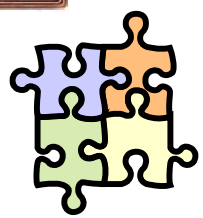
Middleware per applicazioni in ambienti pervasivi

Silvia Vecchi
D.E.I.S.
Università di Bologna

Contesto

Convergenza di tecnologie

Sensori



Wireless Networks

Computing/Internet

Nuovi scenari applicativi e servizi a valore aggiunto

Implementazione dei servizi

Cosa c'e'...

applicazioni ad-hoc, poco flessibili/scalabili, "cablate", costi di sviluppo spesso elevati

... cosa servirebbe

Middleware

per facilitare/migliorare sviluppo, prototipazione, mantenimento, efficienza delle applicazioni

Outline

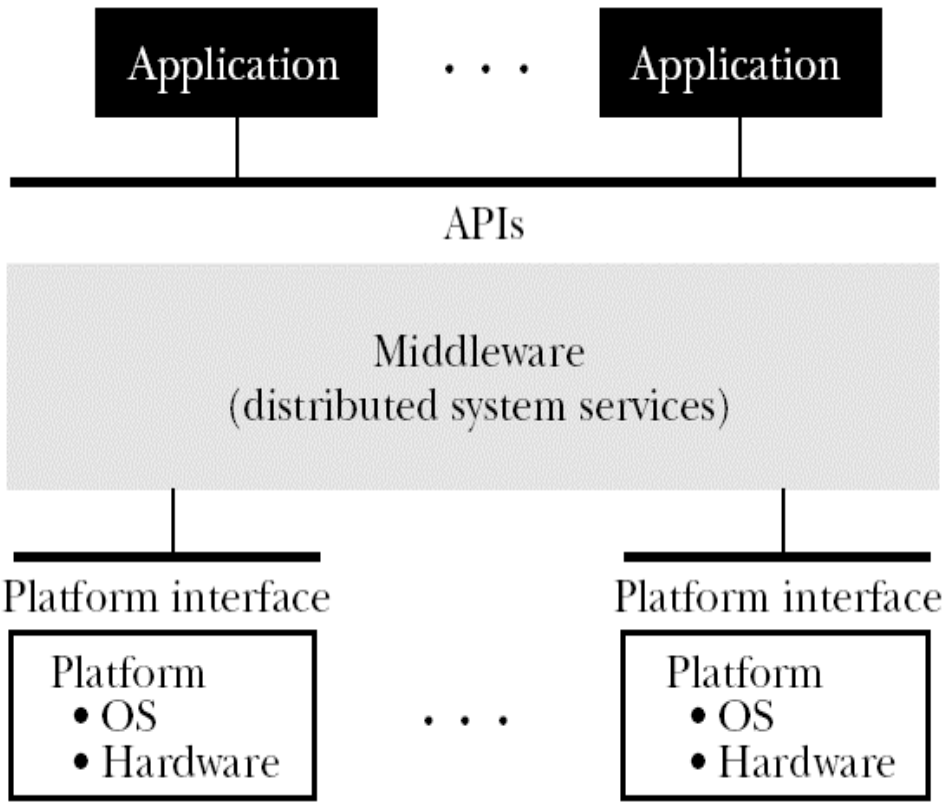
- Concetti di base sui Middleware:
  definizioni, funzionalita', classificazione
- Direzioni di ricerca:
  Middleware per nuovi contesti applicativi
  - mobile computing
  - wireless sensor networks
  - smart environments
- Approfondimenti:
  SOMA, mobile-agent based middleware

Prima Parte:

Concetti di base sui Middleware

Definizioni

- Strato software fra le applicazioni e il Sistema Operativo e/o la rete
- Disaccoppiamento tra i livelli di sistema:
  astrazioni di programmazione e **infrastruttura di implementazione**
- Integrazione tra tecnologie diverse e di sistemi legacy



