.1.2 Architettura dell'applicazione

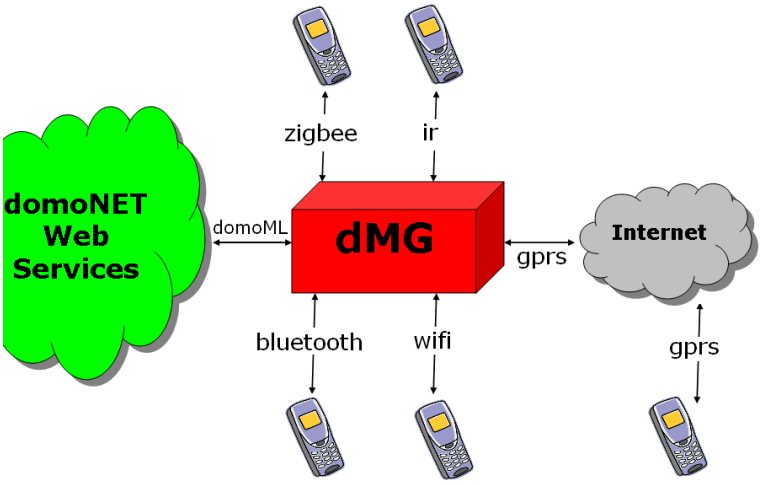

Figura 6: Architettura dello scenario domoNET con dMG e i client dMC

Osservando lo schema a nuvolette in figura 6 vediamo quindi 3 tipi di entità:

1. **domoNET Web Services**: è l'entità server, il cuore dell'architettura domoNET. Quest'applicazione è in grado di ricevere messaggi e richieste nel linguaggio *domoML*; i moduli collegati a questa entità, detti *techmanager*, sono fisicamente attaccati ai dispositivi della tecnologia che singolarmente devono gestire (Konnex, UPNP, Lonworks, X10

ecc.) e traducono gli input da essi provenienti nel linguaggio comune domoML per inviarli ai *Web Services*. I Web Services (domoNETWS) quindi, sono la destinazione di tutti i messaggi provenienti dalle diverse tecnologie e, grazie all'astrazione dei dispositivi fornita da domoML, che permette di conoscerne le caratteristiche e l'ubicazione, reindirizzano i messaggi al modulo della tecnologia destinataria del messaggio, che a sua volta ritradurrà il messaggio da domoML al linguaggio specifico del middleware domotico da lui stesso gestito.

2. **domoNET Mobile Gateway**: dMG è un'entità dalla duplice funzione; server, per i client di controllo esterni, client verso il core domoNET. Quest'applicazione è preposta a accettare richieste, secondo un protocollo prestabilito, da dispositivi che vogliono interagire con il server domoNET, traducendo quindi le richieste esterne in linguaggio domoML. dMG permette a domoNET di essere accessibile dal mondo esterno tramite un semplice protocollo basato su stringhe di caratteri; l'applicazione, che nella sua versione *alpha* rilasciata con questa tesi implementa un *techmanager* sul protocollo TCP/IP, è stata pensata per essere facilmente ampliata per gestire altri tipi di protocolli e *media*, come Bluetooth,Zigbee o Infrarossi (IR), sviluppando un package Java apposito come vedremo fra poco nel dettaglio, o reimplementando il protocollo di interazione nell'ambiente desiderato, non ponendo vincoli legati al runtime di un particolare linguaggio o allo sviluppo su una determinata piattaforma.

3. **domoNET Mobile Controller**: dMC è un esempio di una generica

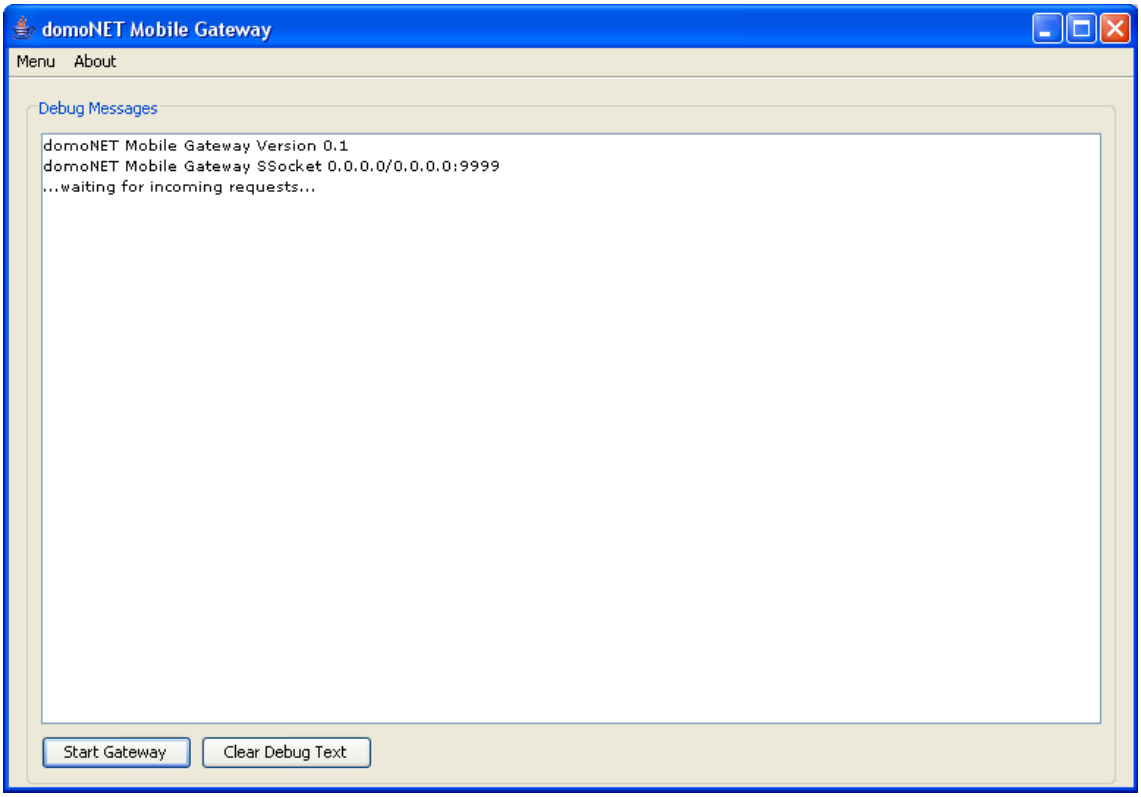

Figura 7: La schermata di log di dMG.

