L'integrazione della tecnologia BACnet nel sistema per l'interoperabilità delle reti domotiche

Salvatore Mazzurco, Vittorio Miori

Indice

Capitolo 1

DomoNet

1 Il framework DomoNet

Il framework DomoNet è una soluzione di alto livello a uno dei problemi classici della Domotica, vale a dire il **problema dell'interoperabilità** tra tecnologie distinte.
Questo problema è affrontato anche e soprattutto dalle tecnologie domotiche emergenti che, utilizzando la strategia della loro interoperabilità con gli standard più affermati, mirano ad imporsi sul mercato.
Spesso invece la risposta tecnologica a tale problema degli standard affermati prevede la presenza di **gateway** (solitamente Hardware) capaci di tradurre da uno standard ad un altro.
DomoNet riveste una posizione in antitesi alle soluzioni "gateway" in quanto:

- le traduzioni 1:1 effettuate dai gateway non scalano all'aumentare degli standard da tradurre;
- questo tipo di soluzioni prevedono comunque la presenza nella rete di ulteriori dispositivi potenzialmente non necessari. Questo va a impattare sia sui costi, sia dal punto di vista dello spreco di indirizzi;
- la scelta di Gateway "pronti all'uso" prevede comunque una fase di studio per conoscerne il grado di trasparenza e capire se e quanto si perde nella traduzione/interpretazione dei protocolli.

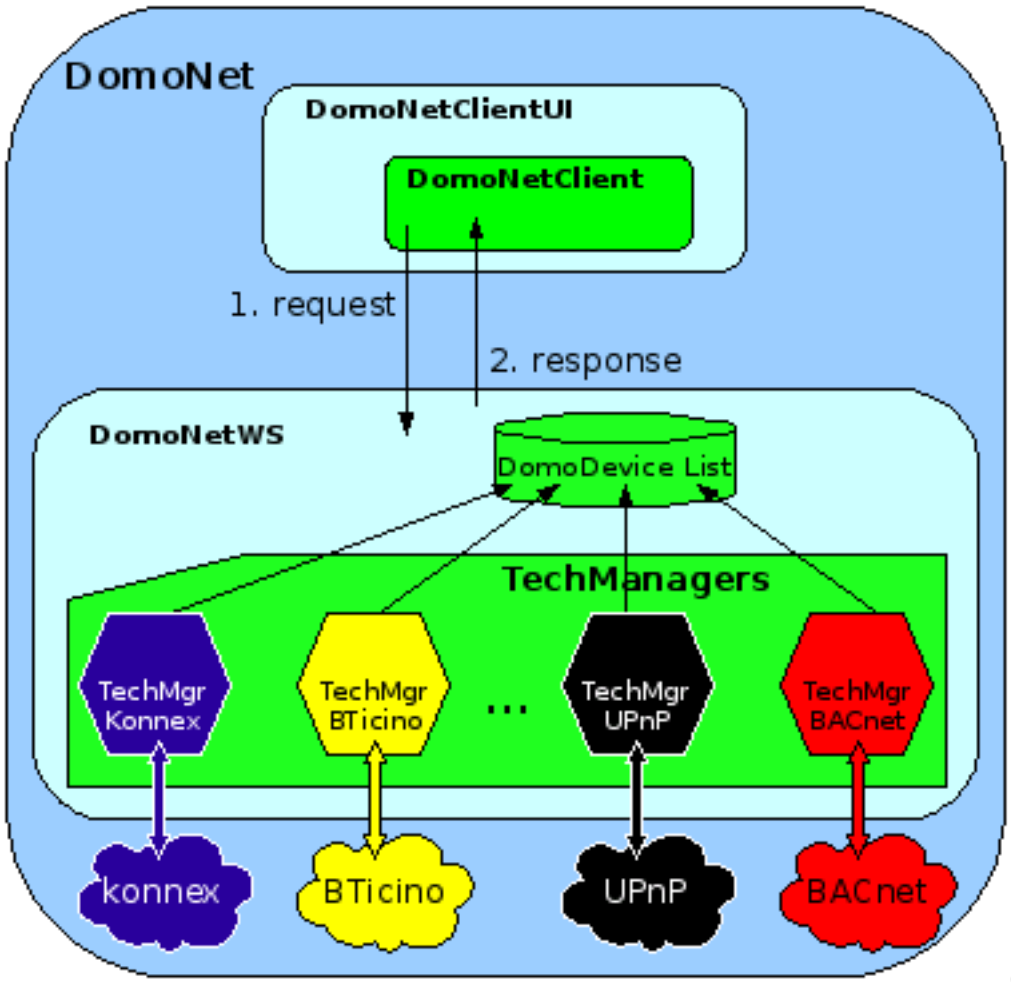

Figura 1.1: L'infrastruttura DomoNet.

L'approccio usato da DomoNet prevede la definizione di un'astrazione generale indipendente dai *middleware*[1] sottostanti capace di offrire una serie di servizi per:

- identificare i dispositivi presenti nei vari *middleware*;
- conoscere lo stato e le funzionalità offerte da ciascun dispositivo presente nei vari *middleware*;

[1] Con il termine *middleware* si intende una sottorete all'interno della quale sono presenti sia le infrastrutture hardware e software per il supporto, sia i dispositivi conformi (pag 60 di [2]).

- modificare, quando possibile[2], lo stato di ciascun dispositivo presente nei vari *middleware*;
- associare le funzionalità dei dispositivi anche se quest'ultimi appartengono a *middleware* distinti.

In particolare l'ultimo punto è quello che realizza l'interoperabilità. Nella soluzione DomoNet il tipo di *middleware* non rappresenterà però alcun vincolo.
La figura 1.1 mette in evidenza l'infrastruttura di DomoNet.
Le entità principali sono il **DomoNetClient** e il **DomoNetWS** che instaurano la classica comunicazione cliente servente.
Contrariamente al *DomoNetClient* che può essere eseguito ovunque, il *DomoNetWS* va installato e opportunamente configurato nell'ambiente in cui sono presenti i *middleware* domotici, i quali dovranno essere collegati fisicamente alla macchina su cui viene eseguito *DomoNetWS* affinché possa avvenire la comunicazione con gli opportuni *TechManager*.
Il *DomoNetWS* è costituito da una lista di TechManager (rappresentati con degli esagoni nella fig. 1.1). I TechManager sono la parte modulare di *DomoNetWS* e la figura 1.1 mostra solamente qualche esempio.
Con le "nuvolette" vengono invece rappresentati i *middleware* comunicanti con i TechManager di DomoNet.
Il *DomoNetWS* mantiene inoltre una struttura dati contenente la **lista di DomoDevice**. Questi ultimi sono la rappresentazione in **DomoML** di un qualunque dispositivo appartenente a uno qualunque dei *middleware* presenti. Nei paragrafi successivi verranno descritti con maggior dettaglio gli elementi di DomoNet.

1.1 DomoML

Il livello astratto di DomoNet è implementato attraverso un linguaggio, basato su XML, chiamato **domoML**. Con questo linguaggio è possibile descrivere tutti i dispositivi domotici (domoDevice) e i relativi messaggi (domoMessage).

[2]Tale possibilità dipende esclusivamente dallo Standard e dalla tipologia di dispositivo, questo significa che DomoNet è totalmente trasparente!

Attraverso *DomoML* vengono descritti i servizi e le interazioni di un dispositivo domotico, astraendo dalla tecnologia di appartenenza.
DomoML in questa architettura riveste il ruolo di linguaggio intermedio da e verso il quale tradurre le rappresentazioni dei dispositivi e dei pacchetti per cui tutta la parte logica dell'architettura usa *domoML* lasciando ai *tech manager* il compito di tradurre in interazioni fisiche i messaggi descritti in sintassi *DomoML*.

1.1.1 DomoDevice

Il costrutto *DomoDevice* ha lo scopo di creare un meccanismo generale per rappresentare i dispositivi astraendo dalla tecnologie.
Il *DomoDevice* è strutturato ad albero e contiene informazioni come:

- la **URL** del web Service usata per rintracciare l'area del dispositivo (Questa informazione viene usata per recuperare il web Service di appartenenza in una rete costituita da diversi DomoNetWS cooperanti);
- l' **ID** che identifica il dispositivo all'interno del DomoNetWS. Attraverso questo identificatore si possono ricavare dalla struttura dati "DomoDeviceList" tutte le altre informazioni del dispositivo;
- il **Serial Number** vale a dire un identificatore che permette il riconoscimento del dispositivo all'interno del *middleware*;
- una lista di **servizi** offerti dal dispositivo che rappresentano le funzionalità che il dispositivo è in grado di realizzare. A sua volta un servizio è modellato in generale[3] con **parametri d'ingresso** e **parametri d'uscita** entrambi tipati;
- all'interno di ogni servizio si possono definire i **LinkedService** per associare l'esecuzione del servizio ad un altro servizio.

Per la visione di un esempio e per maggiori dettagli vedere il par. 5.1 (pag 66-71) di [2].

[3]Questo significa che ogni servizio in relazione alla natura della funzionalità mappata può avere ingressi e uscite ma può anche non averne se la funzionalità da mappare non li prevede.

1.1.2 DomoMessage

Il costrutto *DomoMessage* ha lo scopo di creare un meccanismo generale per descrivere le interazioni tra i *DomoDevice* astraendo dalle tecnologie.
Il *DomoMessage* è strutturato ad albero e contiene tutti i dettagli necessari affinché qualunque *TechManager* possa interpretarlo e tradurlo in un messaggio fisico da inviare nella propria rete domotica[4].
Per la visione di un esempio e per maggiori dettagli andare al par. 5.2 (pag 71-73) di [2].

1.1.3 I LinkedService

I *LinkedService* sono il cuore di DomoNet. Sono il costrutto di DomoML per mezzo del quale si rende possibile l'interoperabilità tra i diversi standard domotici e senza il quale DomoNet sarebbe una semplice interfaccia nei confronti di varie tecnologie.

I LinkedService sono formalizzati mediante il linguaggio DomoML e permettono, mediante il proprio formalismo, l'associazione dei servizi resi disponibili dai vari device domotici.
Superata la fase di trasformazione da device domotici in DomoDevice ogni dispositivo si presenta come una collezione di servizi indirizzabili che possono ricevere dei dati in input e possono produrre dei dati in output.

In queste condizioni l'azione di dover creare il "linkaggio" dei servizi, viene semplificata passando le coordinate del servizio da chiamare all'interno del servizio chiamante.

Attraverso il costrutto dei LinkedService, via via evolutosi nel tempo, si possono definire diverse azioni semantiche al fine di garantire un passaggio di dati corretto non solo a livello di tipi di dati e di sintassi, ma anche a livello di interpretazione dei dati stessi da parte dei TechManager.
Per la visione di qualche esempio e per maggiori dettagli vedere il par. 2.2 (pag 42-45) di [3].

[4]Nella parte logica dell'applicazione è gestito l'invio solamente al *TechManager* di interesse e non a chiunque.

N	Firma metodo
1	`String getAllDomoDeviceList ()`
2	`String getDomoDeviceList ()`
3	`void registerToClientsUpdate (String in0, String in1)`
4	`void registerToClientsUpdatePort (String in0)`
5	`void domoDevicesUpdate (String in0)`
6	`void finalize ()`
7	`String execute (String in0)`

Tabella 1.1: La firma dei metodi del web Service DomoNetWS esposti con una sintassi Java.

1.2 DomoNetWS

DomoNetWS è un Web Service e svolge il lato server dell'applicazione DomoNet offrendo le sue funzionalità attraverso i metodi elencati nella tabella 1.1.
L'*engine* del DomoNetWS ha il compito di:

- gestire la cooperazione tra le diverse tecnologie domotiche;
- permettere il controllo dello stato dei device;
- processare le richieste (da parte di altri web service o dei client come mostrato in fig. 1.2) che arrivano come invocazioni dei metodi della tabella 1.1.

Nel momento in cui il DomoNetWS viene avviato, carica tutti i *TechManager* presenti nella *techManagerList*[5] che a loro volta recuperano e restituiscono tutti i device collegati. Per ogni device restituito dai techManager viene generato secondo il formalismo DomoML un *DomoDevice* che viene inserito nella DomoDeviceList.

Le interazioni (mostrate nella figura 1.2 con delle frecce di *request* e di *response*) mostrano l'invocazione di un generico metodo del Web Service che in generale[6] fornisce una risposta. Uno dei casi più frequenti (ed anche

[5]Anche se non viene riportata nelle immagini il DomoNetWS mantiene una lista di TechManager che vengono presi in considerazione in fase di installazione in base alle tecnologie domotiche presenti.

[6]Nel caso specifico dei metodi *void* la freccia *response* è nulla.

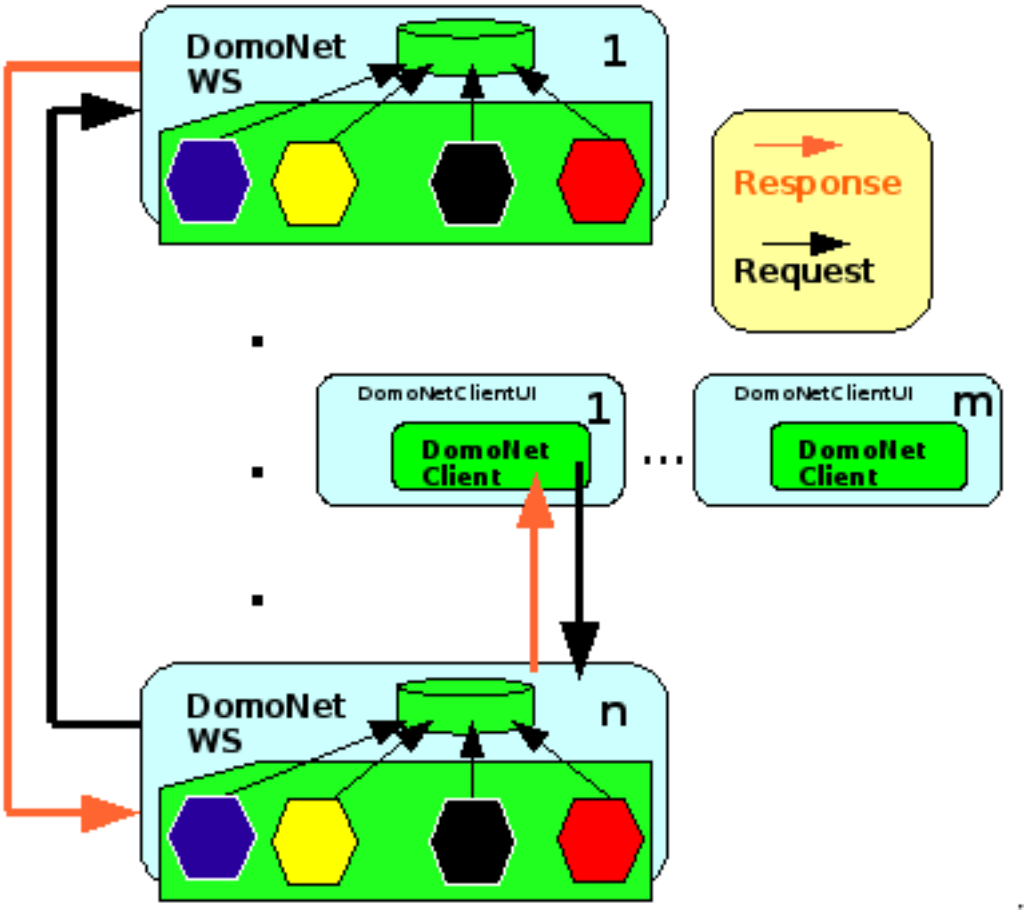

Figura 1.2: Interazioni tra i DomoNetWS e i DomoNetClient in un scenario generalizzato con n DomoNetWS e m DomoNetClient.

più significativi) è l'invocazione del metodo *execute* parametrizzato con un DomoMessage in formato stringa (*in0*).
Le richieste di *execute* di un comando possono provenire dallo stesso web service, da un altro web service (per implementare la cooperazione) o da un client (per il controllo e la modifica dello stato dei dispositivi).

Le azioni che si susseguono all'interno del DomoNetWS hanno la seguente sequenza temporale:

1. il DomoNetWS ricostruisce il DomoMessage a partire dalla stringa *in0* e lo invia al rispettivo[7] *TechManager*;

[7]Il DomoMessage viene inviato a un solo TechManager perché il DomoMessage ha come

2. il TechManager provvede a processare il domoMessage intraprendendo un'azione corrispondente nel proprio *middleware*. Il risultato che fornirà a DomoNetWS sarà un domoMessage di successo o di fallimento;
3. il DomoNetWS ricevuto il *response*, lo invia al mittente della richiesta.

1.2.1 TechManager

Il *Tech manager* è un modulo (o driver) del web service che ha la funzione di interfacciarsi fisicamente con i dispositivi appartenenti ad una determinata tecnologia (Nella figura 1.1 le tecnologie sono rappresentate dai *Middleware* Konnex, UPnP, BTicino, BACnet) gestendo la corretta interazione.
Il ciclo di vita di un TechManager prevede:

1. il riconoscimento di quei dispositivi che erano stati caricati in precedenza nel *DomoNetWS* con il riuso delle precedenti configurazioni evitando dunque la crezione di una copia inutile e potenzialmente inconsistente in memoria;
2. l'esecuzione del metodo *start* necessario a inizializzare tutte le strutture dati e gli eventuali ulteriori processi di supporto definiti e usati all'interno dello specifico TechManager.

Finita la fase di inizializzazione il TechManager risulta disponibile a ricevere non deterministicamente:

- richieste inviate dal *DomoNetWS* (tali richieste vengono processate con il metodo *execute* che provvederà a creare anche il messaggio di risposta);
- gli aggiornamenti di stato rinvenuti dai dispositivi presenti nel *middleware* in gestione.

Il TechManager prevede inoltre il metodo *finalize* usato nei casi in cui il *middleware* preveda l'esecuzione di procedure di uscita.

1.3 DomoNetClient

Il DomoNetClient come raffigurato nella figura 1.1 è composto da un'interfaccia utente e dall'*engine*.

destinatario un DomoDevice e tutti i domoDevice mantengono un campo che permette di riconoscere la tecnologia di appartenenza.

L'*engine* viene richiamato semplicemente per invocare i metodi del DomoNetWS e quindi ottenere i response testuali in formato DomoML. L'interfaccia utente viene usata per offrire all'utente la possibilità di collegarsi a un DomoNetWS e dunque interagire con i dispositivi in esso contenuti.
La figura 1.3 mostra la maschera di avvio del DomoNetClientUI.

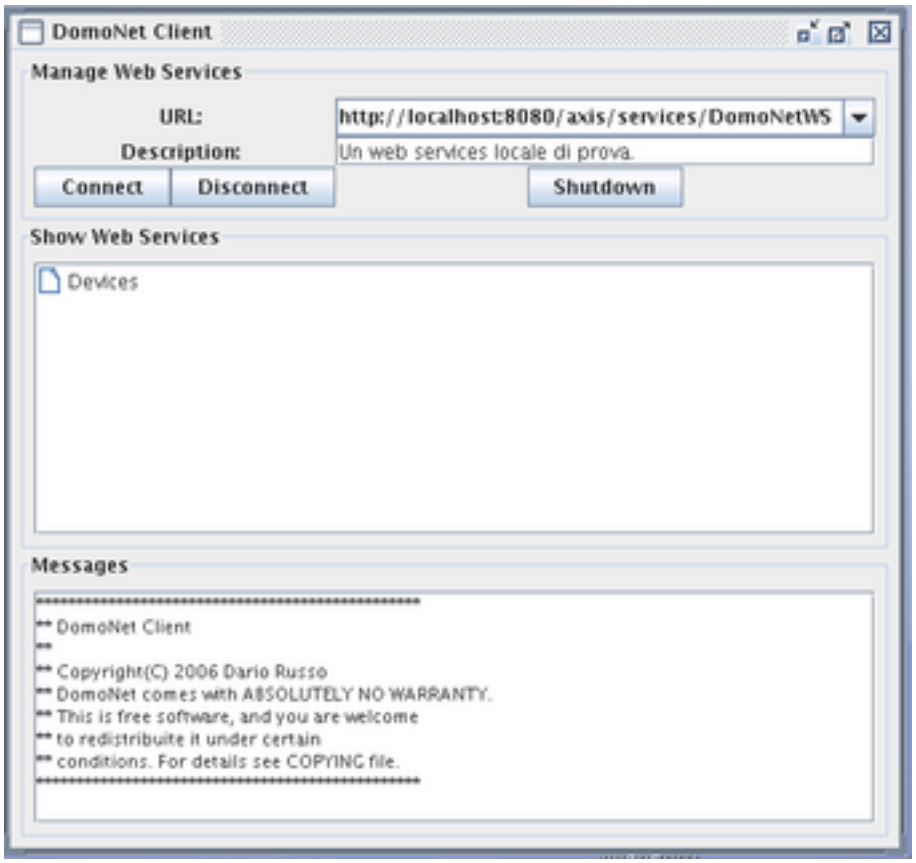

Figura 1.3: Interfaccia utente della DomoNetClientUI.