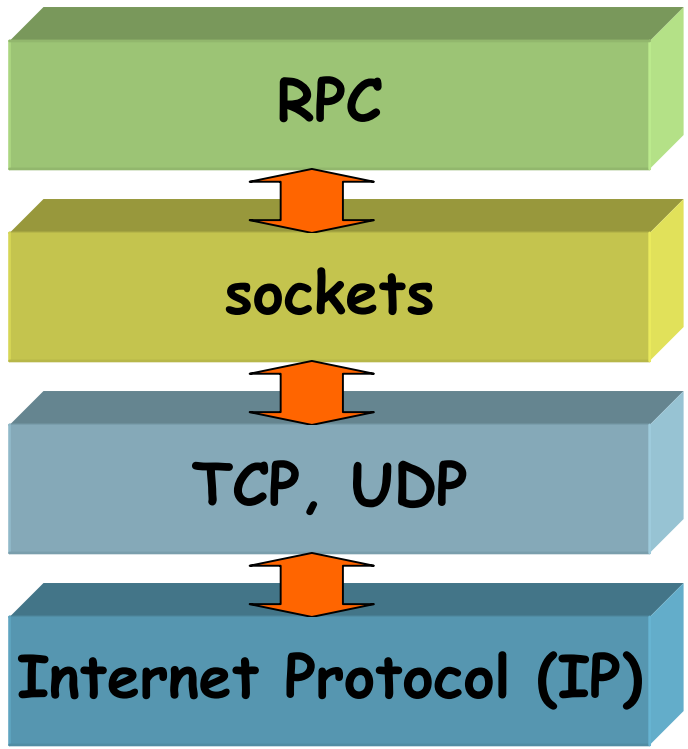
Astrazioni di programmazione

Il middleware fornisce astrazioni di programmazione che evitano alle applicazioni di implementare alcune funzionalita' "from scratch"

Es. Remote Procedure Calls (RPCs)

Evitano di:

- Gestire il canale di comunicazione tra le sockets
- Definire un protocollo tra le due applicazioni
- Specificare un formato di rappresentazione delle informazioni da scambiare



Infrastruttura di implementazione

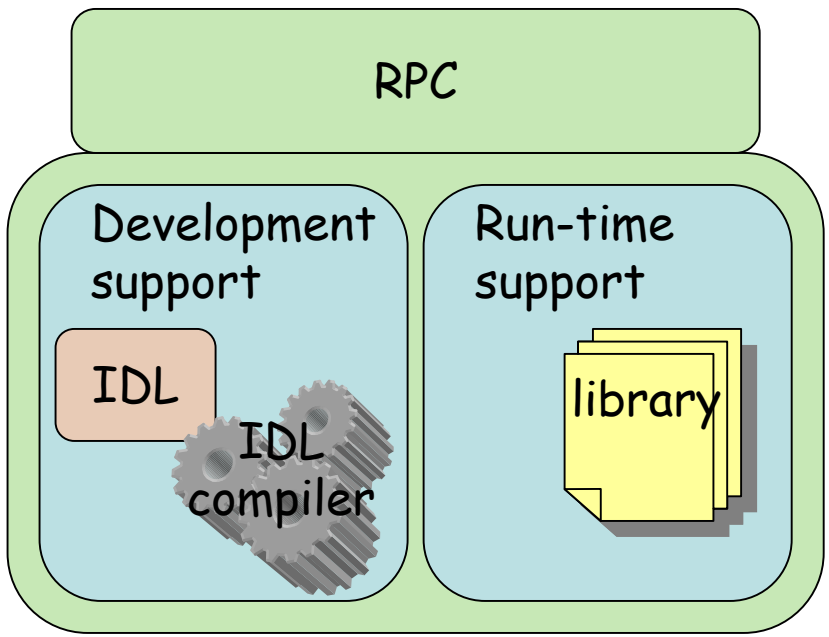
L'implementazione delle astrazioni di programmazione necessita di un'infrastruttura software di supporto, a development time e a run time