applicazione client, che interagisce con domoNET tramite dMG. è stata sviluppata su piattaforma J2ME (Java 2 Micro Edition[12]) la cui Virtual Machine è implementata nella maggior parte dei dispositivi mobili presenti attualmente sul mercato. In particolare è stata posta attenzione ai telefoni cellulari, dispositivi ormai posseduti da chiunque e sui quali la JVM è implementata in maniera standard da quasi tutti i costruttori. Utilizzando quindi il protocollo TCP/IP, e testando l'applicazione su un telefono cellulare dotato di scheda di rete WIFI (Nokia N80[13]), è stato implementato il semplice protocollo a domanda e risposta necessario per interagire con dMG. L'interfaccia grafica dell'applicazione, rilasciata sempre in versione alpha, è comunque stata sviluppata nel modo più generale possibile, cosicché dMC può esser già installato su una vastissima gamma di dispositivi, anche quelli considerati generalmente come obsoleti. E' stato infatti anche testata l'interazione attraverso Internet (nuvoletta grigia in figura 6), installando il client dMC su un telefono con un semplice collegamento *GPRS*.

2.1.3 domoNET Mobile Gateway

dMG è un'applicazione *multi-threaded* che mette a disposizione i metodi della classe **DomoNETClient**, sviluppata dal Laboratorio di Domotica, che implementa le comunicazioni con il server domoNET; inoltre permette l'utilizzo delle classi di libreria Java che permettono di manipolare il linguaggio domoML.

[12] http://java.sun.com/javame/index.jsp
[13] http://www.nokia.it

Figura 8: domoNET Mobile Controller in attesa di connettersi a domoNET Mobile Gateway.

E' stato necessario, oltre che per una scelta votata all'efficienza e alla scalabilità, frapporre un'applicazione tra i Web Services e il client J2ME. La *Java Virtual Machine* disponibile su tutti i telefoni cellulari esistenti, per uno specifico accordo tra Sun Microsystems[14] e i maggiori produttori di dispositivi mobili, non viene sviluppata e aggiornata fin dalla versione 1.3 del linguaggio Java.

Sarebbe stato quindi impossibile, per l'introduzione di alcuni nuovi meccanismi nel linguaggio stesso, tra cui i *Java Generics*, sfruttare le librerie per domoML già sviluppate dal Laboratorio di Domotica sulla versione *J2SDK 1.5*. Inoltre la quantità di informazioni fornite da domoNET Web Services ad ogni richiesta di dMG è stata giudicata eccessiva, sia per un'interfaccia dalle possibilità limitate come quella di un dispositivo mobile, sia per la scarsa potenza di calcolo del dispositivo stesso, sia per il costo in termini di latenza di trasmissione dei media utilizzati.

Per quel che riguarda l'obiettivo della scalabilità, domoNET Mobile Gateway sicuramente offre molte più opportunità di espandibilità di un client con accesso diretto a domoNET; l'applicativo è stato infatti strutturato in modo tale da rendere possibile un facile sviluppo di un server per il media desiderato, aggiungengo poche righe di codice nel metodo **main()** per integrare il nuovo *package* sviluppato.

Nel caso specifico per integrare un nuovo package basta aggiungere:

1. import miopackage.MioServer;

2. MioServer nome = new Mioserver();

[14] http://www.sun.com

3. nome.start();

Affinchè un nuovo *thread-server* sia un package valido per essere integrato in dMG, è necessario che implementi l'interfaccia `domoNETGatewayServer`, secondo le specifiche del linguaggio Java;

- public class MioServer extends Thread implements domoNETGatewayServer

Il codice dell'interfaccia Java `domoNETGatewayServer`, che descrive inoltre il protocollo di comunicazione ed è ampiamente commentato per permettere la corretta implementazione dei metodi, è riportato nell'appendice **??**.

Nel caso del thread server TCP/IP implementato contestualmente a dMG, il package *tcpserver*, la sua struttura è composta da:

- SSocket.java:

  questa classe, che estende la classe standard *Thread*, crea un *ServerSocket* in attesa di connessioni esterne; al momento in cui perviene la connessione alla porta 9999, viene creato un nuovo thread della classe *Handler* a cui è delegata la totale gestione del client appena collegatosi. Questo modello permette al server TCP/IP implementato di poter gestire potenzialmente un numero illimitato di client esterni, lanciando per ognuno un nuovo *handler* autonomo (figura 9).

  N.B: nella versione *alpha* di dMG, così come per domoNET, non è stata prevista nessuna forma di sicurezza per l'accesso al sistema (autenticazione del client via login o certificato) perchè presenta solo un aspetto marginale delle applicazioni, mentre si è preferito concentrarsi sulle funzionalità.

- Handler.java:

  questa classe, che è il cuore del thread-server TCP/IP, implementa l'interfaccia domoNETGatewayServer, e quindi il protocollo di comunicazione con il client esterno, facendo da tramite per esso verso domoNET, ed estende la classe *Thread*. La completa gestione di un client esterno è quindi delegata totalmente a questa classe, che attende i messaggi di richiesta ed agisce di conseguenza, sfruttando i metodi della classe *DomoNetClient* per comunicare con i Web Services di domoNET.