In alto a destra nel campo URL si inserisce l'indirizzo del DomoNetWS e per effettuare una connessione occorre cliccare sul bottone *Connect*.
Una volta collegati nel box *Show Web Services* verranno mostrati i dispositivi scoperti dal DomoNetWS. A questo livello la rappresentazione dei dati è tutta in DomoML e dunque attraverso l'interfaccia sarà possibile cliccando sui servizi dei domoDevice invocare dei comandi passando dei parametri vincolati dai tipi definiti in DomoML.
Le richieste possono essere:

- ottenere lo stato di un dispositivo invocando servizi di Get (Es: conoscere lo stato di una lampadina [accesa o spenta]);
- richiedere il cambio di stato di un dispositivo invocando servizi di tipo Set (Es: nel caso più semplice accendere una lampadina spenta o spegnere una lampadina accesa passando come parametro lo stato obiettivo);
- anche se l'utente non agisce attivamente, nella fase di *Connect* viene richiesta la lista di *DomoDevice* invocando (a livello di *engine*) il metodo *getDomoDeviceList*.

Capitolo 2

Lo standard BACnet

BACnet è un protocollo standard di *Comunicazione Dati per l'automazione di edifici e per le reti di sensori*[1], sviluppato dalla società American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) [8].
BACnet nasce dalla frustrazione causata dall'inadeguatezza dei sistemi di controllo (proprietari) nel comunicare con altri sistemi all'interno di un edificio o con i sistemi di altri produttori[10]. Per questa ragione fornisce [9] l'infrastruttura di comunicazione necessaria per:

- integrare **prodotti** realizzati sia da differenti case produttrici, sia per *target* differenti;
- integrare **servizi** offerti in modo indipendente da ciascun prodotto.

Il protocollo BACnet oggi è:

- uno standard nazionale in America;
- uno standard Europeo;
- uno standard adottato in oltre 30 paesi;
- uno standard ISO [2] globale;

e per le sue similitudini con il framework DomoNet ha suscitato molto interesse ipotizzando uno sviluppo congiunto verso un'unica grande piattaforma capace di offrire la massima interoperabilità con molti standard domotici.

[1] BACnet è l'acronimo di **B**uilding **A**utomation and **C**ontrol **Net**works.

[2] L'ISO, o *International Organization for Standardization* (Organizzazione internazionale per le standardizzazioni), è la più importante organizzazione a livello mondiale per la definizione di standard industriali e commerciali. Suoi membri sono gli organismi nazionali di standardizzazione di 157 Paesi del mondo.
Per maggiori dettagli visitare `http://it.wikipedia.org/wiki/ISO`

1 Aspetti Tecnici

BACnet fornisce un modello per trasferire dati come:

- valori di binary input e output;
- valori presi dall'Hardware come analog input e output;
- informazioni come allarmi ed eventi;
- quelli contenuti all'interno di un file, o quelli ricavati dalla logica di controllo.

BACnet non definisce né la configurazione interna, né le strutture dati, e tanto meno la logica di controllo dei controllori o dei dispositivi su cui viene eseguito il protocollo.
Il *focus* è spostato sull'informazione che necessita essere visibile sulla rete di comunicazione astraendo dai dettagli implementativi. Il tutto si realizza attraverso l'uso di **oggetti standard**. Il mapping tra oggetti standard e i dati o i processi sottostanti è lasciato a carico del produttore del dispositivo. Un ricco insieme di servizi è definito dallo standard ma sono definite anche una serie di specifiche (per capirne l'utilizzo), cosicché lo standard può essere implementato dai dispositivi, anche quelli con un ampio numero di funzionalità.

1.1 I Dispositivi di BACnet

Una rete BACnet è in grado di gestire fino a 4194305 dispositivi, poiché ogni dispositivo va identificato con un indirizzo unico e, in tutto l'*internetwork* BACnet, gli indirizzi disponibili vanno da 0 a 4194304.
Il generico dispositivo BACnet viene modellato per essere visto dall'esterno come un contenitore di oggetti.
Tra tutti gli oggetti che un dispostivo può contenere, è richiesta dallo standard la presenza di **un solo** oggetto di tipo Device, che ha il compito di mantenere nelle sue properties tutte quelle informazioni che caratterizzano il dispositivo.
Lo stato dell'*internetwork* BACnet è dinamico, in quanto i dispositivi possono essere rintracciati "a caldo" per cui la rete BACnet riesce a gestire autonomamente i cambi di stato di un dispositivo.

1.2 Gli Oggetti standard

Dato che la progettazione e la configurazione interna di un dispositivo sono a carico dei produttori e in generale variano per ogni dispositivo, BACnet definisce una *collezione di strutture dati astratte* chiamate **oggetti** a cui ogni produttore può fare riferimento.
Un oggetto al suo interno è costituito da una collezione di proprietà che rappresentano i vari aspetti hardware, software e operazionali che il dispositivo possiede. Attraverso gli oggetti, BACnet fornisce un mezzo di identificazione e di accesso alle informazioni senza richiedere conoscenze di dettaglio sulla progettazione interna del dispositivo. All'interno dello standard sono definite le seguenti tipologie di oggetti:

Tipo di Oggetto	**Caso d'uso**
Analog Input *	Sensor input, mostra il valore registrato da un generico sensore.
Analog Output *	Control output, raccoglie dei dati che l'utente deve inserire.
Analog Value *	Un Setpoint o altro parametro del sistema di controllo analogico.
Binary Input *	Switch input in formato binario (i valori possibili sono (1|0).
Binary Output *	Per trasmettere dei valori binari in output.
Binary Value *	Parametro del sistema di controllo in Binario (digitale).
Calendar	Definisce una lista di date, come le ferie o eventi speciali, per lo scheduling.
Command	Per scrivere valori multipli su multipli oggetti in molteplici dispositivi che dovranno adempiere uno specifico scopo. Per esempio per passare da una modalità giorno a una modalità notte, o modalità emergenza (dove in generale possono essere coinvolti più entità).

[2] I tipi di oggetto asteriscati con * supportano opzionalmente il *change of value reporting* (COV)

Device	Le Properties di questo oggetto mantengono informazioni come - quali oggetti e servizi il dispositivo supporta, ma anche informazioni specifiche del dispositivo come il produttore o la revisione del firmware ecc... Questo oggetto è sempre presente in unica istanza all'interno di un dispositivo BACnet.
Event Enrollment	Descrive un evento che può essere una condizione di errore (per es: l'Input è out of range) o un allarme che altri dispositivi sanno riconoscere. Può dire direttamente a un dispositivo di usare un oggetto della Notification Class o dirlo a molteplici dispositivi.
File	Permette l'accesso in lettura e in scrittura ai dati del file supportati dal dispositivo.
Group	Fornisce l'accesso a multiple properties o a multipli oggetti in una singola operazione di lettura.
Loop *	Fornisce un accesso standardizzato ai "control loop".
Multi-state Input *	Rappresenta lo status di un processo multi-status. Per esempio può modellare lo stato di un refrigeratore che può essere in stato On, Off o nel ciclo di sbrinamento.
Multi-state Output *	Rappresenta lo stato desiderato di un processo multi-status (Per esempio il refrigeratore è in fase di raffreddamento, vorrebbe essere abilitato il passaggio alla fase di sbrinamento).
Notification Class	Contiene una lista di dispositivi che devono essere informati qualora un oggetto di Event Enrollment stabilisce che è necessario spedire un *warning* o un messaggio di allarme.
Program	Dato un programma eseguibile nel dispositivo, con questo oggetto è possibile conoscere lo stato di avanzamento. Inoltre, permette dall'esterno la modifica di tale stato di avanzamento. Per esempio si può decidere di mandare in esecuzione, stoppare, caricare o scaricare il programma.

Schedule	Definisce uno scheduling settimanale di operazioni (realizzate scrivendo a una lista specifica di oggetti con eccezioni come le ferie. Può usare un oggetto Calendar per le eccezioni).

1.2.1 Un esempio: L'Analog Input Object

Come riportato in [10] per capire come sono relazionate le entità discusse, viene mostrata la figura 2.1.

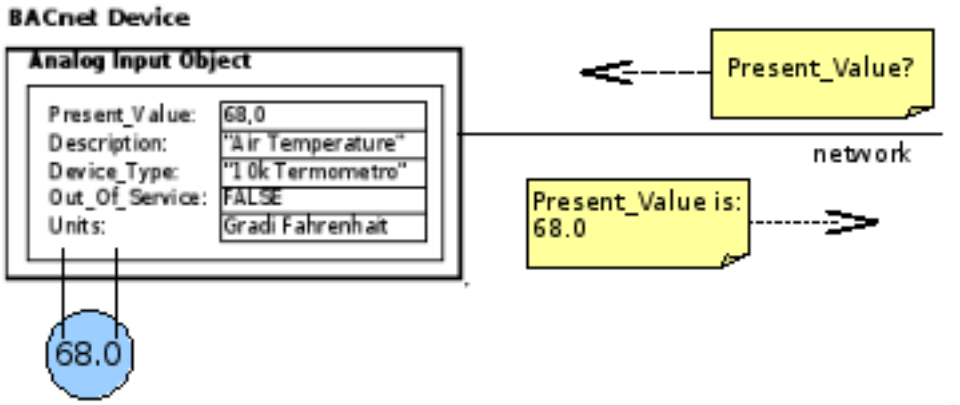

Figura 2.1: Un esempio di oggetto standard: L'Analog Input Object [9]

Come accennato già nella tabella riportata al par. 1.2 degli oggetti standard, un Analog Input Object può essere usato per rappresentare un sensore analogico di input così come un *thermistor* (sensore di temperatura che cambia la sua resistenza elettrica appena percepisce un cambio di temperatura). La figura 2.1 mostra un diagramma in cui un dispositivo con un sensore di temperatura contiene nella sua parte logica un Analog Input Object. Quest'ultimo espone sulla rete le sue cinque proprietà.
Alcune di queste proprietà come *Description*, *Device_Type*, e *Units* sono settate durante l'installazione.
Altre, come *Present_Value* e *Out_Of_Service*, forniscono lo stato del sensore di input rappresentato dall'Analog Input Object. Ancora altre (un Analog Input Object può avere fino a 25 proprietà) possono essere settate dall'equipaggiamento del produttore. In questo esempio, la query sulla property *Present_Value* di questo Analog Input Object darà come risposta "68.0".

1.3 Le Properties

Lo standard BACnet identifica 123 Properties differenti che possono essere contenuti all'interno degli Oggetti [10].
Senza entrare nel dettaglio di ogni Properties si può affermare che la specifica definisce i seguenti vincoli a riguardo:

- ciascun tipo di Oggetto ha associato un differente sottoinsieme di Properties definito dalle specifiche;
- l'identificativo di una Property caratterizza il valore contenuto all'interno d'essa;
- alcune Properties devono essere presenti qualunque sia il tipo di Oggetto, altre Properties sono invece opzionali;
- le Properties sono opzionalmente scrivibili, mentre sono tutte leggibili;
- possono esistere (e quindi possono essere nuove Properties) delle Properties proprietarie, ed è onere del produttore stabilirne le modalità di accesso.

Per maggiori dettagli si suggeriscono le fonti [9] e [10].

1.4 I Servizi

Se gli oggetti offrono la rappresentazione astratta della porzione di dati del dispositivo BACnet che dovrà essere "visibile dalla rete", i **Servizi** sono il mezzo attraverso cui un dispositivo BACnet:

- acquisisce quelle informazioni "visibili dalla rete" da un altro dispositivo;
- richiede ad un altro dispositivo l'esecuzione di qualche azione;
- annuncia a uno o a più dispositivi che qualche evento è avvenuto.

Ciascuna *service request* emanata e *service acknowledgment* (reply) ritornata diviene un pacchetto messaggio trasferito sulla rete da un dispositivo mittente verso un dispositivo ricevente.
I servizi possono essere "Confirmed" ovvero si attende una risposta con i dati, oppure "Unconfirmed" se invece non si attende risposta.
I servizi definiti dallo standard sono 35 e sono classificati in sei categorie:

- Allarmi e Eventi
- Accesso ai File

- Accesso agli Oggetti
- Gestione Remota del dispositivo
- Servizi di Virtual Terminal
- Sicurezza

Le cose da rilevare sono:

- i dispositivi **non** sono obbligati a fornire l'implementazione di tutti i servizi;
- l'unico servizio che ogni dispositivo è obbligato a fornire è il *ReadProperty* appartenente alla categoria *Accesso agli Oggetti*;
- attraverso l'uso dei servizi viene gestito il *COV* (change of Value) ovvero, il meccanismo a eventi con cui BACnet gestisce (dove previsto) il cambio di stato delle properties (implementato con paradigmi noti come *subscribe/notify*).

Per maggiori dettagli si suggeriscono le fonti [9] e [10].

2 Lo Stack BACnet

BACnet ha un'architettura a protocollo straficato basato su una versione collassata dello standard ISO-OSI.

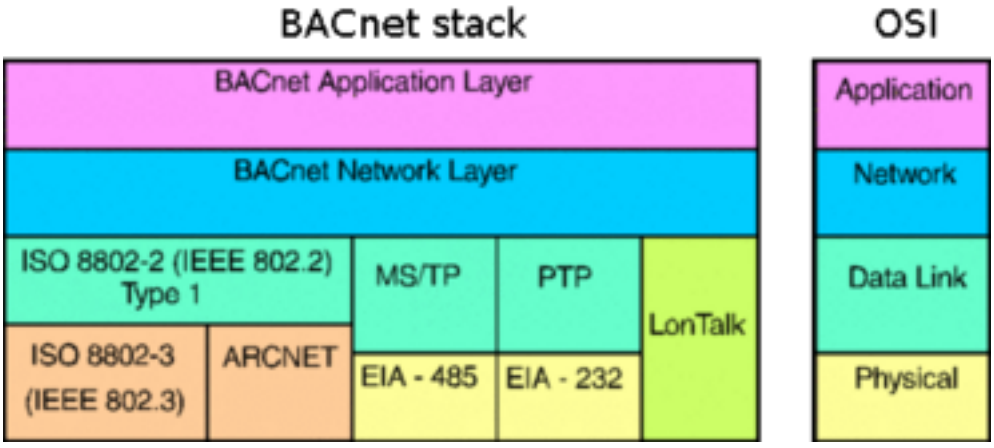

Figura 2.2: Lo stack di BACnet e gli strati equivalenti dello stack ISO-OSI [9]

Osservando la figura 2.2 è possibile notare tutti gli strati dello stack BACnet con la collocazione equivalente nello stack ISO-OSI.
Come in altri esempi di protocolli a pila, un ruolo importante è svolto dal messaggio che viene trasmesso da un dispositivo a un altro. Il messaggio in questi casi prevede una struttura organizzata in 2 parti principali:

- l'*Intestazione* usata per mantenere informazioni, spesso ridondanti, per garantire la correttezza e per gestire il routing;
- il *Body* usato per contenere le informazioni che devono essere effettivamente comunicate (Il PayLoad).

Nel caso di BACnet (come nella gran parte dei protocolli a pila) lo strato Application si preoccupa di riempire il *Body* ed eventualmente parti dell'intestazione. Tutti gli altri strati sottostanti l'Application si preoccupano di aggiungere nell'intestazione tutta una serie di informazioni necessarie a gestire le esigenze dello strato al fine di offrire quelle funzionalità utili agli strati superiori.
Nei seguenti sottoparagrafi verranno illustrate le funzionalità offerte dai vari strati.

2.1 Strato Application

Lo strato Application di BACnet racchiude sia il **modello astratto dell'informazione** cioè gli **oggetti**, sia i **servizi** usati per scambiare le informazioni tra i dispositivi, i cui dettagli sono stati già affrontati in precedenza.

2.2 Strato Network

Lo scopo di questo strato è fornire il mezzo attraverso cui i messaggi possono essere trasferiti da una rete BACnet a un'altra, indipendentemente dalla tecnologia data-link in uso. Dovendo gestire dei *media* molto distinti, un vincolo che si ha a questo livello sta nel fatto che non esiste la segmentazione e il riassemblaggio dei messaggi (che risultano a carico dello strato Application). BACnet impone un limite sulla lunghezza dei messaggi che passano dai router. La lunghezza massima non dovrà mai eccedere la capacità di un qualsiasi data-link incontrato lungo il *path* da mittente a destinatario.

L'entità principale a questo livello è il router. BACnet stabilisce quali sono i compiti da adempiere:

- gestire la propria tabella di routing;
- cercare il *path* per raggiungere il dispositivo destinatario;
- gestire il fatto che può esistere al più un solo *path* attivo tra 2 dispositivi;
- indicare al dispositivo cliente se al path richiesto non è possibile localizzare alcun destinatario.

Data l'eterogeneità della rete, BACnet definisce inoltre cosa deve fare un router per effettuare il "tunneling" di un pacchetto quando questo deve attraversare reti IP o IPX[3].

2.3 Strato Data Link e strato Fisico

Come mostrato nella figura 2.2 BACnet fornisce 4 soluzioni di tecnologia LAN e una soluzione che prevede il protocollo *point-to-point*. La scelta di una particolare soluzione dipende da ragioni come velocità, chip o schede disponibili a implementare il protocollo, grado di compatibilità con i sistemi esistenti, costo, ecc.
La flessibilità a questo livello permette al progettista di scegliere sempre la soluzione più appropriata ai particolari scenari di applicazione che si presentano.
Per esempio, nei grandi sistemi frequentemente si fa uso di soluzioni miste organizzate in strutture gerarchiche, per cui i controllori di campo, risiedendo agli estremi, saranno collegati direttamente a reti a basso costo e a bassa velocità, e saranno supervisionati da altri controllori che sono interconnessi da reti ad alta velocità come la LAN.
Per completezza viene fornita una descrizione generale delle 5 tecnologie gestite da questi strati, ma per maggiori dettagli, si consiglia la consultazione dei riferimenti che saranno via via richiamati.

[3]IPX/SPX sta per Internetwork Packet Exchange/Sequenced Packet Exchange. É un protocollo di rete usato dal sistema operativo di rete Novell NetWare. Come l'UDP, IPX è un protocollo datagramma usato per comunicazioni senza connessione. Per maggiori dettagli visionare `http://it.wikipedia.org/wiki/IPX/SPX`

Ethernet o *ISO 8802-3* [4] rappresenta la soluzione con le prestazioni più alte. Offre velocità fino a 100 Mbps ed è ormai una tecnologia molto diffusa (soprattutto con l'avvento dell'ADSL ormai è di uso comune vedere un ambiente domestico cablato con una rete di computer funzionante attraverso questo protocollo).
Ethernet utilizza il protocollo *CSMA/CD* (carrier sense multiple access with collision detection)[5] per gestire il mezzo fisico condiviso. Una caratteristica di questo protocollo è che in presenza di collisione viene diminuita la banda di trasmissione dati e per questa ragione non è possibile predire, con assoluta certezza, quanto tempo occorre affinché un dispositivo abbia la possibilità di trasmettere con successo un messaggio.
I miglioramenti futuri di questo protocollo si tradurranno automaticamente in miglioramenti anche per BACnet.