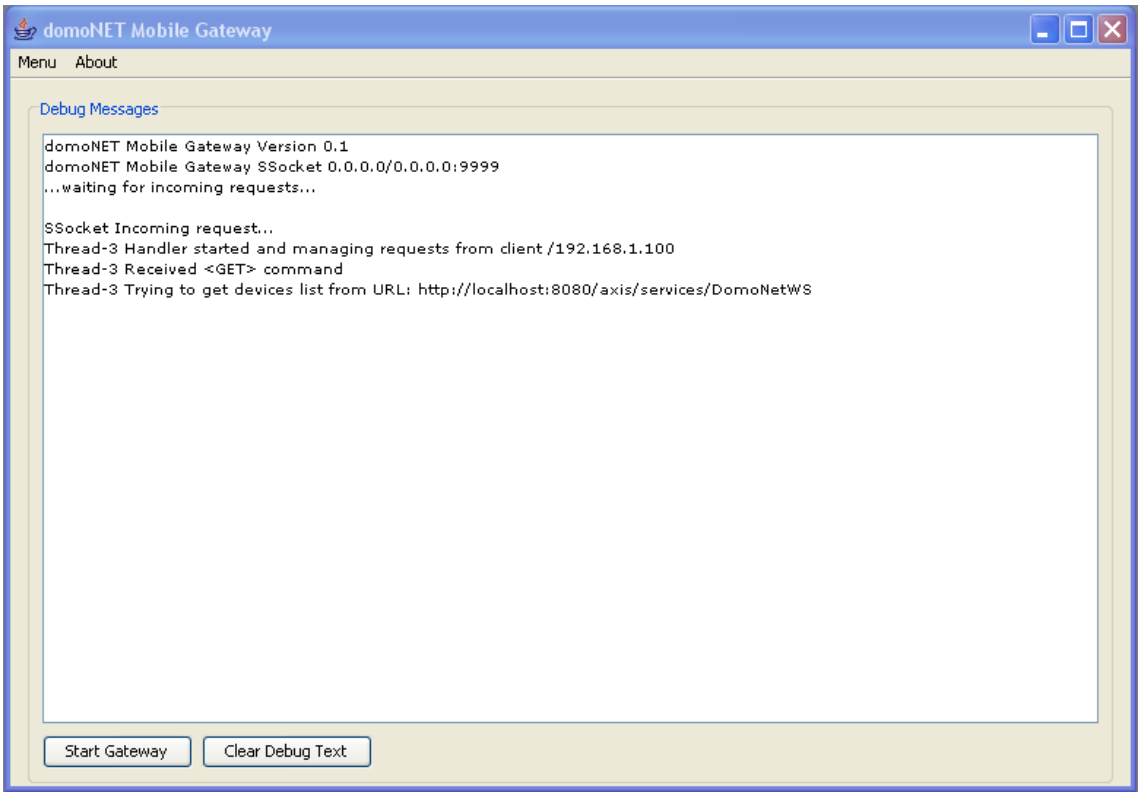

Figura 9: domoNET Mobile Gateway ha ricevuto una richiesta di connessione da un nuovo client TCP/IP e lancia per esso un nuovo *handler* corrispondente.

2.1.4 domoNET Mobile Controller

dMC è un'applicazione standard J2ME, sicuramente priva di un'interfaccia grafica accattivante, ma priva anche di particolari requisiti di piattaforma o di sistema.

I dispositivi su cui può essere installata, eseguita ed utilizzata con successo sono tutti quelli dotati di una JVM[15] standard per dispositivi mobili. Questo ha permesso uno sviluppo rapido da un punto di vista di studio dell'aspetto ed usabilità dell'interfaccia grafica, lasciando però più tempo da dedicare alla sua generalità e alla sua struttura scalabile.

Vediamo ora quali classi compongono il client dMC:

1. domoBILE:

   estende in modo standard la classe *Midlet* e fornisce il campo di testo in cui inserire l'URL del domoNET Mobile Gateway a cui ci si vuole connettere. Nel caso in cui si siano sviluppati clients per altri *media*, il pulsante *connect* diventa un menù a comparsa per la scelta del modo di connessione da utilizzare. Una volta scelto il media tramite cui connettersi avvia il thread corrispondente, nel caso specifico dell'applicazione sviluppata, il thread *TcpClient*.

   N.B.: il client TCP/IP sfrutta una qualsiasi interfaccia di rete sia presente sul dispositivo; nel caso in cui sia presente una scheda WIFI, la connessione avverrà tramite la LAN[16], al contrario la connessione avverrà tramite il servizio GPRS del

[15] Java Virtual Machine
[16] Local Area Network

proprio gestore telefonico, in questo modo dando la possibilità di collegarsi a proprio dMG casalingo anche da remoto.

2. TcpClient:

   è il cuore dell'applicazione dMC. Un thread che fornisce una connessione su protocollo TCP/IP a dMG e implementa il protocollo di comunicazione con esso. Una volta avviato chiede subito al gateway la lista dei dispositivo domotici presenti e fornisce la medesima lista all'utente (figure 10 e 11). Ogni dispositivo è definitivo in un oggetto inizializzato ad hoc nella classe *DomoMobileDevice*; il menù a comparsa permette di vedere i dettagli di ogni dispositivo (gli attributi dell'oggetto corrispondente) [figure 12, 13 e 14]; Rimarchevole per la perseguita generalità, è la capacità di costruire in modo dinamico l'interfaccia per la schermata dei servizi di un dispositivo. Infatti il metodo `buildServiceScreen(DomoMobileDevice device,Vector domoservices)` non fa nessun assunto sul numero di servizi e sul tipo di input di ogni servizio. Questa caratteristica certo va a scapito dell'intuitività dell'interfaccia utente, ma visti i non enumerabili possibili tipi di servizi, abbiamo ritenuto fondamentale anteporre la funzionalità e generalità dell'applicativo, alla sua intuitività ed usabilità. Facendo un rapido esempio, un servizio che ha come input un DataType *BOOLEAN*, potrebbe essere nel 90% dei casi un servizio di accensione/spegnimento; facendo un assunto del genere, e quindi costruendo per il servizio in questione un pannello con un bottone *ON/OFF* piuttosto che con un campo di testo in cui inserire 1 o 0, l'interfaccia risul-

Figura 10: domoNET Mobile Controller attende risposta alla lista di dispositivi richiesta.

Figura 11: domoNET Mobile Controller visualizza la lista di dispositivi presenti sui Web Services di domoNET.

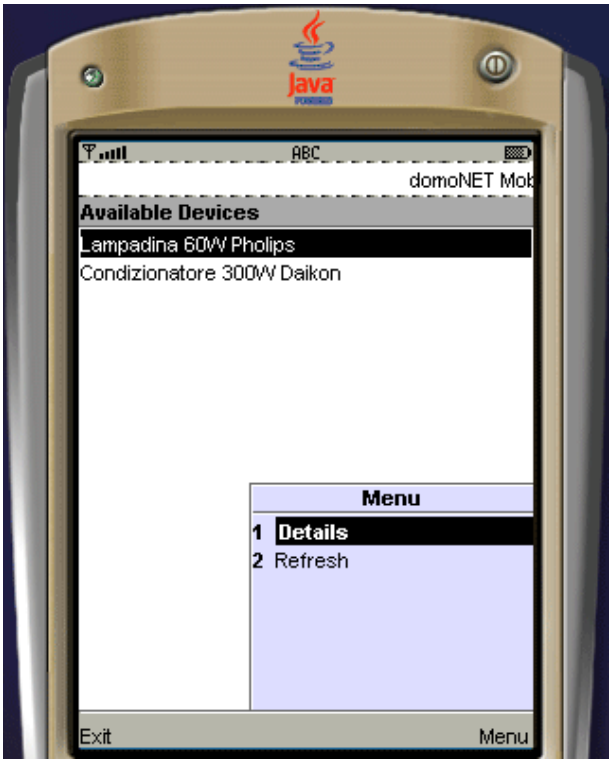

Figura 12: domoNET Mobile Controller e il menù a comparsa.

terebbe sicuramente più *user-friendly*. Nel caso in cui però il servizio

Figura 13: domoNET Mobile Controller visualizza i dettagli di un dispositivo domotico (lampadina).

non sia di tipo acceso/spento, ma fosse ad esempio un *flag* su un indice di temperatura, potrebbe presentarsi il caso in cui il nome servizio non sia significativamente associabile al pulsante, creando ancor più confusione di un campo di testo.

Questa problematica è in ogni modo legata a ciò a cui si faceva riferimento nel capitolo **??** sul linguaggio domoML; se ci fosse un unico nome di servizio, per la funzione *POWER* per esempio, anziché infinite possibilità come SetPower,Power,Set_Power ecc. . . , sarebbe possibile

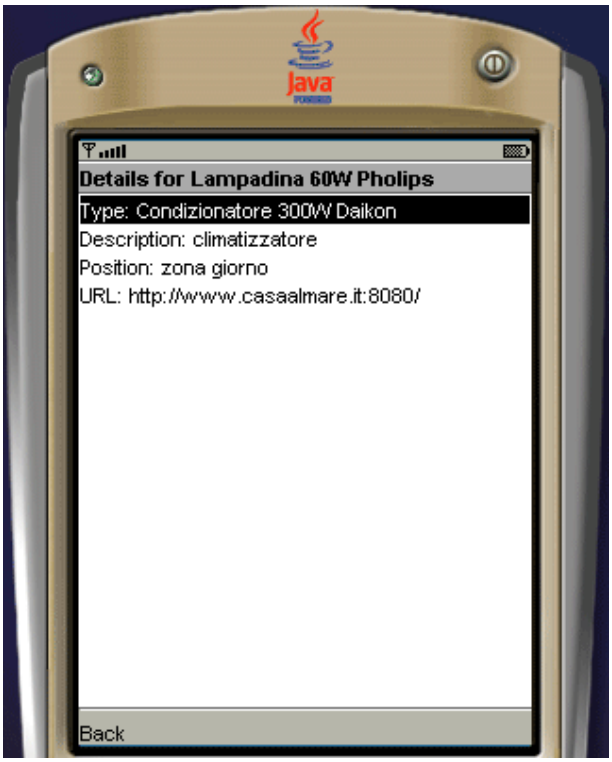

Figura 14: domoNET Mobile Controller visualizza i dettagli di un dispositivo domotico (condizionatore d'aria).

costruire l'interfaccia con un enumerazione finita di possibili stringhe identificanti i diversi servizi, cosa attualmente impossibile;

3. Sender:

   un thread che permette l'invio di una stringa sull'*OutputStream* verso dMG passato come parametro. La presenza di questo thread è fondamentale per non effettuare una *trasmissione sincrona*, e quindi bloccante o con latenza di comunicazione lunga nella maggior parte dei casi, con dMG;

4. package domoBILE. Questo package java è stato incluso per fornire la definizione in formato *"mobile"*, quindi più compatto, degli oggetti DomoDevice, DomoDeviceService e DomoDeviceServiceInput presenti nel *package DomoNetClient*, sviluppato dal Laboratorio di Domotica. E' infatti composto da:

   - DomoMobileDevice

     una classe che descrive un dispositivo domotico presente su domoNET in formato compatto. I suoi attributi sono:

     - URL: l'URL dei Web Services di domoNET a cui appartiene il dispositivo;
     - id: un numero intero associato al dispositivo. Insieme all'URL identifica univocamente un dispositivo
     - type: l'ambito in cui il dispositivo si colloca nell'ambiente domestico (illuminazione, sicurezza, riscaldamento, ecc...);
     - description: una descrizione del dispositivo;

  - position: la stanza o l'ambiente della casa o dell'edificio in cui è collocato il dispositivo;

- DomoMobileService

  una classe che descrive un servizio associato ad un DomoMobileDevice. I suoi attributi sono:

  - URL: l'URL dei Web Services di domoNET a cui appartiene il dispositivo a cui appartiene il servizio;
  - id: un numero intero associato al dispositivo. Insieme all'URL identifica univocamente il dispositivo a cui questo servizio appartiene;
  - name: il nome del servizio.