ARCNET fu sviluppato dalla Datapoint Corporation. Oggi è uno standard nazionale Americano (ATA/ANSI 878.1). ARCNET è un'alternativa a basso costo operante alla rispettabile velocita di 2.5 Mbps. ARCNET afferma di essere la LAN commercialmente disponibile più antica con quasi 3 milioni di nodi installati.
Lo standard ARCNET è incluso in BACnet per riferimento.
ARCNET è un protocollo a *token circolante* e per questo risulta deterministico, ovvero è possibile stabilire un tempo massimo entro il quale il dispositivo dovrà attendere prima di avere la possibilità di trasmettere un messaggio.
Come per Ethernet un eventuale miglioramento futuro coinvolgerà anche BACnet.

EIA-485 *EIA* sta per (Electronic Industries Association) [6]. Questo standard, definito per lo strato fisico è comunemente usato nei sistemi di controllo di edifici, in modo particolare per reti di sensori per applicazioni specifiche.
BACnet definisce un protocollo *MS/TP* (Master-Slave/Token-Passing) per fornire le funzionalità del datalink per accedere a questo *media*.

[4] Per dettagli sullo standard ISO 8802-3 visitare l'indirizzo
`http://it.wikipedia.org/wiki/IEEE_802.3`

[5] Per dettagli andare al paragrafo 5.5.2 di [15]

[6] Per dettagli visitare il sito:
`http://www.interfacebus.com/Design_Connector_RS485.html`

In una rete MS/TP ci possono essere uno o più nodi *master* equivalenti e dei nodi *slave* che non possono trasmettere messaggi fintanto che non viene loro richiesto da un nodo *master*. L'accesso al mezzo viene concesso con la tecnica del *Token-Ring* [7].

LonTalk è un protocollo proprietario sviluppato dalla Echelon Corporation organizzato in 7 livelli e implementato su un singolo chip chiamato *neuron*.
I dispositivi basati sul protocollo LonTalk non sono, in generale, interoperabili perché il protocollo non impone loro nessuna particolare funzionalità applicativa.
In BACnet il protocollo LonTalk viene usato solo come mezzo per trasferire dati da un dispositivo a un altro proprio come per le altre soluzioni LAN. L'interoperabilità viene migliorata usando lo stesso strato *Application* e *Network* come avviene per tutti gli altri dispositivi BACnet.

PTP o protocollo *Point-To-Point*, viene usato per accedere al mezzo condiviso attraverso la porta *EIA-232* [8]. Un'applicazione tipica potrebbe essere connettere a un modem un sistema domotico remoto. Il protocollo PTP non specifica come si stabilisce una connessione fisica.
Il protocollo PTP definisce come una comunicazione BACnet è inizializzata, mantenuta e terminata, tutto questo una volta che si è stabilita la connessione fisica.

3 ANNEX N - BACnet/WS L'Interfaccia Web Service di BACnet

In questa sezione verrà affrontato un aspetto cruciale dello standard BACnet, ovvero come questo standard riesce a fornire un'interfaccia di servizi di alto livello.
Per una visione completa di quanto argomentato in questa sezione si suggerisce la visione del documento [11].

[7] Per maggiori dettagli sulla tecnica del *Token-Ring* andare all'indirizzo
`http://it.wikipedia.org/wiki/Token_ring`

[8] Per maggiori dettagli sulla *EIA-232* andare all'indirizzo
`http://www.interfacebus.com/Design_Connector_RS232.html`

A pagina 2 del [11] troviamo:
"... This addendum defines a standard means of using Web services to integrate facility data from disparate data source, including BACnet networks, with a variety of business enterprise applications."

L'*ANNEX N* definisce il *modello dei dati* e l'*interfaccia del Web Service* per poter integrare strutture dati di qualunque forma dando vita a una grande varietà di applicazioni di *business management*. Il modello dei dati e l'accesso ai servizi risultano sufficientemente generici e dunque adatti per modellare e per accedere ai dati da qualunque sorgente.
L'implementazione dei servizi risulta conforme alle specifiche del *Web Services Interoperability Organization (WS-I) Basic Profile 1.0* [12] ovvero:

- l'utilizzo di Simple Object Access Protocol (SOAP) 1.1 over HTTP/1.1;
- l'uso di XML 1.0 nella codifica dei dati per il trasporto rispettando i tipi di dati con relative rappresentazioni lessicali e canoniche definite dal Worl Wide Web Consortium XML Schema.

Nelle seguenti sottosezioni verranno illustrate solo quelle parti che avranno un richiamo nell'implementazione del codice, per capire tutte le potenzialità, come detto sopra, è consigliabile visionare il documento originale [11].

3.1 Modello dei dati

Per poter scambiare le informazioni con il web Service, questo standard stabilisce un insieme minimale di requisiti per la strutturazione e l'associazione dei dati scambiati con il BACnet/WS server.
Nel modello dei dati viene definita l'entità *nodo* come un dato primitivo fondamentale per il BACnet/WS. I nodi sono organizzati in un sistema gerarchico e dinamico.
Esiste un nodo *radice* nella gerarchia denominato *root* caratterizzato dal fatto che non ha nodi genitori ma può avere nodi figli.
La parte di stato di un nodo "visibile alla rete" é esposta come una collezione di attributi.
Per poter interrogare la gerarchia vengono usati i *path*.
Tutte le entità qui richiamate saranno discusse nei paragrafi successivi.
Sommariamente il modello dei dati va visto come un'astrazione sui dati

costruita sopra quella già esistente di BACnet con lo scopo di offrire un interfacciamento ai dati, attraverso i metodi esposti dal web Service, più semplice rispetto al modello di BACnet.

3.2 Nodi

Il nodo rappresenta una modellazione astratta di un contenitore di dati con struttura gerarchica. Concettualmente un nodo ha le seguenti caratteristiche:

- può avere dei nodi figli;
- può avere un valore;
- mantiene una collezione di attributi che lo caratterizzano, di cui alcuni obbligatori e altri opzionali;
- ha un tipo che viene mantenuto tra gli attributi obbligatori;
- é una struttura dati pensata per essere dinamica;
- alcune parti del nodo possono essere modificate anche dall'utente, e queste modifiche avranno effetto sullo stato dell'entità corrispondente di BACnet.

3.3 Attributi

Definizione:

Un **attributo** è un singolo aspetto o qualità di un nodo. (N.1 di [11])

Esempi di attributi di un nodo sono il valore del nodo, oppure sapere se quel nodo è modificabile dall'utente oppure no.
Ogni nodo espone una collezione di attributi. Alcuni attributi sono richiesti per tutti i nodi, altri invece sono opzionali o la loro presenza è condizionata da altri fattori.
Una lista completa di attributi standard è visionabile a pagina 7 punto N.8.4 del [11].

3.4 Path

Definizione:

Un **Path** è una stringa di caratteri che viene usata per identificare un nodo o uno specifico attributo. (N.2 di [11])

La gerarchia dei nodi si riflette in un *path* come una gerarchia di identificatori

di nodo separati da una serie di caratteri *slash* ("/").
Allo stesso modo, anche gli attributi possono avere una struttura gerarchica e tale gerarchia si riflette in un *path* come una gerarchia di identificatori di attributi separati da una serie di caratteri *colon* (":").
La sintassi di un *path* è la seguente:
/ [*id_Nodo*[/ *id_Nodo*] ...] [: *id_Attributo*[: *id_Attributo*] ...]

dove le parentesi quadre indicano l'opzionalità dell'elemento e i " ..." indicano la ripetibilità dell'elemento.
Tutti gli identificatori sono *case sensitive*, di lunghezza non nulla e possono essere composti solo da caratteri stampabili. Risulta possibile avere degli identificatori contenenti degli spazi, tuttavia viene sconsigliato l'uso.
Un'ulteriore aspetto da notare è che ASHRAE ha definito dei nodi e degli attributi riservati per lo standard. Dunque se un nodo è riservato allora al suo identificatore è anteposto il carattere ".", al contrario se un attributo è riservato allora il suo identificatore non ha anteposto il carattere ".", per chiarire meglio l'argomento osservate la tabella seguente:

	Riservato	**NON Riservato**
Attributo	/:attr	/:.attrFree
Nodo	/.26	/18

4 Esempi di dispositivi BACnet

In questa sezione verranno discussi alcuni esempi di dispositivi BACnet. Ognuno di questi va inquadrato tenendo conto di 2 aspetti fondamentali:

1. lo standard BACnet, come già detto, non ha il compito di determinare come si costruisce l'Hardware, ma definisce le specifiche del protocollo di comunicazione, per cui descrive delle caratteristiche che coinvolgeranno solo una parte della logica del processo in esecuzione sul dispositivo.
   Per questa ragione, il protocollo può essere implementato sia da un generico computer che da uno specifico dispositivo realizzato da una qualunque azienda in grado di rispettare le specifiche BACnet;

2. lo scopo principale di un protocollo "industriale" è quello di automatizzare dei processi industriali, ovvero processi in cui sono presenti delle macchine che eseguono dei compiti ben precisi, pertanto sono equipaggiate con dei sensori e con degli attuatori. Il generico dispositivo Hardware di BACnet deve quindi essere pensato come una "scatola" in grado di interfacciarsi alle macchine industriali, per cui avrà delle porte di ingresso e di uscita al quale si collegheranno i sensori e gli attuatori. I collegamenti saranno tipati come *input* e *output*. Dagli input del dispositivo, il protocollo BACnet associerà oggetti tipici di Input (tipicamente Analog-Input o Multi-state Input a seconda dei casi) e li usarà per leggere i dati. Quando invece dovrà inviare dei comandi dovrà usare i canali di Output che sono logicamente modellati con degli oggetti tipici per l'Output (tipicamente Analog-Output o Multi-state Output a seconda dei casi).

4.1 Dispositivi simulati

In questa sezione verranno discussi i dispositivi di BACnet che vengono simulati attraverso il computer mediante delle applicazioni opportunamente configurate.
I primi 2 esempi sono realizzati dall'azienda *SCADA Engine* [13] e sono dei software di prova con qualche limitazione d'uso. L'ultimo esempio invece è un progetto Open Source guidato da Steve Kargs[9].

4.1.1 SCADA Bacnet Web Service

In tutto il panorama di aziende che curano lo standard BACnet la *SCADA Engine* allo stato attuale risulta l'unica a fornire un'interfaccia di alto livello come i Web Service. All'indirizzo:
`http://www.scadaengine.com/web_service/web_service.html`
viene fornita una spiegazione generale di come usare l'applicazione con diversi esempi e immagini.
È installabile solo su ambiente Windows e necessita delle librerie WinCap[10] per gestire lo strato *data-link* dell'applicazione (In fase di installazione

[9] Il suo sito personale è raggiungibile all'indirizzo `http://steve.kargs.net/`.
[10] Per maggiori dettagli andare all'indirizzo web:
`http://www.winpcap.org/`

l'applicativo stesso rileva l'eventuale necessità d'installazione della libreria). La configurazione risulta minimale e vengono mostrate delle configurazioni valide nella figura 2.3.
Per quanto riguarda i parametri relativi all' *APDU* vanno lasciati quelli impostati di default. Gli unici sul quale si possono creare delle personalizzazioni sono *Name* usato per fornire la descrizione del dispositivo BACnet all'interno della rete BACnet e *Instance* sul quale si setta il *DeviceID* del BACnet Web Service.
Un altro settaggio possibile è il media su cui deve agire BACnet. Gli esperimenti fatti, hanno sempre usato il *BACnet/IP* sul quale viene assegnato identificativo di rete = 1. La trasmissione dei dati avviene su UDP sulla porta standard di BACnet corrispondente alla 47808 (Curiosamente in esadecimale questa porta viene tradotta in BAC0). Una volta partita l'applicazione, per

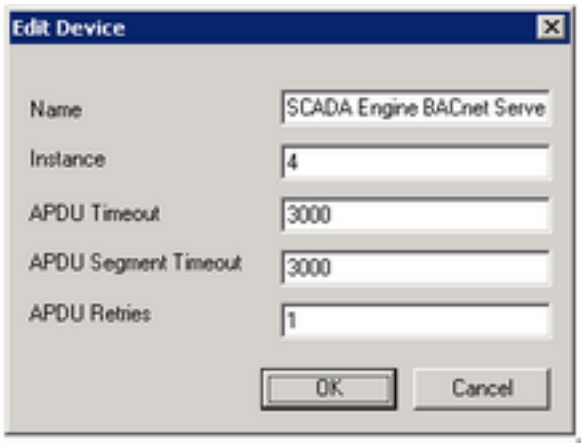

Figura 2.3: Configurazione del BACnet Web Service.

verificare l'effettivo funzionamento a livello di web service è sufficiente aprire un browser web e andare all'indirizzo `http://localhost:8080` ottenendo come risultato una pagina web come quella mostrata in figura 2.4. A questo punto l'applicazione è pronta all'uso ma per un tempo limitato che una volta trascorso bloccherà l'applicazione.
Questo problema si risolve parzialmente chiudendo e riaprendo l'applicazione.

Figura 2.4: Test attraverso il browser di funzionamento del BACnet Web Service.

4.1.2 SCADA Bacnet device simulator

Presenta lo stesso iter d'installazione e del *Bacnet Web Service*. Lo scopo di quest'applicazione è quello di simulare un dispositivo BACnet contenente una serie di *oggetti standard* di BACnet che presentano delle *properties* che possono essere sia lette e dove possibile anche scritte.
Finita l'installazione quando parte l'applicativo occorre cliccare una sequenza di bottoni obbligati dall'interfaccia fino a giungere sull'interfaccia principale mostrata nella figura 2.5.
A questo punto andare al menu *Server* e cliccare sul'unica voce possibile vale a dire *Create Object List*. Come risultato si ha l'apertura di un'altra finestra dove si potrà configurare il Device Simulator (figura 2.6).
Il campo *Frequency* indica la frequenza, ovvero ogni quanti secondi gli oggetti cambieranno il loro *PresentValue*. Il cambiamento può essere incrementale o casuale da scegliere con il radioBox sottostante.
Nella parte centrale si possono spuntare i tipi di oggetti che il simulatore è in grado di gestire (sono solo alcuni, come visto al par. 1.2 ma sono quelli che meglio si prestano per effettuare delle integrazioni con altri standard).
Nella configurazione del simulatore va assegnato il *DeviceID* che deve essere unico (come più volte segnalato nella specifica) in tutto l'internetwork di BACnet, l'ID di rete (che sarà sempre "1" per i nostri scopi) e l'ultimo

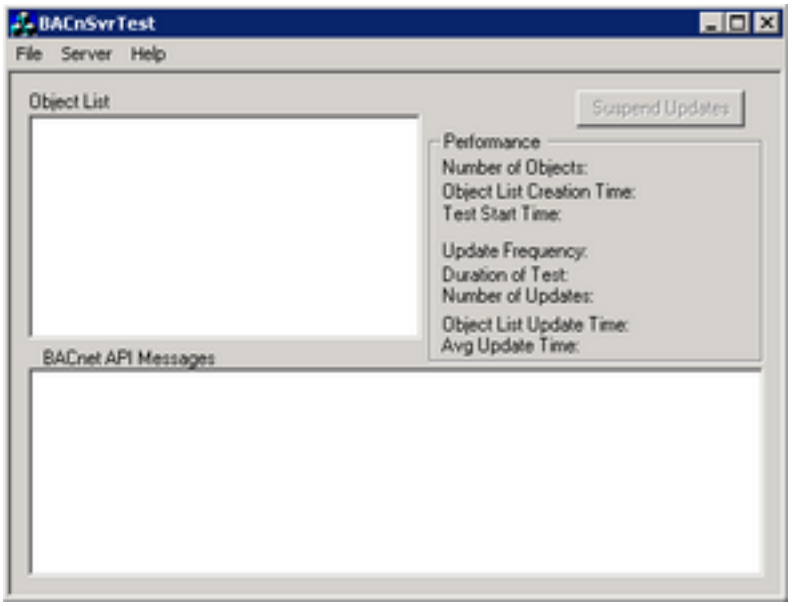

Figura 2.5: Interfaccia principale del BACnet Device Simulator.

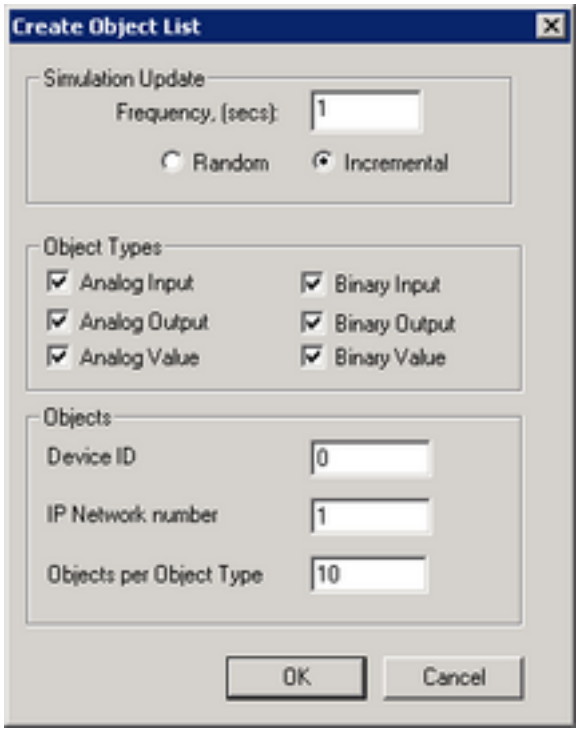

Figura 2.6: Interfaccia di configurazione del BACnet Device Simulator.

campo indica il numero di istanze di oggetto presenti nel dispositivo per ogni tipo di oggetto spuntato in precedenza.

Finita la configurazione si ritorna alla finestra principale che mostra lo stato del dispositivo con diverse informazioni statistiche (Figura 2.7).

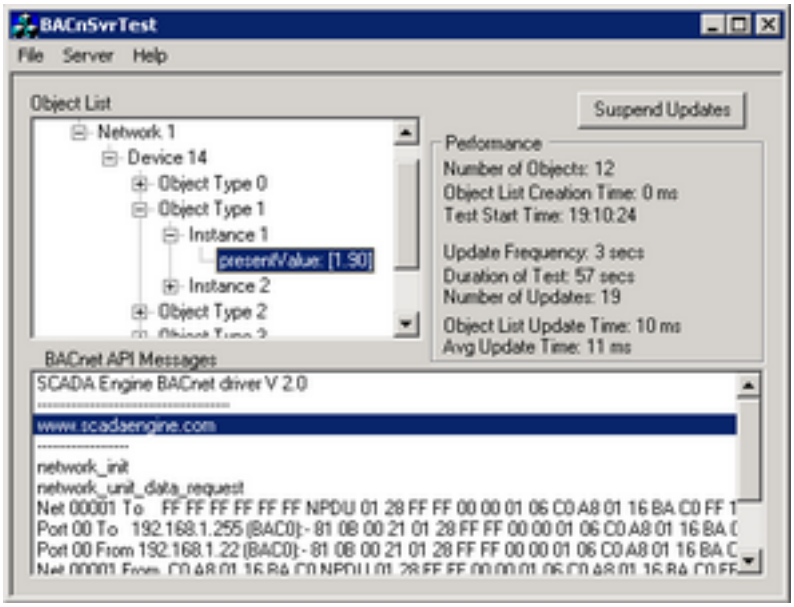

Figura 2.7: Interfaccia principale del BACnet Device Simulator dopo la configurazione.

4.1.3 BACnet Server Demo Open Source

L'applicativo è compreso scaricando lo stack BACnet Open Source dall'indirizzo:

`http://sourceforge.net/projects/bacnet`

In antitesi alla soluzione SCADA il Device Simulator Open Source ha le seguenti caratteristiche:

- È Open Source ed il linguaggio usato è il *C*. È dunque possibile personalizzare il suo comportamento modificando il codice.
- È multipiattaforma.
- Le variabili che espone non mutano automaticamente (per cui si presenta come un dispositivo statico) ma può essere comunque usato per testare i vari *demo* inclusi nel pacchetto dello Stack BACnet e per interfacciarlo con il BACnet Web Service. Tra i vari *demo* è possibile provare il funzionamento della lettura di un file e altri tipi di oggetti non caricati nello SCADA BACnet Device Simulator.