  Il costruttore di un oggetto DomoMobileService è duplice. Infatti il servizio può essere un servizio di tipo `OUTPUT` o di tipo `INPUT`. Nel primo caso il costruttore presenta questi ulteriori campi:

  - output: il nome dell'output fornito da questo servizio;
  - value: il valore associato all'output di questo servizio.

  Nel secondo caso il costruttore presenta questo unico ulteriore campo:

  - inputs: un vettore contenente oggetti di tipo `DomoMobileServiceInput`.

- DomoMobileServiceInput

  una classe che descrive gli input di un DomoMobileService associato ad un DomoMobileDevice. I suoi attributi sono:

- URL: l'URL dei Web Services di domoNET a cui appartiene il dispositivo a cui appartiene il servizio;
- id: un numero intero associato al dispositivo. Insieme all'URL identifica univocamente il dispositivo a cui questo servizio appartiene;
- service: il nome del servizio;
- name: il nome del servizio di input;
- description: una descrizione del servizio di input;
- datatype: il tipo di input del servizio (BOOLEAN, INT, STRING, ecc...).

2.1.5 Protocollo di interazione tra gli applicativi coinvolti

Il protocollo di interazione è stato studiato appositamente per rendere non solo più rapide le comunicazioni, ma anche possibile una sua reimplementazione su un qualsiasi linguaggio diverso da Java.

Infatti sarebbe stato possibile utilizzare il meccanismo di *serializzazione*, messo a disposizione dal linguaggio Java, per inviare sullo *stream* direttamente gli oggetti `DomoDevice`, rendendo il codice molto più semplice, sia da scrivere che da leggere, e il protocollo molto più snello.

Al contrario è stata posta attenzione alla possibilità di sviluppare un client per dMG su una piattaforma e con un linguaggio completamente diversi da quelli utilizzati negli attuali applicativi. Il solo modo per rendere possibile uno scenario futuro di questo tipo è di fatto usare un protocollo a scambio di stringhe.

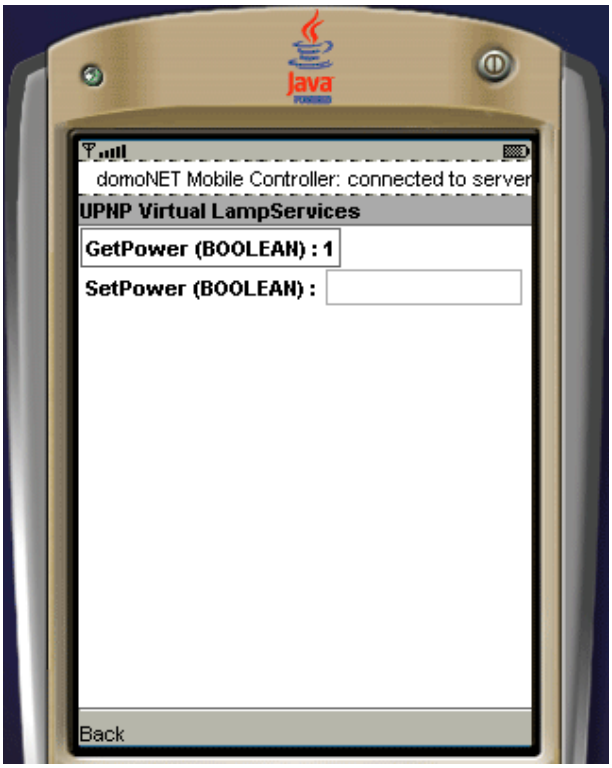

Figura 15: domoNET Mobile Controller visualizza i servizi servizi di un dispositivo domotico (lampadina).

Figura 16: domoNET Mobile Controller visualizza i servizi su di dispositivo domotico (condizionatore d'aria).

La successione di stringhe scambiate tra dMC e dMG, è facilmente reimplementabile con qualsiasi linguaggio possa gestire uno stream di caratteri.

Vediamo ora come concettualmente si verificano le interazioni tra domoNET Mobile Controller, domoNET Mobile Gateway e domoNET Web Services (figura 17). Vediamo che dMG rimane in attesa di richieste da un client es-