Per l'utilizzo è sufficiente scaricare il file compresso ed estrarlo. Supponiamo che a fine estrazione la directory che contiene tutto il pacchetto viene mappata nella variabile d'ambiente *${stackBACnet}*. A questo punto per lanciare il simulator è sufficiente spostarsi al path

```
${stackBACnet}/demo/server
```

ed eseguire il comando

```
make
```

Finita la compilazione si ottiene un eseguibile che una volta lanciato simulerà un dispositivo BACnet in esecuzione sulla rete BACnet/IP.

4.2 Dispositivi reali

BACnet, come più volte detto, non produce dei dispositivi ma si preoccupa di fornire le specifiche sia per il protocollo di comunicazione tra dispositivi sia per esporre le funzionalità dei dispotivi sotto forma di oggetti.
A livello esemplificativo saranno presentati 2 esempi di dispositivi reali per avere una visione più ampia sulla panoramica dei dispositivi.

4.2.1 AddMe III modello AM3-SB

La fonte da cui sono state raccolte le informazioni è la stessa azienda che produce il dispositivo ovvero la *Control Solutions. Inc. (Minnesota [USA])*.
La URL a cui si fa riferimento è

```
http://www.csimn.com/CSI_pages/AddMeIII-SB.html
```

Il dispositivo (vedere la figura 2.8)

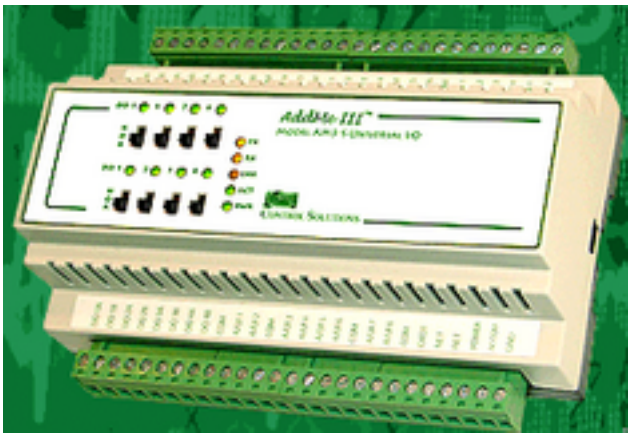

Figura 2.8: Dispositivo BACnet modello AM3-SB.

presenta una serie di ingressi e uscite e dal punto di vista logico offre una configurazione realizzata con i seguenti Oggetti:

- 40 Analog Input (20 uncommitted, writeable);
- 4 Analog Output (commandable);
- 8 Binary Output (commandable);
- 60 Multistate Input (60 uncommitted, writeable);
- 1 Device;
- 2 File;
- 1 Program.

Si connette alla rete BACnet attraverso la porta *EIA-485* per cui a livello Datalink usa il protocollo MS/TP e si presenta come nodo *Slave* (Per chiarimenti rivedere il par. 2.3).

4.2.2 Delta Controls

La conoscenza di questa azienda [14] è avvenuta alla *36-esima mostra convegno* ***expocomfort*** di Milano[11] tramite i nostri referenti BACnet italiani[12] che gentilmente ci hanno invitato a visionare una configurazione reale di una rete BACnet.

L'immagine 2.9 mostra una rete di dispositivi BACnet che, pur essendo diversi per natura e potenza di utilizzo, riescono ad interagire grazie al protocollo BACnet. I dispositivi, indicizzati nell'immagine, sono i seguenti:

1. mod. **DTW-333-OL**:
   Delta Touchscreen con workstation incorporata, consente la supervisione di qualunque sistema BACnet, è a tutti gli effetti una B-OWS (Bacnet Operator Workstation) con capacità grafiche.
2. mod. **DSC-T305**:
   Delta System Controller di categoria B-AAC con 3 ingressi analogici e 5 uscite digitali. È il più piccolo dispositivo liberamente programmabile della sua categoria. Consente, ad esempio, la gestione di oggetti

[11] Che è avvenuta il 11-15 Marzo 2008, per visionare il programma andare all'indirizzo: `http://www.mcexpocomfort.it/aspx/ShowFolder.aspx?idFolder=100`

[12] In particolare, i nostro contatti sono stati il sig. Salvatore Cataldi e il sig. Elio Milanese, che ringrazio per la collaborazione.

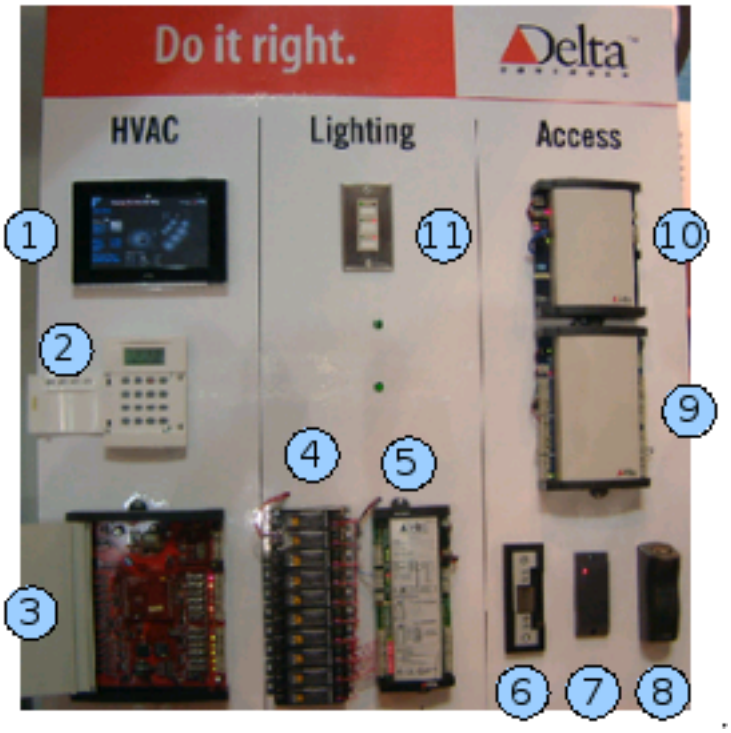

Figura 2.9: Rete di dispositivi BACnet realizzati dalla Delta Controls.

Schedule direttamente dal proprio display. È dotato di due interfacce di rete RS-485 MS/TP ed una porta PTP RS-232.

3. mod. **DSC-1280E**:
   Delta System Controller di categoria B-BC con 12 ingressi analogici, 12 uscite analogiche, due porte seriali MS/TP una porta Ethernet/IP, una porta PTP RS-232.
4. **Relays**:
   Sono un gruppo di relay, vale a dire degli interruttori elettrici controllati da altri circuiti elettrici.
5. mod. **DLC-G1212**:
   Delta Lighting Controller di categoria B-AAC con 12 ingressi digitali e 12 uscite digitali con feedback per controllo luci e sequencer.
6. **Serratura elettrica**.
7. **Prox reader**:
   Il Proximity Reader è una periferica usata per leggere card, badge poste in prossimità d'essa (non è necessario il contatto!).

8. **Fingerprint Scanner**:
   È una periferica che permette l'autenticazione dell'utente attraverso l'impronta digitale delle dita.
9. mod. **ADM-2W704**:
   Modulo di controllo varco con due porte per lettori di badge wiegand[13], diversi ingressi/uscite digitali per gestione dei dispositivi del varco.
10. mod. **ASM-24E**:
    Access System Manager, router per la gestione dei varchi di accesso dotato di porta seriale MS/TP RS-485 e di porta Ethernet. Consente la gestione di tutte le logiche di accesso tra i varchi direttamente dipendenti ed i varchi remoti (dipendenti da altri ASM-24E)
11. **Interruttori di corrente**.

[13] **Wiegand** è una tecnologia ampiamente riconosciuta e garantita da 18 anni di sperimentazione sul campo.
Per effetto Wiegand si intende la generazione, in un filo in lega speciale, di impulsi elaborati in modo che si vengano a creare due aree magnetiche distinte, dette *shell* e *core*, nella stessa parte omogenea di filo. Queste due aree reagiscono in modo diverso all'applicazione di un campo magnetico. La *shell* inverte la polarità solo in presenza di un forte campo magnetico, mentre il *core* la inverte anche con campi magnetici più deboli. Quando la *shell* e il *core* invertono la polarità viene generato l'impulso Wiegand che viene poi rilevato da un trasduttore magnetoelettrico (il lettore). Poiché la produzione del filo Wiegand è molto complessa, risulta pressoché impossibile duplicare le tessere Wiegand, tanto che questa tecnologia continua ad essere una delle tecnologie di controllo accessi attualmente più sicure.
Per maggiori dettagli visitare
`http://www.hidcorp.com/italiano/products/wiegand/`.

Capitolo 3

BACnetManager: Il TechManager di BACnet

Questo capitolo discute le parti più salienti del lavoro svolto per implementare il codice necessario a interfacciare l'ambiente di BACnet con la piattaforma DomoNet. Nella prima sezione verranno discussi gli obiettivi e lo stato di partenza dell'ambiente. Le successive sezioni punteranno a commentare la struttura generale del codice cercando di porre in evidenza quelle che sono le classi principali.
Per maggiori dettagli sarà necessario accedere direttamente al codice.

1 Obiettivi

Considerato che DomoNet è una piattaforma estensibile dal punto di vista dei *middleware* gestiti, aumentare numericamente quest'ultimi, garantisce certamente l'aumento di visibilità e d'importanza dell'intero pacchetto, perché di fatto si aumenta il grado di interoperabilità.
Un primo obiettivo sicuramente è quello di poter inserire un ulteriore *middleware* dentro DomoNet.
Un secondo obiettivo da valutare è quello di poter accedere al *middleware* attraverso dei meccanismi "*Open Standard*" vale a dire dei meccanismi che:

- sono **Open**, ovvero che fanno uso di tecnologie accessibili per chiunque;
- sono **Standard**, ovvero tecnologie che sono approvate da comitati ufficiali, ai quali possono aderire tutti gli interessati;

- interponendosi tra il *middleware* e il generico "Consumatore" , forniscono una visione più astratta del sistema;
- fanno di DomoNet una "generica istanza di Consumatore" che usufruisce di tali meccanismi che, per definizione, sono **usabili da chiunque**.

Questo comporta i seguenti vantaggi:

- la comunicazione effettuata attraverso uno standard aperto a tutti è un vantaggio enorme per qualunque tipo di applicazione a partire dal caso più semplice da immaginare, come il *B2B*[1], fino ad arrivare alla più personalizzata delle applicazioni;
- qualunque modifica sottostante (se tale meccanismo è sufficientemente corretto) come per esempio gli aggiornamenti, non avrà effetti nè su DomoNet nè su un qualunque altro *Consumer*, perché dovrà essere il meccanismo (solitamente gestito dal fornitore della tecnologia) ad essere aggiornato e a garantire un'interfaccia di servizi che non subisce cambiamenti (questo tipo di approccio effettuato sugli aggiornamenti è noto in Informatica come "*Compatibilità all'indietro*") .

Rimane da valutare inoltre il livello di fattibilità e di efficacia di un tale meccanismo di astrazione, ovvero se questo meccanismo interposto tra DomoNet e nel nostro caso BACnet ha un impatto sulle prestazioni oppure se le interazioni, tra i dispositivi BACnet e il resto del mondo di DomoNet, vengono adeguatamente gestite, in termini di tempo.

Il meccanismo di comunicazione che realizza le ipotesi sin ora elencate è quello dei **Web Services**.
Per capire meglio l'approccio con BACnet è possibile osservare la figura 3.1.

Possiamo notare come DomoNet abbia un "accesso diretto" al bus di campo delle altre tecnologie, vale a dire che l'approccio seguito per acquisire informazioni sui dispositivi connessi, in generale, è quello di mettersi in ascolto **direttamente** sul bus di campo e comunicando con i dispositivi secondo le specifiche dello della particolare tecnologia.

[1] *B2B* è un acronimo che sta per *Business to Business*, è un termine comunemente utilizzato per descrivere le transazioni commerciali elettroniche tra imprese, in opposizione a quelle che intercorrono tra le imprese ed altri gruppi, come quelle tra una ditta e i consumatori/clienti individuali (*B2C*, dall'inglese *Business to Customer*)

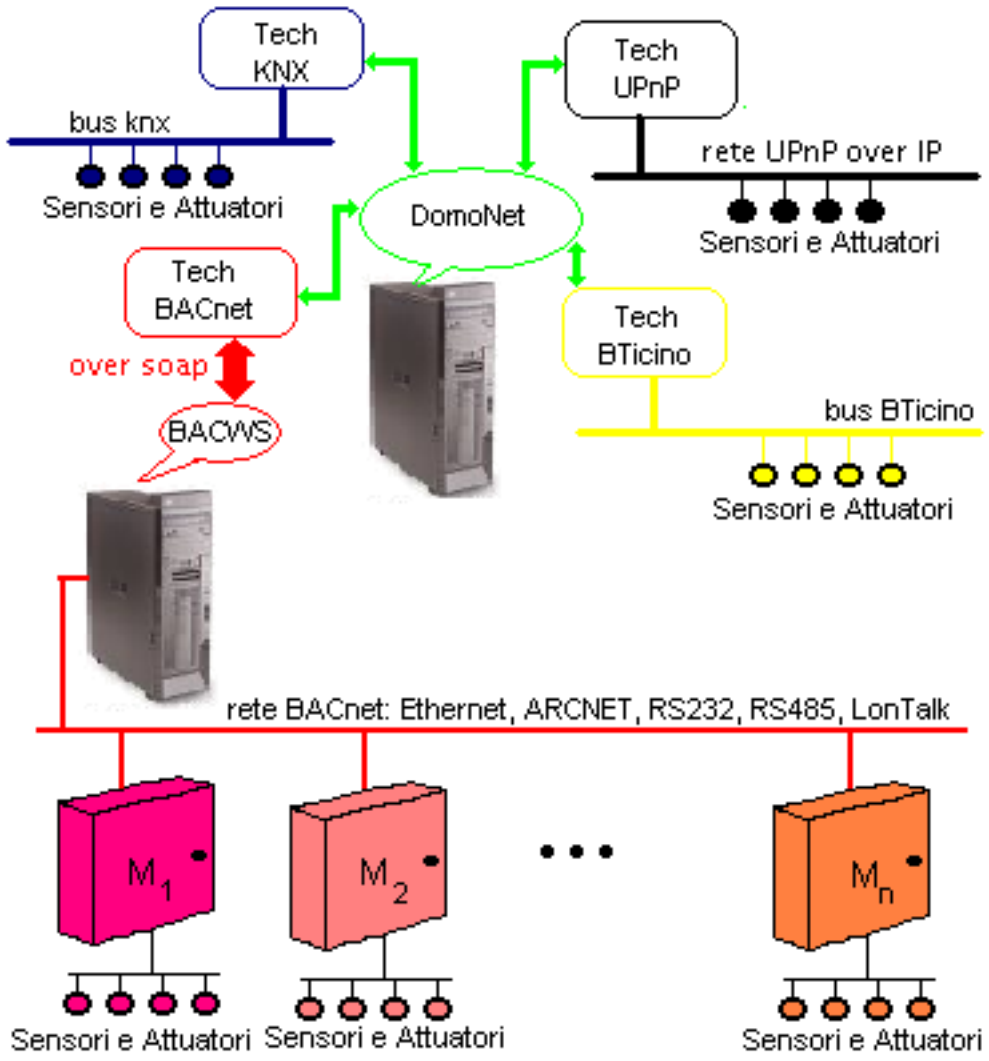

Figura 3.1: Differenza di approccio del TechManager di BACnet con gli altri TechManager di DomoNet.

Questo approccio fa diventare il TechManager un vero e proprio dispositivo fittizio conforme alla particolare tecnologia, in grado di interagire con gli altri dispositivi, per cui risulta possibile leggere e scrivere messaggi sul bus, ma allo stesso tempo occorre:

- conoscere tutti i dettagli tecnologici della tecnologia in questione;
- un eventuale aggiornamento della tecnologia, investe in pieno le classi che si preoccupano di comunicare con il *bus di campo* e da ciò ne consegue l'obbligo di dover aggiornare il TechManager relativo.

Sempre nella figura 3.1 viene evidenziato (con una suddivisione logica) il fatto che BACnet permette la comunicazione di dispositivi fisicamente appartenenti a sottoreti distinte (come discusso al capitolo 2 paragrafo 2.3) dove i box $M_1 \dots M_n$ corrispondono a dei router.

2 Quadro generale

2.1 Semantica di un path

Ricordando la definizione presente nel capitolo 2 paragrafo 3.4, si osservi la seguente istanza significativa di *path*:

```
/1/920400/181/920400/1050:Value
```

Questo path rappresenta il caso più frequente di richiesta che sarà effettuata al web Service di BACnet; il suo significato è il seguente (da sinistra verso destra):

- / indica la *Root* ovvero il nodo radice.
- **1** indica che stiamo cercando la sottorete BACnet identificata con id=1. La necessità di identificare anche la sottorete ha dei risvolti concreti in BACnet in quanto si è visto che lo standard è in grado di inglobare fino a 5 tecnologie distinte (cap. 2 par. 2.3) che possono essere riflesse anche nella visione logica del sistema come delle sottoreti distinte identificate con degli *id*.
- **920400** indica che stiamo cercando all'interno della rete con id=1 il dispositivo avente deviceID=920400.
- **181** poiché un dispositivo viene visto come un contenitore di oggetti e gli oggetti hanno un tipo, 181 sta a indicare il tipo dell'oggetto contenuto all'interno del dispostivo.
- **920400** (quello successivo a 181 cominciando da sinistra) indica l'identificatore di istanza dell'oggetto avente tipo 181.
- **1050** indica che si sta cercando la Property con id=1050 dell'oggetto sopra discusso.
- **:Value** indica che si vuole conoscere l'**attributo** (dato che siamo in presenza del token ":" (colon)) *Value* della property 1050 ovvero il valore che attualmente ha tale property.

2.2 L'albero di sintassi astratta

L'albero di sintassi astratta, rifacendosi al pattern di design *abstract Factory* [16] definisce una serie di *classi astratte* necessarie alla costruzione di una struttura dati basata sulla semantica dei *path* (vedi pragrafo precedente) atta a ospitare tutte le entità di BACnet.
Vengono inoltre definite le *classi concrete* che, estendendo le classi astratte, hanno lo scopo di ospitare i dati, cercando di mantenere una mappatura trasparente con le entità di BACnet reperibili attraverso un generico *path*.
Per **Ereditarietà** le classi concrete riusano quanto messo a disposizione dalle classi astratte, limitando la loro implementazione alla definizione dei metodi astratti e a gestire eventuali rappresentazioni interne.
Le principali classi astratte sono:

- **abstractNode.java**
- **Attribute.java**
- **Node.java**

Le principali classi concrete sono invece:

- **Root.java**
- **Network.java**
- **Device.java**
- **ObjectInstance.java**
- **SenActObjectInstance.java**
- **Attr_Impl.java**

La figura 3.2 mostra il diagramma delle classi mettendo in evidenza l'**ereditarietà**.
Guardando la figure 3.2 si capisce che tutte le classi che rappresentano dei nodi estendono la classe astratta *Node*. Chi invece estende concretamente la classe *Attribute* è *Attr_Impl* che usa tutta la rappresentazione fornita dalla classe astratta e mette a disposizione solo dei costruttori.
Un discorso a parte va fatto per *SenActObjectInstance* che gestisce quegli Oggetti di BACnet che hanno il ruolo di **Sensori** e di **Attuatori** e che quindi estende la classe *ObjectInstance*.
Nei paragrafi successivi verranno illusrate le principali classi nel dettaglio.