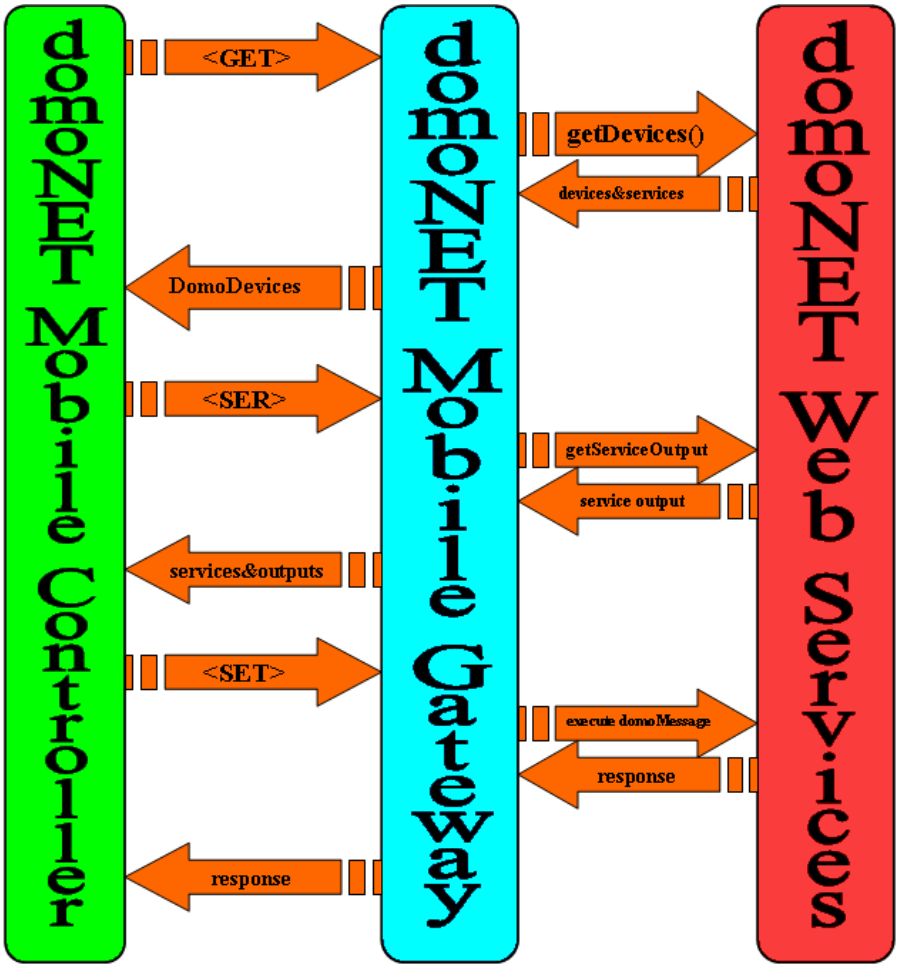

Figura 17: Un esempio delle comunicazioni tra le 3 entità coivolte: dMC(verde chiaro), dMG(azzurro), domoNET(rosa).

terno, nella fattispecie dMC, e una volta "parsata" la richiesta intraprende

le azioni necessarie a soddisfarla. I possibili comandi sono:

1. `GET`:

   è il comando che richiede la lista di dispositivi presenti su domoNET Web Services. In realtà dMG, sfruttando la logica del package *DomoNetClient*, inizializza una struttura dati, detta *HashMap*, in cui mantiene oltre ai dispositivi, anche i servizi ad essi associati per usi futuri (figura 18);

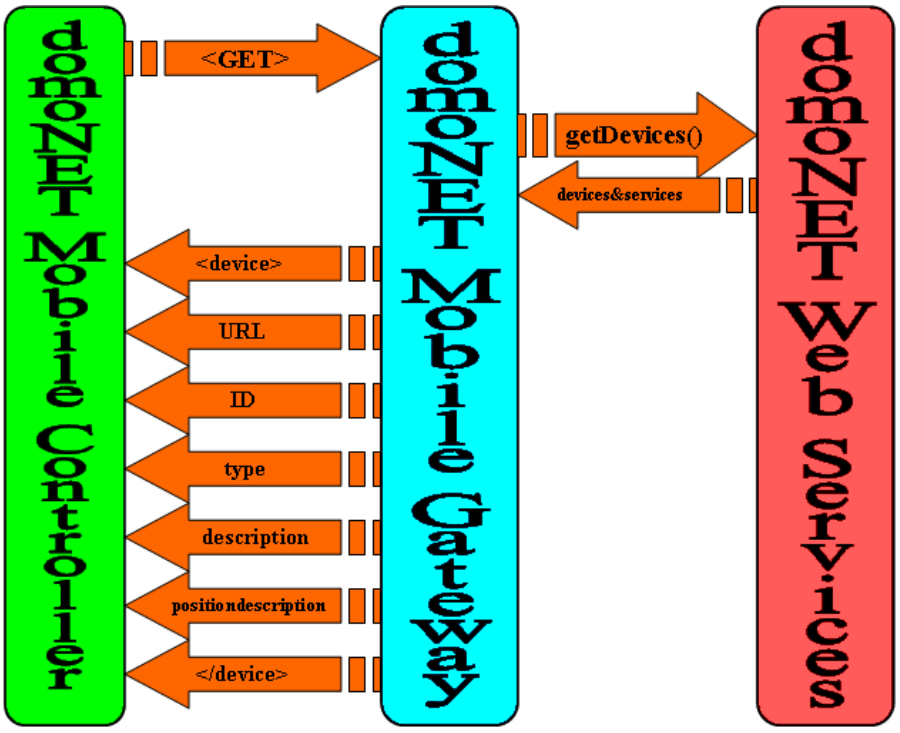

Figura 18: Il caso particolare della GET.

2. `SER`:

   è il comando che richiede la lista dei servizi associati ad un certo dispositivo. dMG non solo restituisce al client la lista dei dispositivi, ma an-

ticipa la conseguente richiesta successiva, richiedendo contestualmente a domoNET i valori di *output* dei servizi di questo tipo, allegandoli alla lista dei servizi (figura 19);

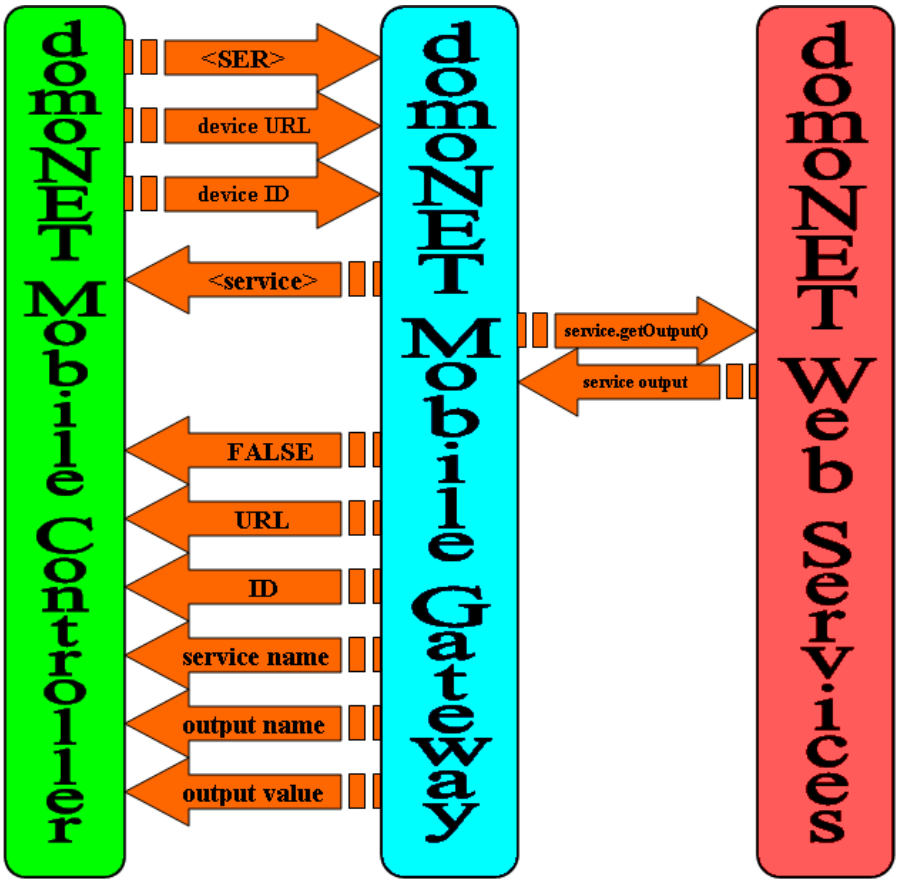

Figura 19: Il caso particolare della SER.

3. `SET`:

   è il comando per eseguire una data azione su un dispositivo tramite il servizio specificato nella richiesta. dMG attende sullo stream tutte le informazioni necessarie per creare un DomoMessage dopodiché lo

esegue, attende il responso da domoNET e infine lo inoltra al client (figura 20);

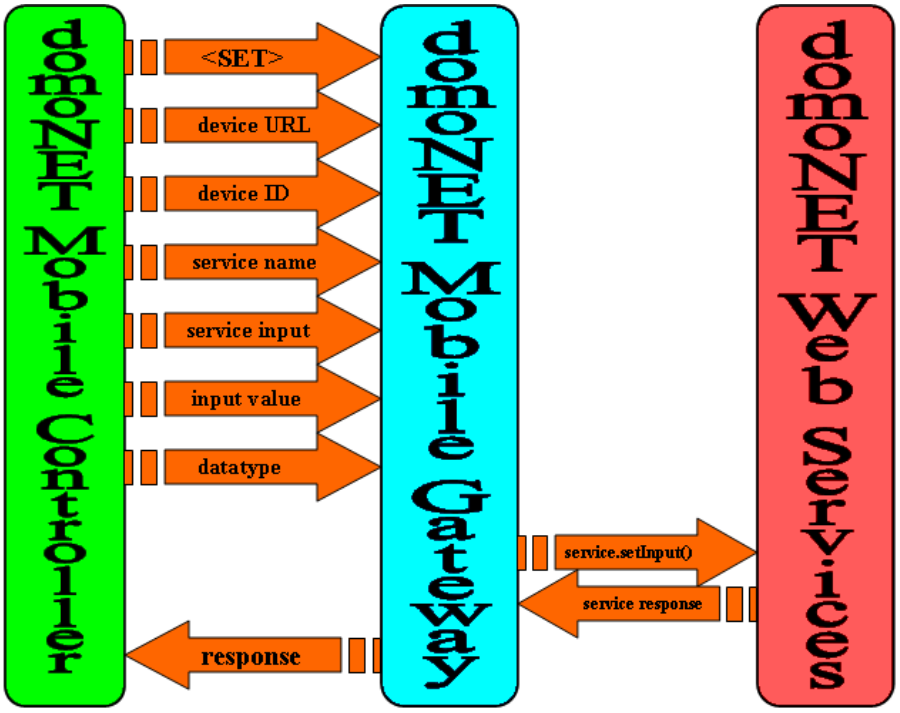

Figura 20: Il caso particolare della SET.

4. `END`:

   è il comando di chiusura della connessione da parte del client. A questo punto il thread *handler* lanciato appositamente per gestire le richieste provenienti da questo client, termina.

3 Conclusioni

L'applicativo realizzato è quindi una componente importante del progetto domoNET, un lavoro di ampio respiro e prospettive interessanti che già dalle premesse non vuole rimanere solo uno studio "accademico", ma fungere da spinta in avanti per un ambito dell'informatica come la domotica, ancora relegato ad una piccola nicchia non solo del mercato, ma anche dalla nostra vita quotidiana.

Da diversi anni ormai, vediamo stabilire un rapporto più stretto tra informatica e azioni quotidiane; in questa sede non disquisirò certo degli aspetti positivi o negativi di questa "tecnologizzazione", certo è che la domotica davvero potrebbe apportare un sensibile miglioramento alla nostra vita quotidiana, a partire dalla gestione "intelligente" dei consumi energetici delle nostre case che, a fronte di una spesa leggermente superiore per l'acquisto dei dispositivi domotici, ridurrebbe i costi delle nostre bollette, nonché darebbe un contributo piccolo, ma significativo se apportato da milioni di utenti, alla riduzione dell'inquinamento.

domoML, domoNET, domoNET Mobile Gateway e domoNET Mobile Controller sono solo i primi passi verso l'auspicabile interoperabilità globale tra tutti gli standard domotici esistenti.