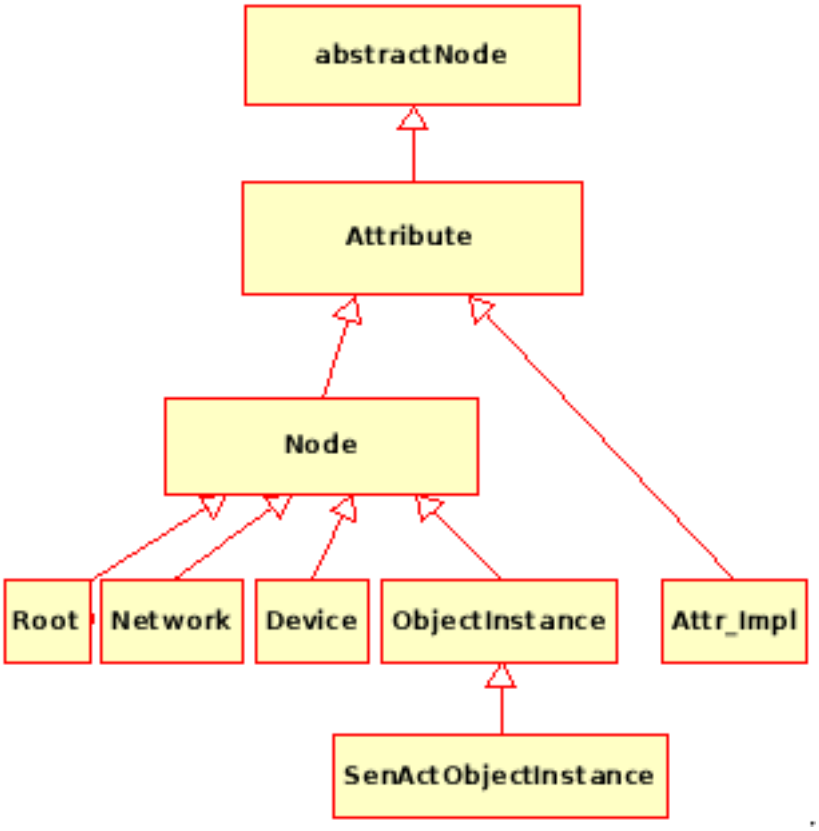

Figura 3.2: Ereditarietà tra le classi

2.2.1 abstractNode.java

La classe *abstractNode* fornisce un primo livello di astrazione con lo scopo di raccogliere tutti i membri comuni delle 2 entità principali che sono *nodi* e *attributi*[2].

Il diagramma di classe mostrato in figura 3.3 mostra la rappresentazione (ovvero costanti e variabili, statiche e d'istanza della classe) e i metodi di questa classe.

[2]Entrambe le entità si riferiscono a quanto presentato nel capitolo 2 paragrafi 3.2 e 3.3 rispettivamente.

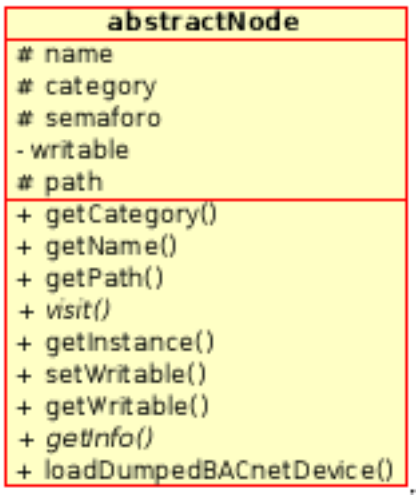

Figura 3.3: AbstractNode

Richiamo sulla notazione UML:

Per quanto riguarda la visibilità sia per la rappresentazione che per i metodi:

- – indica che il membro è **private**.
- # indica che il membro è **protected**.
- + indica che il membro è **public**.

Per quanto riguarda i metodi:

- I metodi **statici** sono sottolineati.
- I metodi **astratti** sono scritti in *corsivo*.
- I metodi d'**istanza** sono scritti normalmente.

Attraverso i metodi statici è possibile generare un'istanza di tipo *abstractNode*. La fonte dei dati risulta decisiva sulla scelta di quale metodo usare per generare un'istanza.
Usando *getInstance* vuol dire che i dati provengono dal webService ovvero, partendo dalla conoscenza minima del *path* e opzionalmente anche del *nome* del nodo, verrà creato un particolare Nodo funzionalmente al *path* passato per argomento.

Usando *loadDumpedBACnetDevice* vuol dire che la sorgente dei dati è un file xml salvato in precedenza. Il contenuto del file xml si ottiene richiamando il metodo *getInfo*. A regime sia *getInfo* che *loadDumpedBACnetDevice* non verranno mai usati, in quanto il loro utilizzo risulta circoscritto alle sole fasi di test per verificare la coerenza dell'ambiente e per effettuare delle simulazioni in assenza di connessione con il webService.
Il metodo *visit* viene richiamato sul nodo per popolarlo di tutti i dati che inizialmente erano sul webService e che a fine invocazione saranno anche nella copia locale in memoria organizzate come una struttura dati ad albero costituito in generale da *abstractNode*.
Poiché alcuni nodi sono modificabili anche dall'esterno, esiste una variabile privata *Writable* utilizzata a tale scopo e i relativi metodi di accesso *get* e *set*, di cui quest'ultimo (per coerenza) va richiamato solo all'interno di *visit* in quanto questa variabile dice solo se il nodo è/non è modificabile.
Un'ulteriore caratteristica di ogni *abstractNode* è data dall'appartenenza a una tra le seguenti categorie:

- NETWORK
- PROPRIETARY_NODE
- ROOT
- DEVICE
- ATTRIBUTE
- OBJECT_ID
- PROPERTY
- PROPRIETARY_ATTRIBUTE
- SENACTOBJECT_ID

La categoria di un nodo verrà sempre inizializzata dentro il costruttore delle classe concreta, che sarà richiamato opportunamente in funzione del *path* passato come parametro al metodo *getInstance*. Attraverso la categoria è possibile fare dei casting espliciti senza che ciò provochi degli errori a *runtime*.

2.2.2 Attribute.java

La classe *Attribute* modella l'entità *Attributo* discussa al cap. 2 par. 3.3. Una caratteristica sempre valida di un attributo è il fatto di avere associato un valore al nome che lo identifica.

Per questa ragione, come mostrato anche nella figura 3.4, un oggetto Attribute viene definito come un *abstractNode* che ha nella sua rappresentazione un membro di tipo stringa denominato *value*.
Altro aspetto da notare è il fatto che un attributo a sua volta può avere altri attributi che lo qualificano. Per questa ragione viene aggiunta una struttura dati con accesso per chiave per gli eventuali ulteriori attributi.
La struttura dati è stata scelta opportunamente con accesso per chiave in quanto non ha alcuna rilevanza l'ordine con cui vengono inseriti gli attributi figli e in fase di ricerca di un attributo figlio, questa struttura dati ci garantisce un tempo di accesso costante, quindi non lineare con il numero di attributi figli.
Per quanto riguarda i metodi della classe, sono stati definiti ulteriori metodi

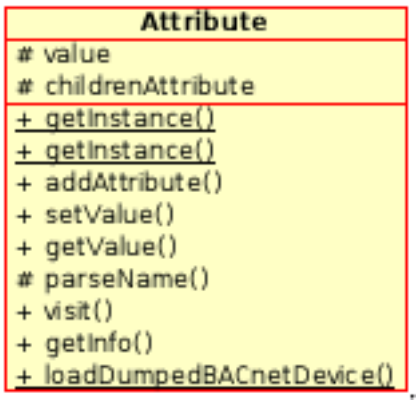

Figura 3.4: Attribute

per gestire i membri aggiuntivi alla rappresentazione che nel caso specifico sono *addAttribute*, *getValue* e *setValue*.
Inoltre viene data una definizione generale di *visit*, e di *getInfo* sugli attributi. Questa scelta evita la definizione, caso per caso di questi metodi, anche se attualmente, come mostrato nella figura 3.2, l'unica classe concreta è *Attr_Impl*.
Da notare che questa classe non espone alcun metodo astratto pur essendo dichiarata *abstract*. Lo scopo di questa scelta è quello di definire una classe con caratteristiche generali obbligando il programmatore alla definizione di classi concrete per gestire le eventuali differenze di dettaglio.

2.2.3 Node.java

La classe astratta *Node* viene usata per definire le caratteristiche generali di un nodo del Bacnet Web Service (cap. 2 par. 3.2).
Poiché un nodo può avere sia una lista di nodi figli che una lista di attributi, la classe*Node* per prima cosa estende la classe *Attribute* ereditando rappresentazione e metodi per la gestione degli attributi.
Per quanto riguarda la gestione dei nodi figli deve aggiungere una struttura dati con accesso per chiave, in quanto non è necessario mantenere l'ordine di caricamento dei dati.

Nella classe *Node*, il popolamento di dati della generica istanza, oltre che

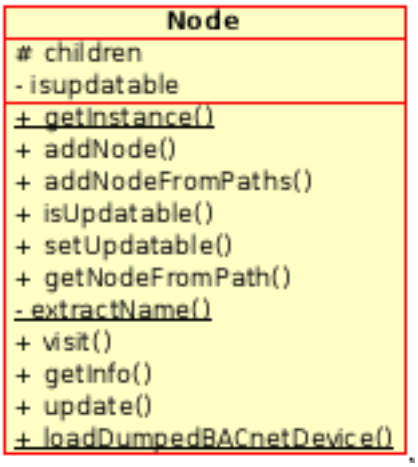

Figura 3.5: Node

con l'approccio *visit* avviene anche con l'approccio *update*.
La differenza tra i 2 approcci è la seguente:

- con *visit* avviene una visita totale del nodo, e ciò si traduce in una grossa mole di richieste che andranno fatte al BACnet webService per chiedere tutti i dettagli che il nodo possiede. Solitamente si userà *visit* nella fase di **Discovery**[3];
- con *update* si richiedono solamente quei dati che sono variabili nel tempo.

[3] Discovery significa scoperta, è la fase in cui il generico algoritmo di routing rintraccia e riconosce i dispositivi presenti all'interno di una rete. Alla fine di questa fase, in presenza di un nuovo dispositivo, l'algoritmo informa l'eventuale cambio di stato della rete dovuto al fatto che esiste nella rete un nuovo dispositivo.

Lo standard in tal proposito non fornisce nessun tipo di specifica, per cui potenzialmente qualunque dato potrebbe cambiare.
Nella realtà ci sono delle *properties* (cap. 2 par. 1.3) che, per la logica dell'applicazione non cambieranno mai con il dispositivo in stato di *running* (Si pensi per esempio la property contenente la versione del firmware o la descrizione del dispositivo).
Queste *properties* non saranno mai richieste nell'*update* al contrario di quanto avviene in *visit*.

Come la classe *Attribute*, la classe *Node* viene dichiarata *abstract* anche se non viene esposto alcun metodo astratto. Questa scelta è obbligata per le stesse ragioni esposte per la classe *Attribute*.
Come mostra la figura 3.5 anche questa classe fornisce l'implementazione dei metodi dichiarati astratti nella classe *abstractNode* perché nella maggior parte dei casi (dei nodi concreti) tale implementazione risulta essere coincidente.
Delle classi concrete che estendono *Node* verrà analizzata solo la classe *Device* in quanto risulta essere la più significativa.

2.2.4 Device.java : un esempio di classe concreta

La classe *Device* modella logicamente il concetto che BACnet ha di un dispositivo (cap. 2 par. 1.1).
La classe *Device* è stata scelta come esempio di classe concreta in quanto, oltre ad espletare i comportamenti definiti dallo standard BACnet, definisce alcune ottimizzazioni mirate a minimizzare il numero di comunicazioni col web service considerato il peso di un tale tipo di protocollo di comunicazione.
La figura 3.6 mostra il relativo diagramma di classe.
Fondamentalmente espleta le seguenti funzionalità:

1. recupera tutte le informazioni ritenute necessarie dell'oggetto Device di BACnet (cap. 2 par. 1.2) ed effettua un'unica richiesta (per dettagli osservare il metodo *myPropertyMan*) richiamando il metodo *getValues* del web Service di BACnet.
   Queste informazioni saranno rielaborate e usate successivamente nella conversione da Device a DomoDevice al fine di fornire informazioni

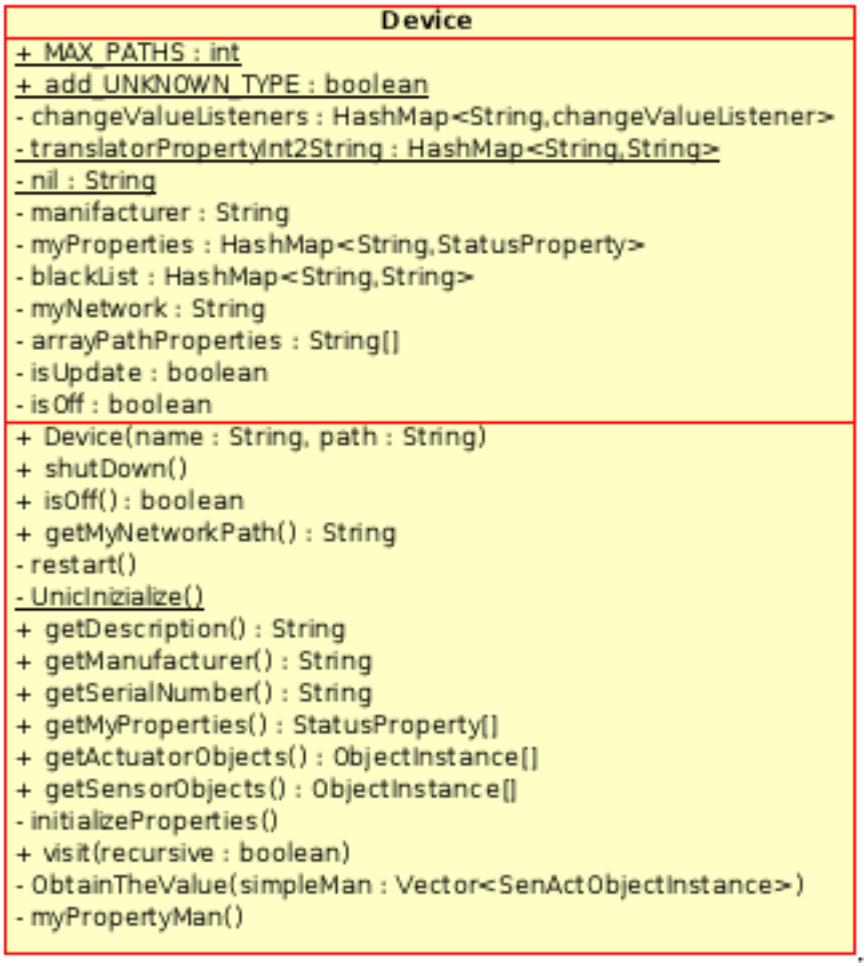

Figura 3.6: La classe concreta Device

aggiuntive sul dispositivo. I metodi *getSerialNumber*, *getManufacturer* e *getDescription* vengono usati per offrire all'esterno tali informazioni aggiuntive;

2. considerato che in BACnet un dispositivo può assumere i comportamenti classici di un dispositivo mobile (può entrare e uscire dalla rete in un qualunque momento) è stato modellato uno stato di **acceso/spento** basato sul reperimento della property **OBJECT_IDENTIFIER** [4]

[4]La property OBJECT_IDENTIFIER è recapitata all'interno dell'oggetto Device. Il suo codice identificativo è 75, quando il dispositivo BACnet è visibile il valore di questa property mantiene la coppia (TipoOggettoDevice, DeviceID), in caso contrario contiene un messaggio di errore. I messaggi di errore sono caratterizzati dal fatto che comincianto tutti con il carattere '?'.

I metodi *shutDown* e *restart* agiscono (secondo la logica classica) sulla variabile privata *isOff* che indica tale stato. Il metodo *isOff()* informa esternamente dello stato dell'omonima variabile privata, questa informazione si rivela importante per informare DomoNet di un eventuale cambio di stato del dispositivo. Questo avrà evidenti ripercussioni sulle strutture dati di DomoNet;

3. per quanto riguarda gli oggetti *SenActObjectInstance* propone un'ulteriore ottimizzazione. Nel caso normale, avrebbe dovuto effettuare delle ottimizzazioni limitate al solo contesto dell'oggetto e questo avrebbe comportato almeno n richieste al web service per aggiornare gli n oggetti del dispositivo composte dei soli *path* del singolo oggetto. Tale ottimizzazione locale avrebbe potuto generare un meccanismo fuori controllo che in alcuni casi si sarebbe tradotto in un sottoutilizzo e in altri in un sovrautilizzo dell'uso della richiesta multipla di valori al web service (il metodo *getValues* per intendersi). Per questa ragione è stata definita una variabile statica denominata **MAX_PATHS**[5] con lo scopo di agglomerare fino a MAX_PATHS *path* tutti in un'unica richiesta;

4. considerato che il meccanismo con cui avviene la gestione dell'aggiornamento dei dati è il Polling[6] al web Service, ad ogni ciclo di aggiornamento sarebbe stato necessario aggiornare anche DomoNet. Al fine di rompere la catena di richieste, la classe *Device* instaura un **meccanismo a eventi** [17] con il Thread *AbstractSintaxTreeManager* in cui, soltanto quando avviene un cambiamento di valore sul sensore o sull'attuatore, viene fatto partire un evento dalla classe Device che verrà opportunamente gestito dal Thread;

5. riusa la rappresentazione di *Node* per contenere i suoi oggetti;

6. della lista di oggetti proposti nella tabella al cap. 2 par. 1.2 riesce a trattare solo:

[5] Il valore di default di questa variabile è 50 ma sarebbe stato interessante poter valutare con maggior accuratezza il limite dei *path* richiedibili in una singola richiesta.

[6] Per maggiori dettagli sul Polling visitare:
`http://it.wikipedia.org/wiki/Polling_(informatica)`.

- ANALOG_{INPUT | OUTPUT | VALUE} in modo ottimizzato in quanto è stato possibile effettuare diversi test con questo tipo di oggetti Standard;
- BINARY_{INPUT | OUTPUT | VALUE} in modo ottimizzato in quanto è stato possibile effettuare diversi test con questo tipo di oggetti Standard;
- MULTI_STATE_{INPUT | OUTPUT | VALUE} senza ottimizzazione perché non è stato possibile avere degli scenari con questo tipo di oggetti.

Gli altri tipi di oggetto Standard di BACnet e quelli non Standard allo stato attuale vengono ignorati in quanto, escluso il monitoraggio, non é possibile instaurare altro tipo di comunicazione attraverso il BACnet Web Service.
Questa scelta non pregiudica eventuali esigenze future di aggiornamento, in quanto la definizione astratta dei dati consente eventuali cambi di gestione in proposito, senza che ciò provochi enormi trasformazioni.

2.3 La Struttura Dati

Le classi concrete definite nell'albero di sintassi astratta, insieme ad altre classi di supporto, hanno lo scopo di definire una struttura dati ad albero, che ripercorre la semantica dei *Path* (par. 2.1).
Seguendo tale modello si ottiene la struttura dati mostrata in figura 3.7.

Nella figura si può notare il confronto tra l'informazione mantenuta sul *Path*[7] e il corrispondente *mapping* nella struttura dati.
Osservando le frecce blu si può notare come sia il **tipo** che l'**ID** di un oggetto vengono mantenuti in un solo nodo di tipo ObjectInstance.
Questa scelta riduce la quantità di oggetti da creare e da mantenere aggiornati, con conseguente miglioramento generale sulle prestazioni.

La struttura dati prevede anche dei meccanismi di sincronizzazione attraverso dei *semafori di lock* in quanto risulta possibile l'accesso multiplo alla struttura dati dovuta al fatto che esiste sia il thread di aggiornamento che il

[7] Il *path* è quello mostrato come esempio nel par. 2.1

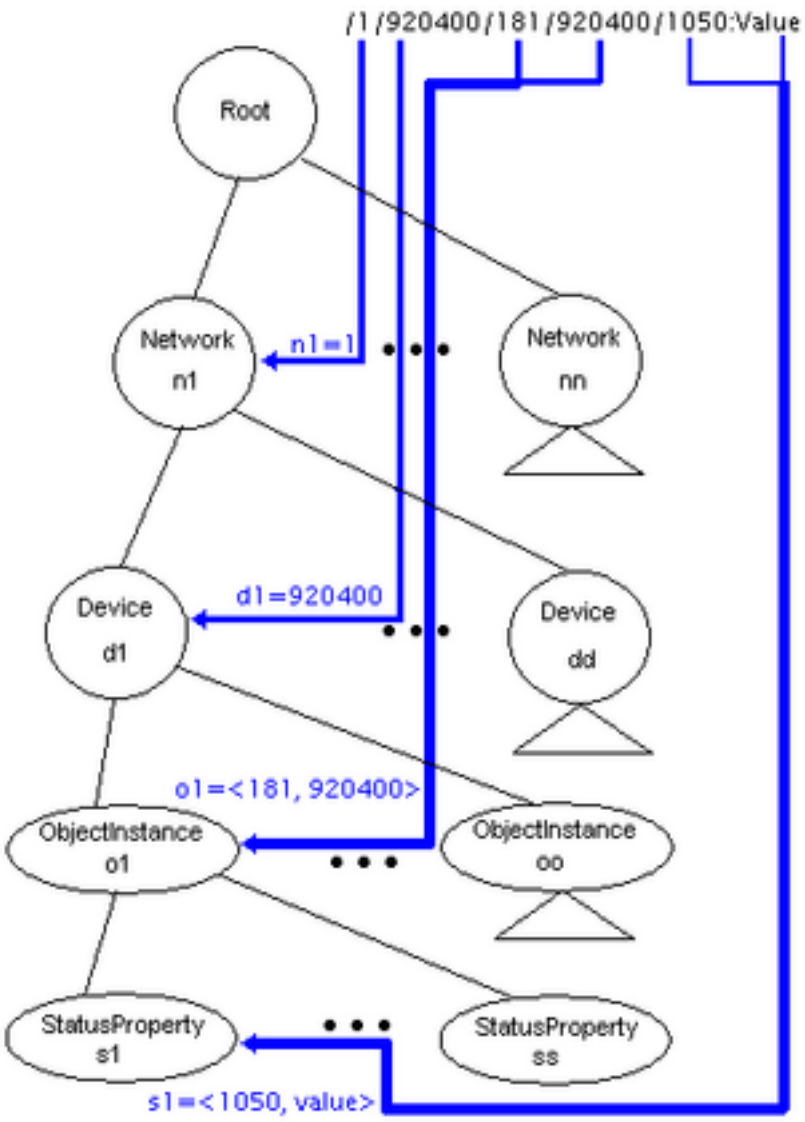

Figura 3.7: La struttura Dati in memoria: I triangolini indicano i sotto-alberi mentre i nodi dell'albero sono indicati con dei cerchi. Le linee nere indicano il legame di parentela (padre-figlio), mentre le frecce blu indicano la corrispondenza tra elemento del Path e nodo sulla struttura dati.

BACnetManager che possono sia leggere che modificare la struttura dati. I *semafori di lock* sono realizzati per sovvenire ai problemi classici dei **lettori-scrittori su una struttura dati condivisa** [19]. Ogni istanza ha dunque un suo semaforo che viene utilizzato per garantire la consistenza dei dati.

2.4 AbstractSintaxTreeManager

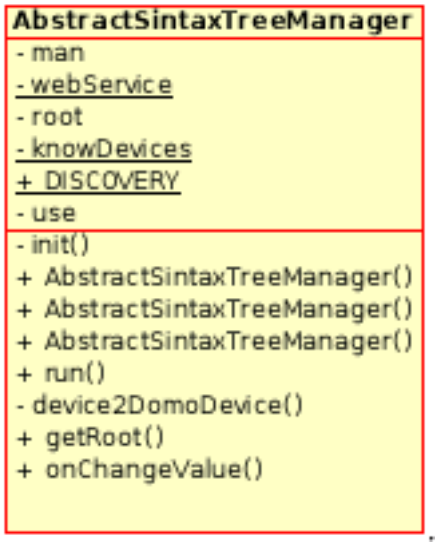

Figura 3.8: Diagramma della classe *AbstractSintaxTreeManager*.

L'*AbstractSintaxTreeManager* è un *Thread* richiamato all'interno del *BACnetManager* e viene utilizzato per adempiere i seguenti scopi:

1. creare la struttura dati (discussa nel paragrafo precedente);
2. mantenere aggiornata la struttura dati, mantenendo inoltre la coerenza tra *BACnetManager* e i dati aggiornati sulla struttura dati;
3. trasformare i dati mantenuti nella struttura dati in dati usabili dal *BACnetManager*.

L'ordine con cui sono stati elencati costituisce in qualche modo la sequenza temporale delle azioni perseguite dall'*AbstractSintaxTreeManager*.
La creazione della struttura dati (il punto 1) si realizza in 2 fasi:

1. attraverso il costruttore avviene l'inizializzazione della *URL* del web-Service e della modalità d'uso impostata su **DISCOVERY** [8]:

   ```
   public AbstractSintaxTreeManager(BacnetManager man,
   String webServiceAddress, int use)
   ```

   Ricordando inoltre che tale inzializzazione avviene all'interno del *BACnetManager*, il primo argomento del costruttore sarà sempre *this*;

[8] DISCOVERY è una costante pubblica definita all'interno di questa classe.

2. nella fase di *Running* viene inizializzato un oggetto di tipo *Root* e su questo viene richiamato il metodo *visit*.

L'aggiornamento della struttura dati (lo scopo numero 2 del AbstractSintaxTreeManager), viene realizzato in 2 fasi:

1. attraverso il costruttore avviene l'inizializzazione del nodo *Root* ottenuto dal *AbstractSintaxTreeManager di Discovery*:

   ```
   public AbstractSintaxTreeManager(BacnetManager man, Root root)
   ```

2. nella fase di *running* (che realizza in un ciclo infinito), al nodo *root* viene chiesto di effettuare a intervalli regolari l' *update*. A fine *update* l'*AbstractSintaxTreeManager* controlla la presenza di eventuali cambiamenti (per esempio se ci sono nuovi dispositivi) e in tal caso li comunica al *BACnetManager*.

La terza funzionalità dell'AbstractSintaxTreeManager nella realtà viene effettuata sia all'interno del punto 1 che del punto 2.
La "Trasformazione dei dati" avviene attraverso 2 azioni distinte usate per adempiere 2 compiti differenti.

- la trasformazione di un *Device* in *DomoDevice* attraverso il metodo:
  `private void device2DomoDevice(Device dev)`

  Questo metodo è *void* in quanto è lui stesso che, a fine trasformazione, si incarica di comunicare al *BACnetManager* di aggiungere il nuovo *DomoDevice*.
- l'aggiornamento delle *properties* di un oggetto attraverso il meccanismo a eventi fornito dalla classe Device.

  Questo meccanismo viene gestito con:
  - la registrazione/deregistrazione di *AbstractSintaxTreeManager* alle istanze di Device (par. 2.2.4 punto 4) richiamando rispettivamente i metodi d'istanza:
    `addChangeValueListener(this);`
    `removeChangeValueListeners(this);`
    di device;
  - l'implementazione del metodo [9]:
    `public void onChangeValue(changeValueEvent e)`

[9] in quanto AbstractSintaxTreeManager implementa l'interfaccia *changeValueListener*

da parte di *AbstractSintaxTreeManager*. In questo metodo viene definito "cosa fare" qualora ci sia un evento come il cambio di valore di una property. Viene richiamato da *Device* in presenza di tali eventi.

2.5 BACnetManager

La classe *BACnetManager* estende la classe astratta *TechManager* che dichiara astratti i seguenti metodi:

```
public abstract void addDevice(final DomoDevice domoDevice,
String address);
public abstract DomoMessage execute(final DomoMessage domoMessage);
public abstract void finalize();
public abstract void loadDumpedDomoDevice(final DomoDevice domoDevice);
public abstract void start();
```

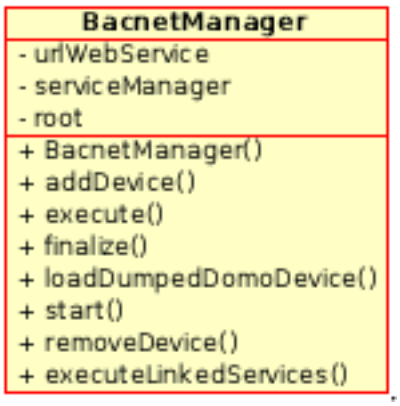

Figura 3.9: BACnetManager il techManager di BACnet

e dei quali è necessario fornire l'implementazione.
Per quanto riguarda il significato di questi metodi si rimanda alla tesi [2].
Invece, per quanto riguarda l'implementazione, le parti da evidenziare sono le seguenti:

- nel metodo *start* il BACnetManager prima dichiara (e fa partire) l'*AbstractSintaxTreeManager* di Discovery per conoscere lo stato della Rete BACnet, successivamente crea una seconda istanza (e fa partire pure questa) che verrà usata per mantenere aggiornata la struttura dati creata attraverso il Discovery;

- nel metodo *execute*, usato per acquisire o modificare lo stato di una *property* all'interno di un *ObjectInstance*, il *BAcnetManager* grazie al supporto delle classi dell'*AbstractSintaxTree* è in grado di:
  - recuperare dalla struttura dati la property richiesta;
  - effettuare l'operazione richiesta dal DomoMessage (Lettura o Scrittura).

Ai metodi suddetti, come mostrato nella figura 3.9 vengono aggiunti i seguenti:

```
public void removeDevice(final String address)
```

in quanto in BACnet i dispositivi possono essere inseriti e disinseriti nella rete in un qualunque momento e quindi va gestito tale cambio di stato.

```
public final void executeLinkedServices(ObjectInstance obi)
```

utilizzato per aggiornare eventuali *LinkedServices* (Cap. 1, par. 1.1.3), collegati allo stato dell'oggetto *obi*. Attraverso il meccanismo a eventi discusso in precedenza, questo metodo viene richiamato dall'*AbstractSintaxTreeManager* solamente quando lo stato di tale oggetto è veramente cambiato (e questo non è detto che avvenga ad ogni richiesta di *update*).

3 Test

In questa sezione vengono mostrati i test effettuati per verificare l'effettivo funzionamento del *BACnetManager*. Gli ultimi test prenderanno in esame il funzionamento dei *LinkedService* costruiti per mettere in relazione un dispositivo BACnet con un dispositivo di un'altra tecnologia già presente. A titolo di esempio si illustreranno i test di cooperazione con la tecnologia UPnP.

3.1 Preparazione dell'ambiente di test

Per poter realizzare i test, che saranno elencati di seguito, risulta necessaria la seguente configurazione minimale:
HARDWARE:
3 macchine collegate in rete sul quale mettere in esecuzione i seguenti processi:

- Macchina <A>: con sistema operativo della famiglia Windows per poter lanciare il processo *SCADA Bacnet device simulator* (cap. 2 par. 4.1.2).
- Macchina <B>: con sistema operativo della famiglia Windows per poter lanciare il processo *SCADA Bacnet Web Service* (cap. 2 par. 4.1.1). La macchina A e la macchina B devono necessariamente far parte di un'unica rete al fine di realizzare una rete *BACnet* minimale composta dal web Service e da un dispositivo emulato.
- Macchina <C>: con sistema operativo Linux (anche se il sistema operativo in questo caso non costituisce un vincolo) su cui mandare in esecuzione *DomoNet*. Inoltre questa macchina sarà usata per mettere in esecuzione ulteriori processi ausiliari.

SOFTWARE:

1. <C> Si suppone che la directory radice di *DomoNet* sia mantenuta nella variabile d'ambiente

   ```
   ${DomoRoot}
   ```

2. <C> Verificare che nel costruttore del *DomoNetWS*[10] siano caricati come lista minimale i *techManager* di *BACnet* e di *UPnP*. Se sono commentati, scommentarli e salvare tutto.
3. <C> Verificare che nel file

   ```
   ${DomoRoot}/domonet/trunk/src/domoNetWS/domoNetWS.properties
   ```

   la properties *URLBacnetWebService* abbia una *URL* coerente con l'indirizzo IP della macchina <B>.
4. <C> aprire una shell *S1* e posizionarsi nella directory

   ```
   ${DomoRoot}/tools/apache-tomcat-6.0.10
   ```

   per poter visionare i logs di *DomoNet* generati dentro l'ambiente di Tomcat.

[10] La classe DomoNetWS.java si trova in
${DomoRoot}/domonet/trunk/src/domoNetWS

5. <C> aprire 2 shell distinte *S2* e *S3* e posizionarsi in entrambi nella directory

   ```
   ${DomoRoot}/domonet/trunk
   ```

   Queste 2 shell saranno usate per lanciare il *DomoNet* Client (Cap. 1 par. 1.3) e il *DomoNet* Server (Cap. 1 par. 1.2).
6. <C> aprire una shell *S5* e spostarsi nella directory

   ```
   ${DomoRoot}/tools/clink/upnp-sample-aircon
   ```

   Questa shell verrà usata nel test 5 per lanciare il processo del condizionatore d'aria *UPnP*.
   Verificare che i path delle librerie presenti nel file *loadUPnP.sh* siano coerenti con quelli dell'ambiente globale.
7. <C> aprire una shell *S6* e spostarsi nella directory

   ```
   ${DomoRoot}/tools/clink/upnp-sample-light
   ```

   Questa shell verrà usata nel test 6 per lanciare il processo della lampadina *UPnP*.
   Verificare che i path delle librerie presenti nel file *loadUPnP.sh* siano coerenti con quelli dell'ambiente globale.

3.2 Test 1: BACnet Web Service irraggiungibile

AZIONI:

1. dalla shell *S2* di <C> dare il comando[11]

   ```
   ant startServer
   ```

   che compila tutte le classi necessarie a far partire il *DomoNetWS* e fa partire Tomcat.
2. dalla shell *S4* di <C> dare il comando

   ```
   tail -f logs/catalina.out
   ```

[11] Si suppone che nell'ambiente sia correttamente configurato ANT [24].

che mostra i logs di Tomcat e sul quale si riverseranno tutti i commenti del *DomoNetWS*.

COMMENTI:
Questo test ha lo scopo di valutare il comportamento del *BACnetManager* qualora il *BACnet Web Service* risulti irraggiungibile.
I risultati visualizzati nella tabella 3.1 riguardano il *logs* di *DomoNet*.
Finché non viene trovato il *BACnet Web Service* il *BACnetManager* segnala sempre tale problema attraverso gli ultimi 2 messaggi della tabella 3.1.

Il risultato dimostra il livello di disaccoppiamento tra l'entità gestore (*BACnetManager*) e l'entità gestita (il *BACnet Web Service*), in quanto anche se il BACnet Web Service risulta irraggiungibile, il TechManager di BACnet continua a funzionare in modo indipendente.

3.3 Test 2: Caduta del BACnet Web Service

AZIONI:

1. Identico al punto 1 del Test 1.
2. Identico al punto 2 del Test 1.
3. Sulla macchina <B> lanciare il processo *webService di BACnet*.
4. Sulla macchina <A> lanciare il processo *BACnet Device Simulator* e nella maschera mostrata nella figura 3.10 inserire i seguenti parametri:.
   - Frequency(secs) = 3 (non fondamentale)
   - DeviceID = 40
   - IP Network Number = 1
   - Objects per Object Type = 5
5. Sulla macchina <B> dopo qualche minuto di esecuzione del *webService di BACnet* (verificato sul logs). provocare la caduta forzata del servizio chiudendo l'applicazione.

COMMENTI:
Lo stato di partenza di questo test prevede che inizialmente la comunicazione con il *BACnet Web Service* sia avvenuta e sono state caricate nella struttura dati[12] del *BACnetManager* le informazioni relative allo stato della rete

[12]Vedere il par. 2.3 per maggiori dettagli

```
OS = Linux
***********************************************
** DomoNet                                   **
**                                           **
** Copyright(C) 2006 Dario Russo             **
** DomoNet comes with ABSOLUTELY NO WARRANTY. **
** This is free software, and you are welcome **
** to redistribuite it under certain         **
** conditions. For details see COPYING file.  **
***********************************************
[2008-04-17 10:45:48]  Loading class domoNetWS.techManager.upnpManager.UPNPManager
using default ssdp and http port...
done.
[2008-04-17 10:45:48] <BacnetManager> Make the Discovery AbstractSintaxTreeManager
that use the web Service's URL "http://192.168.1.201:8080/"
[2008-04-17 10:45:48] Starting  class domoNetWS.techManager.upnpManager.UPNPManager
using localhost:0...
CyberGarage message : accept ...
CyberGarage message : accept ...
done.
[2008-04-17 10:45:48] <BacnetManager>Starting  the Discovery
<BACnet[Node-visit]> Connection problem with the webService!
[2008-04-17 10:45:48] <BACnet[AbstractSintaxTreeManager]> The Discovery fase is:
128 that is't same at: 0 min  and 0 secs
[2008-04-17 10:45:48] <BacnetManager> Discovery done.
[2008-04-17 10:45:48] <BacnetManager> make and start the Update AbstractSintaxTreeManager
[2008-04-17 10:45:48] dumpSockets.xml can't be readed. Check if file exists or
the permission flags.
[2008-04-17 10:45:48] Initialization of the web service terminated.
[2008-04-17 10:45:48] Give the list of devices: <devices></devices>
[2008-04-17 10:45:48] <BACnet[Root-update]> Connection problem with the webService!
[2008-04-17 10:45:48] <BACnet[AbstractSintaxTreeManager]> The update fase is:
94 that is't same at: 0 min  and 0 secs
[2008-04-17 10:45:53] <BACnet[Root-update]> Connection problem with the webService!
...
```

Tabella 3.1: Log di DomoNet con web Service irraggiungibile.

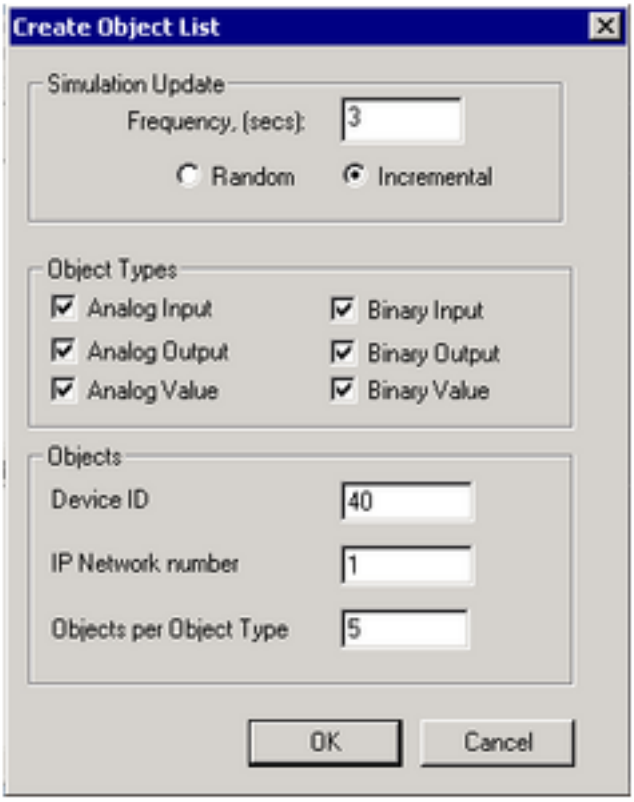

Figura 3.10: Maschera di configurazione del *BACnet Device Simulator* accessibile dal menu *Server* della maschera principale.