Basti pensare alla diffusione delle reti di calcolatori fino alla fine degli anni '70; prima solo poche e grosse aziende potevano permettersi dei calcolatori che "parlassero" tra di loro, peraltro vincolandosi all'acquisto di altre macchine dello stesso produttore nel caso volessero ampliare la propria rete. Questa è al giorno d'oggi la situazione della domotica.

Proseguendo su questo parallelo, la grande ambizione di domoNET è di essere per la domotica ciò che fu lo standard *Ethernet* per le reti di calcolatori. Oggi, dopo 30 anni dalla sua affermazione, riesce difficile, se non impossibile, immaginare che 2 calcolatori non possano comunicare tra di loro; infatti, anche nel caso in cui un calcolatore non abbia già integrato un adattatore Ethernet, basta spendere pochi euro per acquistarne uno e permettere al proprio calcolatore di "affacciarsi sul mondo".

Per quel che riguarda la domotica certamente i produttori di dispositivi hanno già fatto molti sforzi per convergere su un unico standard (vedi Konnex), ma quei produttori che hanno sviluppato un loro standard, con alti costi d'investimento, e tutt'ora producono dispositivi conformi ad esso, difficilmente rinunceranno ad imporre il proprio prodotto sul mercato.

Ecco perché un ente *super partes* come il CNR e un progetto con le caratteristiche di domoNET, hanno tutte le carte in regola per riuscire quantomeno a dare una "scossa all'ambiente".

Riferimenti bibliografici

[1] Cay Horstmann, *Fondamenti di Java 2,* (Second Edition 2000 McGraw Hill).

[2] Elliotte Rusty Harold, *Java Network Programming,* (Second Edition 2000 O'Reilly).

[3] Cay Horstmann, Gary Cornell, *Java 2, Tecniche avanzate,* (Second Edition 2000 McGraw Hill).

[4] J. F. Kurose , Keith W. Ross, *Computer Networking,* (Second Edition 2003 Addison Wesley).

[5] Micheal J. Young, *XML Passo per Passo,* (Second Edition 2001 Microsoft Press).

[6] Steven Holzer, *XML Tutto e Oltre,* (Apogeo 2001).

[7] Bianchi Bandinelli R., *Lucidi del corso di Domotica (CLS Informatica, a.a. 2005/06)* http://www.di.unipi.it/~bandinel/domotica.html

[8] Konnex Association, *http://www.konnex.org*

[9] Konnex Standard System Specification: *Architecture Vol.3 Part I (Konnex Association, 2001)* (non di dominio pubblico)

[10] UPnP Forum, *http://www.upnp.org*

[11] UPnP Device Architecture 1.0 (UPnP Forum, 2003) *http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf*

[12] Mahmoud Q.H., *Registration and Discovery of Web Services Using JAXR* (2002)

[13] Topley K., *Java Web Services in a nutshell* (O'Reilly,2003)

[14] *Java Web Services Tutorial* (Sun Microsystems, 2004)

[15] McLaughlin B., *Java and XML 2nd Edition: Solutions to Real-World Problems* (O'Reilly, 2001)

[16] Harold E.R., *XML Bible 2nd Edition* (John Wiley and Sons, 2001)

[17] Hyde P., *Java Thread Programming: The Authoritative Solution* (SAMS, 1999)

[18] Horstmann C.S., Cornell G., *Core Java 2 Volume II:Advanced Features* (Prentice Hall, 1999)

[19] *http://www.netbeans.org/kb/articles/form getstart40.html*

[20] *http://java.sun.com/developer/technicalArticles/WebServices/jaxrws/*

[21] *http://java.sun.com/webservices/tutorial.html*

[22] *http://java.sun.com/docs/books/tutorial/native1.1*

[23] Nichols, B.A. Myers, M. Higgins, J. Hughes, T.K. Harris, R. Rosenfeld and M. Pignol, *Generating Remote Control Interfaces for Complex Appliances*, Proceedings of the 15th annual ACM symposium on user interface software and technology, ACM Press, October 2002.

[24] F. Furfari, C. Soria, V. Pirrelli, O. Signore, R. Bandinelli, *NICHE: Natural Interaction in Computerized Home Environment*, Ercim News no. 58, July 2004.

[25] L. Tarrini, V. Miori, *LIGHT: XML-Innovative Generation for Home Networking Technologies*, Ercim News no. 62, July 2005.

[26] V. Miori, L. Tarrini, M.Manca, and G. Tolomei, *An Open Standard Solution for Domotic Interoperability*, IEEE TRANSACTIONS on Consumer Electronics, Vol.52, num. 1, February 2006

[27] V. Miori, L. Tarrini, M.Manca, and G. Tolomei, *An innovative, open-standards solution for Konnex interoperability with other domotic middlewares,* Konnex scientific conference 2005, Pisa, Italy. Atti pubblicati su CD-Rom.

[28] Vittorio Miori, *La casa intelligente-Ricerca scientifica e futuri scenari*, ScienzaOnline, numero 21, anno 2

[29] V. Miori, L. Tarrini, M.Manca, and G. Tolomei, *DomoNet: A framework and a prototype for interoperability of domotics middlewares based on XML and WebServices*. 2006 IEEE International Conference on Consumer Electronics (ICCE) (Las Vegas, USA, 7-11 gennaio 2006).

[30] Dario Russo, *La domotica e Internet. Una soluzione per l'interoperabilità*. Tesi di Laurea in Informatica. 2006 Università di Pisa.

Elenco delle figure