BACnet con alcuni dispositivi presenti.
A un certo istante, per una qualunque ragione il *BACnet Web Service* non è più raggiungibile.
Nella tabella 3.2 viene mostrato il *logs* di *DomoNet* in cui la presenza dei "..." indica che sono state tagliate alcune parti. Il comportamento mostrato è molto chiaro e può essere così riassunto:

1. Alle 11:12:57 il *BACnetManager* non ha ancora rilevato nessun *BACnet Web Service*.
2. Alle 11:13:03 la fase di *Update* risulta molto più lunga della precendente (626, 4) che indica che qualcosa è avvenuto; il tutto si chiarisce alla 11:13:08 ovvero il momento in cui viene segnalato l'ingresso del *BACnet Web Service*.
3. Alle 11:14:40 abbiamo un altro *Update* molto lungo (5895 millisecondi) che presagisce l'arrivo di un nuovo dispositivo che avviene alle 11:14:53.

```
...
[2008-04-17 11:12:57] <BACnet[Root-update]> Connection problem with the webService!
[2008-04-17 11:12:57] <BACnet[AbstractSintaxTreeManager]> The update fase is:
4 that is't same at: 0 min  and 0 secs
[2008-04-17 11:13:03] <BACnet[AbstractSintaxTreeManager]> The update fase is:
626 that is't same at: 0 min  and 0 secs
[2008-04-17 11:13:08] Adding dumped domoDevice:
<device description="SCADA Engine BACnet Server" id="0"
manufacturer="SCADA Engine [123]" position="" positionDescription="ESAC sede di via
Manzoni n 1, Collegno (TO)" serialNumber="/1/4" tech="BACNET" type="" url="" />
[2008-04-17 11:13:08] Sending update message: <device description="SCADA Engine
BACnet Server" id="0" manufacturer="SCADA Engine [123]" position=""
positionDescription="ESAC sede di via Manzoni n 1, Collegno (TO)"
serialNumber="/1/4" tech="BACNET" type="" url="" /> to
[2008-04-17 11:13:08] <BACnet[AbstractSintaxTreeManager]> The update fase is:
110 that is't same at: 0 min  and 0 secs
...
[2008-04-17 11:14:40] <BACnet[AbstractSintaxTreeManager]> The update fase is:
5895 that is't same at: 0 min  and 5 secs
[2008-04-17 11:14:53] Adding dumped domoDevice:
<device description="My BACnet Server" id="2" manufacturer=" [0]"
...
[2008-04-17 11:15:30] <BACnet[AbstractSintaxTreeManager]> The update fase is:
12016 that is't same at: 0 min  and 12 secs
[2008-04-17 11:15:41] Sending update message: <message message="/1/40/5/1"
messageType="UPDATE" receiverId="2" receiverURL="" senderId="2" senderURL="">
<input name="BINARY_VALUE[1]" type="BOOLEAN" value="false" /></message> to
...
[2008-04-17 11:16:01] <BACnet[AbstractSintaxTreeManager]> The update fase is:
5 that is't same at: 0 min  and 0 secs
[2008-04-17 11:16:06] <BACnet[Root-update]> Connection problem with the webService!
[2008-04-17 11:16:06] <BACnet[AbstractSintaxTreeManager]> The update fase is:
5 that is't same at: 0 min  and 0 secs
[2008-04-17 11:16:11] <BACnet[Root-setOffAllDevice]> Start to shut down all BACnet Devices!
[2008-04-17 11:16:11] <BACnet[Root-setOffAllDevice]> shutted down all BACnet Devices!
[2008-04-17 11:16:11] <BACnet[Root-update]> Connection problem with the webService!
[2008-04-17 11:16:11] Removing domoDevice: <device description="SCADA Engine BACnet Server"
id="0"  serialNumber="/1/4" tech="BACNET"
...
[2008-04-17 11:16:11] <BACnetManager> Removed the Device with address: /1/4
[2008-04-17 11:16:11] Removing domoDevice: <device description="My BACnet Server" id="2"
serialNumber="/1/40" tech="BACNET"
...
[2008-04-17 11:16:11] <BACnetManager> Removed the Device with address: /1/40
[2008-04-17 11:16:11] <BACnet[AbstractSintaxTreeManager]> The update fase is:
10 that is't same at: 0 min  and 0 secs
[2008-04-17 11:16:16] <BACnet[Root-update]> Connection problem with the webService!
...
```

Tabella 3.2: Log di DomoNet con web Service caduto.

Questo dispositivo come si nota dal logs era già presente nella lista dei dispositivi "dumped".

4. A questo punto proseguono gli aggiornamenti tanto che alle 11:15:41 avviene pure un cambio di valore su un BINARY_VALUE del dispositivo /1/40.
5. Tutto proscede serenamente finché alle 11:16:06 il *BACnet Web Service* viene forzatamente staccato (chiudendo l'applicazione del Bacnet Web Service 4.1.1) per cui diventa irraggiungibile.
6. Il comportamento del *BACnetManager* prevede che dopo 3 tentativi falliti avviene lo spegnimento logico dei dispositivi *BACnet* per cui vengono segnalati tali eventi alle 11:16:11, con lo scopo di informare *DomoNet* dell'**assenza** di tutti i dispositivi *BACnet*.

La descrizione in DomoML dei dispositivi BACnet, nell'esempio della tabella 3.2 è incompleta per la verbosità che ne avrebbe comportato.

3.4 Test 3: Ingresso di un Dispositivo nella rete BACnet

AZIONI:

1. Identico al punto 1 del Test 1.
2. Identico al punto 2 del Test 1.
3. Identico al punto 3 del Test 2.
4. Identico al punto 4 del Test 2.

COMMENTI:

Sempre nella tabella 3.2 si può visionare che alle 11:14:53 nella rete BACnet viene rilevato il nuovo dispositivo per cui viene aggiunto nella struttura dati del *BACnetManager* e contemporaneamente viene informato *DomoNet*. Quest'ultimo verificato che tale dispositivo era presente nella lista dei dispositivi "dumped", recupera la descrizione "dumpata" inibendo la creazione di un nuova descrizione con un eventuale ID.

3.5 Test 4: Uscita di un Dispositivo dalla rete BACnet

AZIONI:

1. Identico al punto 1 del Test 1.

```
...
[2008-04-17 12:46:36] Adding dumped domoDevice:
<device description="My BACnet Server" id="2" manufacturer=" [0]" position=""
positionDescription="ESAC sede di via Manzoni n 1, Collegno (TO)" serialNumber="/1/40"
tech="BACNET" type="" url=""><service description="Luce Salotto" name="/1/40/3/3"
output="BOOLEAN" outputDescription="" prettyName="BINARY_INPUT[3]" />
<service description="Luce corridoio" name="/1/40/3/2"
...
[2008-04-17 12:46:36] <BACnet[AbstractSintaxTreeManager]> The update fase is:
8124 that is't same at: 0 min  and 8 secs
[2008-04-17 12:47:15] Sending update message: <message message="/1/40/0/1"
messageType="UPDATE" receiverId="2" receiverURL="" senderId="2" senderURL="">
<input name="ANALOG_INPUT[1]" type="DOUBLE" value="1.200000" /></message> to
...
[2008-04-17 12:48:25] Removing domoDevice: <device description="My BACnet Server" id="2"
...
[2008-04-17 12:48:25] Sending update message: <message message="" messageType="REMOVE"
receiverId="2" receiverURL="" senderId="2" senderURL="" /> to
[2008-04-17 12:48:25] <BACnetManager> Removed the Device with address: /1/40
[2008-04-17 12:48:25] <BACnet[AbstractSintaxTreeManager]> The update fase is:
5803 that is't same at: 0 min  and 5 secs
[2008-04-17 12:48:32] <BACnet[AbstractSintaxTreeManager]> The update fase is:
1879 that is't same at: 0 min  and 1 secs
...
```

Tabella 3.3: Log di DomoNet che segnala l'uscita di un dispositivo.

2. Identico al punto 2 del Test 1.
3. Identico al punto 3 del Test 2.
4. Identico al punto 4 del Test 2.
5. Sulla macchina <A> dopo qualche minuto di esecuzione del *BACnet Device Simulator* (verificato sul logs della macchina <C>). provocare la caduta forzata del dispositivo emulato chiudendo l'applicazione.

COMMENTI:

Nella tabella 3.3 possiamo notare il comportamento temporale del dispositivo avente serial number /1/40.

1. Alle 12:46:36 viene aggiunto il dispositivo /1/40 (di cui viene mostrata una descrizione incompleta).
2. Alle 12:47:15 viene segnalato a *DomoNet* che un ANALOG_INPUT del dispositivo /1/40 ha un nuovo valore corrispondente a 1.200000. Questa segnalazione serve a capire cosa succede appena viene rilevato un cambio di stato rilevato su un oggetto di un dispositivo *BACnet*, mettendo in evidenza anche il livello di granularità dell'informazione trattata.
3. Alle 12:48:25 viene rilevata l'uscita del dispositivo /1/40 (ottenuta chiudendo l'applicativo Device Simulator).

Come si può notare il *BACnetManager* ha gestito completamente tutte le fasi del ciclo di vita di un dispositivo *BACnet* tenendo sempre aggiornato *DomoNet*.

3.6 Test 5 - LinkedService 1: Da Condizionatore d'aria UPnP ad ANALOG_OUTPUT di BACnet

AZIONI:

1. Nel file

   ```
   ${DomoRoot}/domonet/trunk/xml/domoDevices.xml
   ```

   è stato definito il LinkedService che collega il servizio *SetTemp* del dispositivo UPnP al servizio relativo all'Oggetto ANALOG_OUTPUT (nel nostro esempio il ANALOG_OUTPUT[2]) del dispositivo BACnet; come mostrato nella tabella 3.4.

2. Identico al punto 1 del Test 1.
3. Identico al punto 2 del Test 1.
4. Identico al punto 3 del Test 2.
5. Identico al punto 4 del Test 2.
6. Sulla macchina <C> dalla shell *S5* lanciare il comando

   ```
   loadUPnP.sh AirconFrame
   ```

   Se l'operazione va a buon fine dovrebbe comparire una maschera come quella mostrata nella figura 3.11 che simula le caratteristiche di un condizionatore d'aria.
7. Sulla macchina <C> dalla shell *S3* lanciare il comando

   ```
   ant startClient
   ```

   che manda in esecuzione l'applicazione *DomoNetClient* comprensiva di una interfaccia grafica. Su questa interfaccia va cliccato il bottone **Connect** per effettuare una connessione al *DomoNetWS*.
8. A questo punto navigando sulla directory del web Service, come mostrato nella figura 3.12, vengono visualizzati tutti i dispositivi trovati da DomoNet. Una volta trovato il il domoDevice *aircon*, occorre andare a cercare al suo interno il servizio *SetTemp*. Cliccare sull'icona di quest'ultimo per fare aprire una finestra usata per settare la temperatura.
9. Si ripetono le stesse operazioni del punto precedente stavolta andando a cercare il servizio *[GET]_ANALOG_OUTPUT[2]* (corrispondente all'oggetto BACnet ANALOG_OUTPUT[2]) dentro il domoDevice *My BACnet Server* con lo scopo di supervisionare l'effettivo cambiamento di stato del servizio.
10. Settando la temperatura a 23 gradi si ottiene il comportamento della figura 3.13 e per sottolineare la dinamicità dell'applicazione viene mostrata la figura 3.14 dove la temperatura viene settata a 27 gradi.

COMMENTI:

Attraverso i passaggi descritti sopra si dimostra come modificando lo stato di un dispositivo UPnP questo si ripercuote su un'altro dispositivo che in

```
...
<device description="CyberGarage AirCon Device" id="7" manufacturer="CyberGarage"
position="" positionDescription="" serialNumber="uuid:cybergarageAirConDevice"
tech="UPNP" type="aircon">
...
<service description="" name="SetTemp" output="BOOLEAN" outputName="Result"
prettyName="SetTemp (Result)">
<input description="" name="Temp" type="STRING" />
<linkedService  id="2" service="/1/40/1/2" url="">
<linkedInput from="Temp" to="ANALOG_OUTPUT[2]" />
</linkedService>
</service>
...
</device>
...
```

Tabella 3.4: Frammento xml per configurare il LinkedService del test 5.

Figura 3.11: Maschera del condizionatore d'aria UPnP.

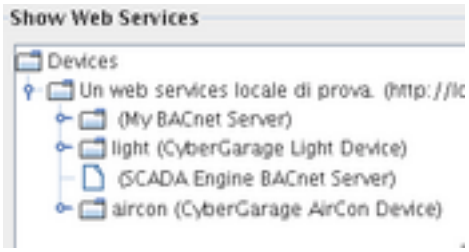

Figura 3.12: Lista Dispositivi.

questo caso fa parte della tecnologia BACnet. Va notato che per poter visionare il cambiamento sul dispositivo BACnet occorre richiederlo cliccando sul bottone della finestra (quella posta più in basso nelle figure 3.13 e 3.14).

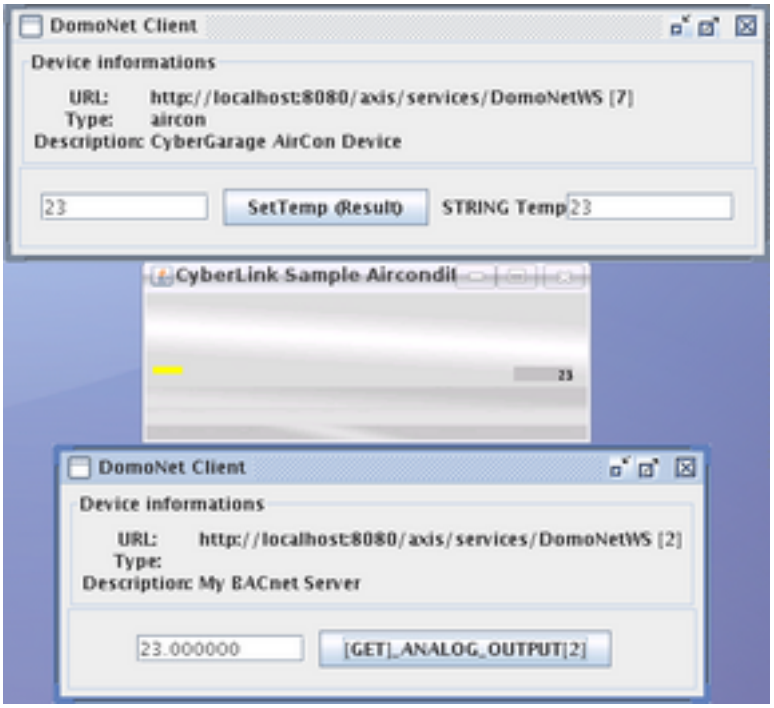

Figura 3.13: Settaggio della temperatura a 23 gradi sul dispositivo UPnP e corrispondente risultato sull'oggetto ANALOG_OUTPUT[2] della rete BACnet.

Si può notare che il LinkedService in questo caso non fa nessuna interpretazione dei dati in ingresso, infatti qualunque valore scritto nella finestra di settaggio, viene passato integralmente fino a raggiungere il servizio del dispositivo BACnet.
Un altra cosa da notare è che potenzialmente il dispositivo BACnet avrebbe gestito settaggi di numeri in virgola mobile, ma provando a settare nel dispositivo UPnP una temperatura come 23.5 l'operazione non va a buon fine per i limiti del dispositivo UPnP, per cui gli esempi sono stati effettuati con numeri interi.
Si potrebbe immaginare un'applicazione pratica in cui all'utente potrebbe essere proposta la configurazione dei dispositivi reali attraverso delle interfacce grafiche (scritte per esempio in java e comunicanti attraverso UPnP) e le varie modifiche di stato effettuate sull'interfaccia grafica verrebbero passate attraverso DomoNet al dispositivo reale disaccoppiando l'ambiente per realizzare la grafica dall'ambiente per realizzare la logica dell'applicazione

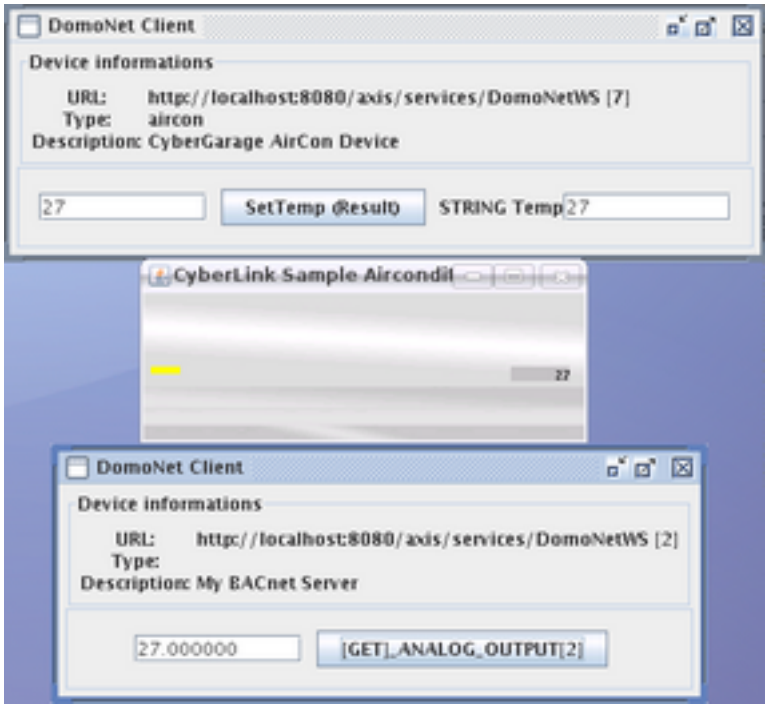

Figura 3.14: Settaggio della temperatura a 27 gradi.

(spesso per esempio programmando i PLC esistono ambienti unici che vengono usati sia per la grafica di controllo sia per programmare la logica del dispositivo sul quale si possono usare solo le librerie definite per quell'ambiente con evidenti limiti sull'estensibilità).

3.7 Test 6 - LinkedService 2: Da ANALOG_VALUE BACnet a Lampada UPnP

AZIONI:

1. Nel file

   ```
   ${DomoRoot}/domonet/trunk/xml/domoDevices.xml
   ```

   è stato definito il LinkedService che collega il servizio relativo all'oggetto ANALOG_VALUE (nel nostro esempio il ANALOG_VALUE[1]) al servizio (setPower) offerto dalla Lampada UPnP come mostrato nella tabella 3.5.
2. Identico al punto 1 del Test 1.

3. Identico al punto 2 del Test 1.
4. Identico al punto 3 del Test 2.
5. Identico al punto 4 del Test 2.
6. Sulla macchina <C> dalla shell *S5* lanciare il comando

   ```
   loadUPnP.sh LightFrame
   ```

   Se l'operazione va a buon fine dovrebbe comparire una maschera come quella mostrata nella figura 3.15 che simula le caratteristiche di un lampada (inizialmente spenta).
7. Identico al punto 7 del Test 5.
8. A questo punto navigando sulla directory del web Service, come mostrato nella figura 3.12, vengono visualizzati tutti i dispositivi trovati da DomoNet. Una volta trovato il *domoDevice My BACnet Server* occorre cercare al suo interno l'oggetto ANALOG_VALUE[1] come mostrato nella figura 3.16.
9. Cliccando sull'oggetto ANALOG_VALUE[1] si apre una finestra simile a quella proposta nelle immagini 3.17 e 3.18 lato sinistro. Su questa finestra, inserendo nella riga di comando di destra il valore "1" e premendo sul bottone centrale si noterà l'accensione della lampadina, se invece si inserisce "0" premendo il bottone centrale la lampadina si spegnerà. Il tutto viene mostrato rispettivamente nelle figure 3.17 e 3.18.

Figura 3.15: Maschera della Lampada UPnP.

```
...
<device description="My BACnet Server" id="2" manufacturer=" [0]" position=""
positionDescription="ESAC sede di via Manzoni n 1, Collegno (TO)"
serialNumber="/1/40" tech="BACNET" type="" url="">
...
<service description="" name="/1/40/2/1" output="DOUBLE" outputDescription=""
prettyName="ANALOG_VALUE[1]">
<input description="" name="ANALOG_VALUE[1]" type="DOUBLE" />
<linkedService hasValue="0.000000" id="1" ifInput="ANALOG_VALUE[1]" service="SetPower" url="">
<linkedInput to="Power" value="0" />
</linkedService>
<linkedService hasValue="1.000000" id="1" ifInput="ANALOG_VALUE[1]" service="SetPower" url="">
<linkedInput to="Power" value="1" />
</linkedService>
</service>
...
</device>
...
```

Tabella 3.5: Frammento xml per configurare il LinkedService del test 6.

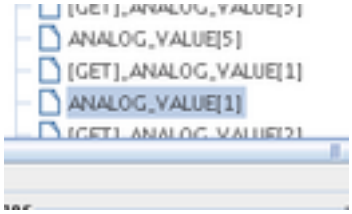

Figura 3.16: Dettaglio su ANALOG_VALUE[1].

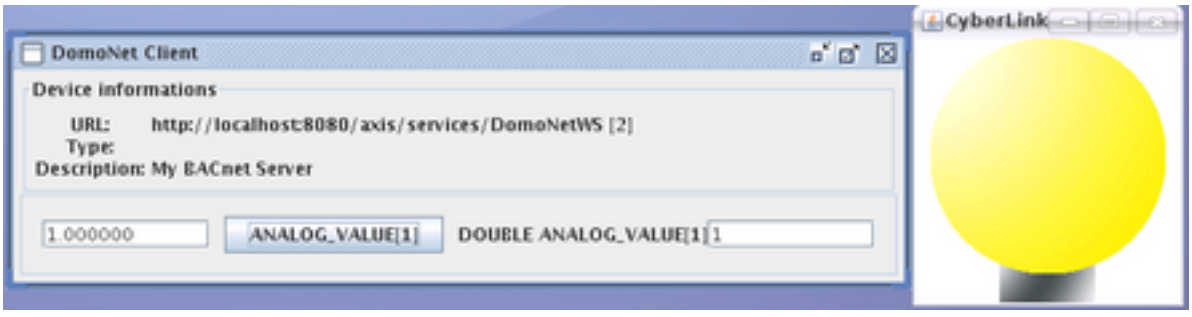

Figura 3.17: Accensione lampadina inserendo 1.

COMMENTI:

In questo test viene dimostrato il passaggio inverso del test 5, vale a dire che questa volta ad un cambio di stato all'interno della tecnologia BACnet ne

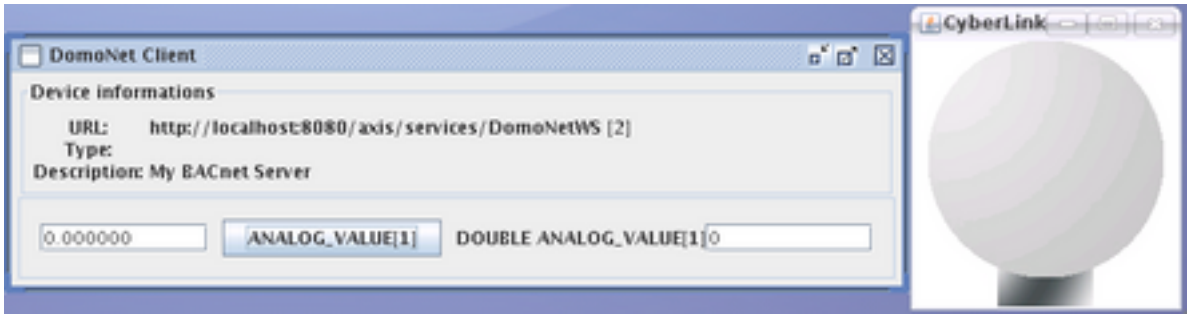

Figura 3.18: Spegnimento lampadina inserendo 0.

corrisponde un altro cambio di stato all'interno della tecnologia UPnP.
Guardando la descrizione del linkedService si può notare come è stato necessario effettuare un'interpretazione numerica nel passaggio del dato da BACnet a UPnP (hasValue=0.000000 convertito in value=0).

4 Problemi e Soluzioni

In questa sezione vengono discussi e affrontati i problemi riscontrati nell'implementazione del *BACnetManager*.

4.1 Il Polling al Web Service

L'architettura dei Web Service pur essendo più raffinata dell'architettura web, presenta lo stesso il problema del **servente passivo**, vale a dire che nelle architetture *client-server* il server non inizia mai una comunicazione, piuttosto rimane in attesa che qualche cliente faccia delle richieste alle quali invia delle risposte.
Questo significa che il server a priori non sa nulla sui suoi potenziali clienti fin tanto che essi non effettuano richieste. Da queste considerazioni generali si arriva alle seguenti conclusioni:

- il server non può inviare eventuali aggiornamenti perchè non sa chi sono i diretti interessati;
- ipotizzando che il server sia in grado di sapere chi sono i diretti interessati, se questi non sono in ascolto, esso non può inviare gli aggiornamenti.

Il web Service di BACnet nel suo *back-end* mantiene un'immagine logica della rete BACnet, che, come abbiamo visto, gestisce gli aggiornamenti dei dati anche con gli eventi.
La **non** trasparenza del BACnetWS e più in generale dei web Service, in merito alla gestione a eventi, costringe l'attuale implementazione all'uso di scansioni iterative sull'intera immagine logica della rete BACnet per verificare la presenza di eventuali cambiamenti di stato al fine di mantenere aggiornata la corrispondente immagine del *BACnetManager*.
Questo comporta che il tempo per effettuare un aggiornamento del *BACnetManager*, al crescere del numero di dispositivi con relativi oggetti e properties associate, cresce linearmente al numero totale di properties. In generale, ciò risulta molto pesante da gestire già con pochissimi dispositivi, considerato che un dispositivo BACnet solitamente mantiene molti oggetti al suo interno.

La scelta dell'implementazione attuale è derivata dal fatto che allo stato attuale, non è stato possibile avere accesso ai sorgenti del Web Service di BACnet.
Considerato che questo problema rende poco praticabile nei casi reali l'intero lavoro di integrazione di BACnet in DomoNet, è stato effettuato uno studio al fine di trovare una soluzione, che sarà oggetto del paragrafo successivo.

4.1.1 Soluzione: WS-Notification

WS-Notification [21] è uno standard che definisce un approccio basato su Web Services fondato su un paradigma *publish/subscribe*. L'informazione oggetto del paradigma *publish/subscribe* viene catalogata in *topics*.
I promotori dello standard sono importanti industrie nel settore della tecnologia, quali IBM, Akamai Technologies, Computer Associates International, SAP AG, Fujitsu Laboratories of Europe, Globus, Hewlett-Packard, Sonic Software, TIBCO Software.
Le specifiche dello standard comprendono:

- La messaggistica che deve essere scambiata tra i fornitori dei servizi coinvolti nel processo di notifica e la messaggistica relativa alla presenza di un eventuale servizio che funga da server centralizzato e che consenta la pubblicazione di servizi e messaggi.

- I requisiti operazionali che devono soddisfare tutti gli elementi coinvolti nel processo di notifica.

- Un modello per la descrizione delle topics.

Tramite WS-Notification risulta quindi possibile la costruzione di un sistema scalabile indipendente dalla piattaforma costituito da nodi autonomi che si scambiano tra loro messaggi in maniera asincrona.

La soluzione non richiede stravolgimenti del protocollo *client-server*, piuttosto prevede una modellazione della trasmissione dei dati in cui le 2 parti in questione (client e server) siano in grado di realizzare sia le funzionalità proprie, sia quelle del proprio reciproco. Lo standard WS-Notification si preoccupa di definire la sintassi e la semantica e dei messaggi di scambio tra le 2 entità ridefinendo con una notazione *SOAP* il classico paradigma **publish/subscribe** [13].

Per semplicità di utilizzo, seguendo le indicazioni date da [22], è stato effettuato un test realizzando 2 *servlet* [20] e utilizzando come ambiente applicativo Tomcat [23].
La prima *servlet* si comporta da cliente, ovvero ha il compito di registrare il proprio interesse verso delle tematiche (Topics) offerte da un'altra servlet che si comporta da servente.

Figura 3.19: Form del Client

[13] Per maggiori dettagli visitare i seguenti indirizzi:
`http://it.wikipedia.org/wiki/Publish/subscribe` e
`http://it.wikipedia.org/wiki/Data_Distribution_Service`

Attraverso la form mostrata in figura 3.19[14] l'utente inserisce i dati di input corrispondenti al *Client* che può essere una qualunque stringa che nel caso di BACnet potrebbe corrispondere a "BACnetManager", anche se proprio questo campo per i nostri fini risulta ridondante. Il campo*Topic* invece nel caso di *BACnet* potrebbe essere il *Path* per il quale si richiedono gli aggiornamenti.
Cliccando sul bottone "Invia Richiesta", parte l'elaborazione della servlet Client che consiste in:

1. Costruzione di un messaggio standard secondo le specifiche di *WS-Notification* partendo dai dati di input al fine di adempiere le intenzioni del client, vale a dire la sottoscrizione a un *topic* della servlet Server.
2. Stampa a video del messaggio standard. Nella tabella 3.6 viene mostrato il risultato con:
   Client="BACnetUser" e *Topic*="/1/22/18/27".
   Si può notare che il campo *Client* non compare nel messaggio in quanto in fase di elaborazione viene associato un identificativo numerico unico (che nell'esempio corrisponde a "4").
3. Invio del messaggio standard alla servlet Server.
4. Attesa di un responso a tale richiesta. che non corrisponde agli aggiornamenti, ma serve a capire se la richiesta fatta alla servlet Server è andata a buon fine.

La servlet Server raccoglie (attraverso il doPost) la richiesta della servlet Client e nel caso di esito positivo memorizza l'identificativo e la URL del partner di comunicazione in una sua struttura dati e contestualmente invia un responso positivo il cui esito è mostrato nella tabella 3.7.

Per aggiornare il valore relativo a un *Topic* occorre compilare una form come quella della figura 3.20[15].
In questa form vanno riempiti i campi di input:

- *Topic* indicante il topic che si vuole aggiornare.
- *Value* ovvero il valore aggiornato del topic.

[14]Eseguendo l'installazione classica di tomcat sulla propria macchina e copiando la webApp *WSNotification* nella directory webApps di tomcat, la URL per ottenere questo risultato è:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header>
<wsa:Action xmlns:wsa="http://www.w3.org/2005/02/addressing">
http://docs.oasis-open.org/wsn/2004/06/WS-BaseNotification/Subscribe
</wsa:Action>
<wsa:To xmlns:wsa="http://www.w3.org/2005/02/addressing" SOAP-ENV:mustUnderstand="1">
http://localhost:8080/WSNotification/servlet/ProducerServiceServlet
</wsa:To>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
<wsnt:Subscribe
xmlns:wsnt= "http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd">
<wsnt:ConsumerReference>
<wsa:Address xmlns:wsa="http://www.w3.org/2005/02/addressing">
http://localhost:8080/WSNotification/servlet/ConsumerServiceServlet
</wsa:Address>
<wsa:ReferenceProperties xmlns:wsa="http://www.w3.org/2005/02/addressing">
<search:subID xmlns:search="http://www.example.com/searches">
4
</search:subID>
</wsa:ReferenceProperties>
</wsnt:ConsumerReference>
<wsnt:TopicExpression
wsnt:Dialect= "http://docs.oasis-open.org/wsn/2004/06/TopicExpression/Simple">
/1/22/18/27
</wsnt:TopicExpression>
</wsnt:Subscribe>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Tabella 3.6: Messaggio di sottoscrizione

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<wsnt:SubscribeResponse
xmlns:wsnt="http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd">
<wsnt:SubscriptionReference>
<wsa:Address xmlns:wsa="http://www.w3.org/2005/02/addressing">
http://localhost:8080/WSNotification/servlet/ProducerServiceServlet
</wsa:Address>
<wsa:ReferenceProperties xmlns:wsa="http://www.w3.org/2005/02/addressing">
<searchService:subID xmlns:searchService="http://www.example.com/searchService">
3
</searchService:subID>
</wsa:ReferenceProperties>
</wsnt:SubscriptionReference>
</wsnt:SubscribeResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Tabella 3.7: Messaggio di risposta.

Figura 3.20: Form del Server

Cliccando sul bottone "Consegna aggiornamento" parte l'esecuzione della servlet Server il cui scopo sarà quello di costruire un messaggio coerente allo standard *WS-Notification* da inviare alla servlet Client.
Un esempio di messaggio è quello mostrato nella tabella 3.8 dove i parametri passati alla form sono:
Topic="/1/22/18/27"e *Value*="13".

`http://localhost:8080/WSNotification/Consumer.html`

[15]Eseguendo l'installazione classica di tomcat sulla propria macchina e copiando la webApp *WSNotification* nella directory webApps di tomcat, la URL per ottenere questo risultato è:

`http://localhost:8080/WSNotification/Producer.html`

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header>
<wsa:Action xmlns:wsa="http://www.w3.org/2005/02/addressing">
http://docs.oasis-open.org/wsn/2004/06/WS-BaseNotification/Notify
</wsa:Action>
<wsa:To xmlns:wsa="http://www.w3.org/2005/02/addressing" SOAP-ENV:mustUnderstand="1">
http://localhost:8080/WSNotification/servlet/ConsumerServiceServlet
</wsa:To>
<search:subID xmlns:search="http://www.example.com/searches">
4
</search:subID>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
<wsnt:Notify
  xmlns:wsnt="http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd">
<wsnt:NotificationMessage>
<wsnt:Topic wsnt:Dialect="http://docs.oasis-open.org/wsn/2004/06/TopicExpression/Simple">
/1/22/18/27
</wsnt:Topic>
<wsnt:Message>
<searchService:NewSearchResult xmlns:searchService="http://www.example.com/searchService">
13
</searchService:NewSearchResult>
</wsnt:Message>
</wsnt:NotificationMessage>
</wsnt:Notify>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Tabella 3.8: Messaggio di notifica aggiornamento Topic.

Come si può notare la servlet Server calcola l'identificatore del cliente ricevente e lo spedisce secondo il *namespace* del cliente (nell'esempio della tabella il "4" corrisponde all'identificativo mentre il *namespace* di riferimento è "`http://www.example.com/searches`". Entrambi i dati erano stati inviati inizialmente dal cliente come si può verificare nel messaggio della tabella 3.6).

A questo punto la servlet Client, ricevendo il messaggio crea (nel nostro esempio di test) un file di logs dove memorizza il messaggio ricevuto.
In un caso reale avrebbe dovuto recuperare i parametri dal messaggio ricevuto e aggiornare la struttura dati del *BACnetManager*.

Il test prevede che sia l'utente a modificare il valore del Topic per enfatizzare il grado di disaccoppiamento tra server e client verificando inoltre che il meccanismo funzioni a eventi.
Attraverso questo esempio si ottiene il risultato sperato, ovvero un meccanismo che riesce a risolvere il problema e tale meccanismo fa uso di un protocollo di comunicazione **open standard** (fa uso di messaggi SOAP).

Attraverso l'uso della servlet è stato possibile scindere (in modo semplice) la provenienza del flusso dei dati. Le comunicazioni *user-servlet* avvengono sempre attraverso richieste di tipo GET mentre le richieste di tipo POST sono state usate per comunicazioni di tipo *servlet-servlet*. É da notare che le *WS-Notification* sono implementabili anche attraverso i *socket*[16] eliminando la necessità di installare l'ambiente applicativo (Tomcat per intendersi!).
Quanto affermato sopra è possibile in quanto lo standard definisce la sintassi e la semantica dei messaggi e non il tipo di canale di comunicazione.

Tutto ciò mostra come è stato possibile gestire il trasferimento dei dati soltanto al verificarsi dell'evento di generazione degli stessi.
Se sarà possibile modificare i sorgenti proprietari di BACnet, e quindi il comportamento del BACnet WebService, allora sarà possibile rendere il BACnet WebService trasparente alla gestione degli eventi.
In questo caso, si potrà ridimensionare in modo significativo il tempo necessario ad effettuare gli aggiornamenti del *BACnetManager*.

[16]Per maggiori dettagli sulle socket andare all'indirizzo:
`http://www.lorenzobettini.it/articoli/socket1/socket.html`

Bibliografia

Riferimenti per DomoNet

[1] Dario Russo, *DomoNet Architecture*
`http://domonet.sourceforge.net/`

[2] Tesi di Laurea di Dario Russo, *La domotica e Internet. Una soluzione per l'interoperabilità*
`http://etd.adm.unipi.it/theses/available/etd-06192006-105322/unrestricted/tesiDomonetDarioRusso.pdf`

[3] Tesi di Laurea di Francesco Conversano, *Una soluzione open standard per l'interoperabilità dei middleware domotici: l'integrazione di My Home BTicino*
`http://etd.adm.unipi.it/theses/available/etd-11222007-172412/unrestricted/Tesi.pdf`

[4] Miori V., Tarrini L., Manca M., Tolomei G (2006), *An open standard solution for domotic interoperability. In: IEEE Transactions on Consumer Electronics*, vol. 52 (1) pp. 97-103. IEEE.

[5] V. Miori, L. Tarrini, M. Manca, G. Tolomei (2006), *DomoNet: A framework and a prototype for interoperability of domotics middlewares based on XML and WebServices*, pp. 117-119. IEEE International Conference on Consumer Electronics (ICCE) (Las Vegas, USA, 7-11 gennaio 2006).

[6] Miori V., Russo D. (2006), *Una piattaforma software universale per i sistemi domotici*, Progetto HATS (Home Automation Technologies and Standard), laboratorio di domotica ISTI-CNR. PR n. D2.0.

[7] Tokunaga, E., Ishikawa, H., Kurahashi, M., Morimoto, Y., and Nakajima, T. (2002), *A Framework for Connecting Home Computing Middleware* In Proceedings of the 22nd international Conference on Distributed Computing Systems (July 02 - 05, 2002). ICDCSW. IEEE Computer Society, Washington, DC, 765-770.

Riferimenti per BACNET

[8] sito della società che ha sviluppato lo standard BACnet
`http://www.ashrae.org/`

[9] *BACnet - A standard communication infrastructure for intelligent buildings*
`http://www.bacnet.org/Bibliography/AIC-97/AIC1997.htm`

[10] *The Language of BACnet-Objects, Properties and Services*
`http://www.bacnet.org/Bibliography/ES-7-96/ES-7-96.htm`

[11] Il riferimento per il Web Service di BACnet, (29 settembre 2006) *Addendum C to ANSI/ASHRAE Standard*
`http://www.bacnet.org/Addenda/Add-2004-135c.pdf`

[12] *Web Services Interoperability Organization (WS-I) Basic Profile 1.0*
`http://www.service-architecture.com/web-services/articles/web_services_interoperability_organization_ws-i.html`
`http://www.ws-i.org/`

[13] The *SCADA Engine* Home Page
`http://www.scadaengine.com/index.html`

[14] Sito web ufficiale dell'azienda *Delta Controls*
`http://www.deltacontrols.com/`

[15] James F. Kurose, Keith W. Ross, *Internet e Reti di Calcolatori*

Riferimenti per BACnetManager

[16] Riferimenti riguardanti l'*Abstract Factory Pattern*
`http://wiki.ugidotnet.org/default.aspx/UGIdotNETWiki/PatternAbstractFactory.html`
`http://it.wikipedia.org/wiki/Abstract_factory`

[17] *Java Event Model*
`http://www.microbyte.it/Italiano/JavaNotes/events.asp`

[18] Raffigurazioni dell'Internetwork di BACnet
`http://ethernet.industrial-networking.com/articles/articledisplay.asp?id=11`

[19] Spiegazione del problema Lettori Scrittori
`http://profs.sci.univr.it/~rocchess/htmls/corsi/LSO/JavaThreads/javathreads5.txt`

[20] *Java Servlet Technology*
`http://java.sun.com/products/servlet/`

[21] IBM, Akamai Technologies, Computer Associates International, SAP AG, Fujitsu Laboratories of Europe, Globus, Hewlett-Packard, Sonic Software, TIBCO Software (01 Oct 2004) *WS-Notification*
`http://www-128.ibm.com/developerworks/library/specification/ws-notification/`

[22] *Using WS-Notification*
`http://www.ibm.com/developerworks/library/gr-ws-not/`

[23] *Apache Tomcat*
`http://tomcat.apache.org/`

[24] *Apache Ant*
`http://ant.apache.org/`