Es. RPCs: IDL+ IDL compiler + librerie

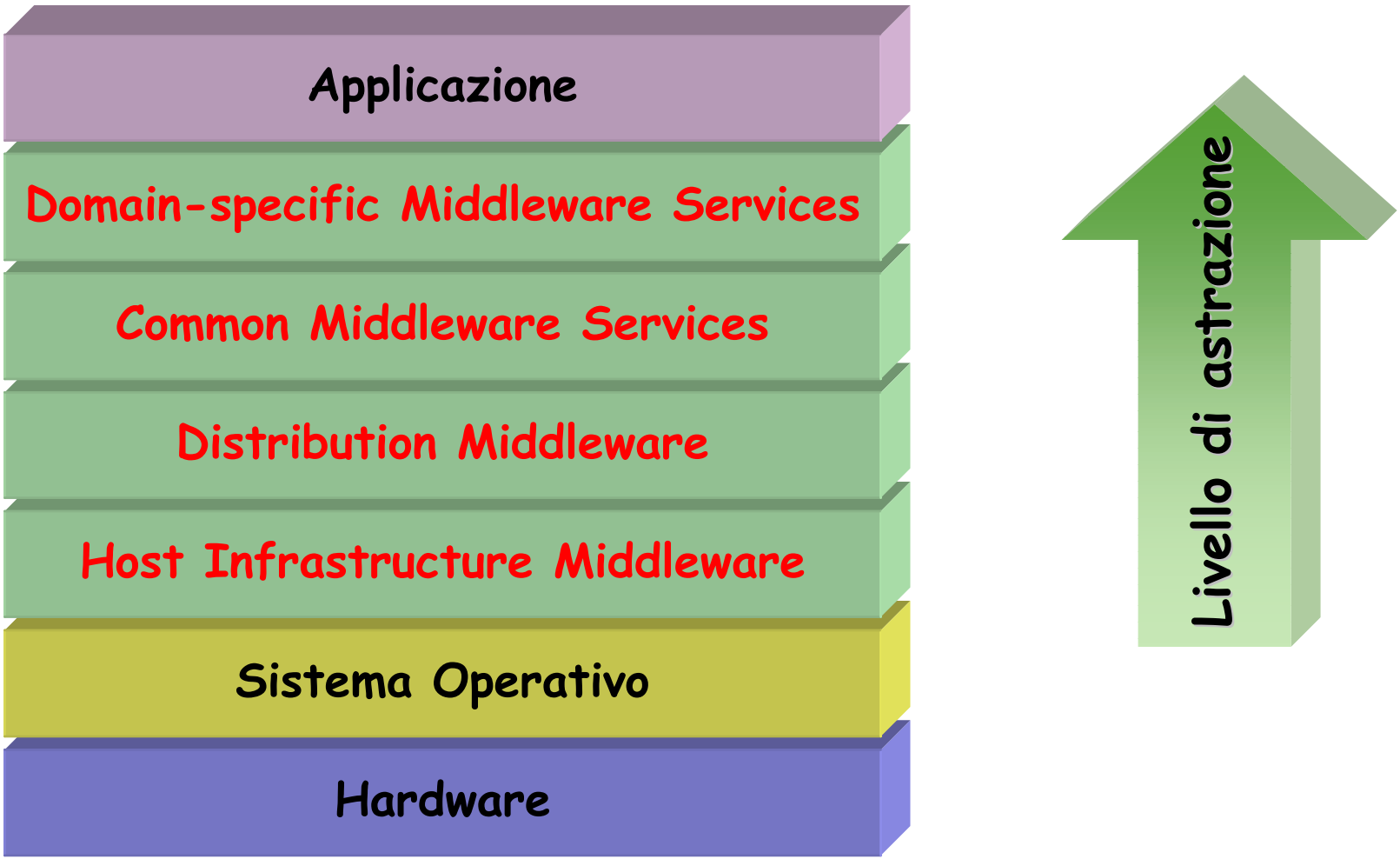
Estensioni:
Autenticazione -> Supporto alla security
Indirizzi dinamici -> Naming service
...

Piattaforme middleware spesso complesse e "pesanti"

Trade-off tra ricchezza delle astrazioni fornite
e complessita' di infrastruttura.



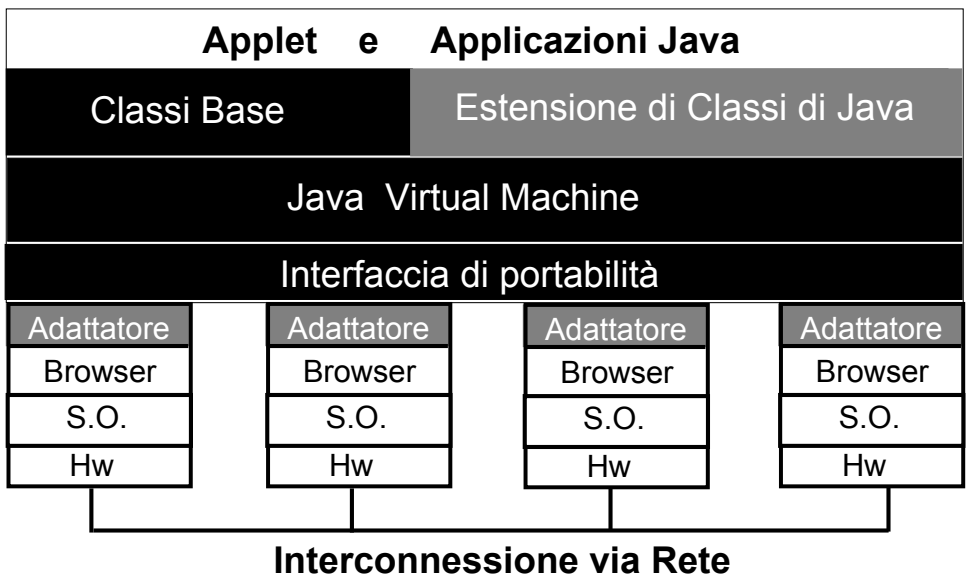
Middleware: livelli di sistema



Host Infrastructure MW

Livello che incapsula e migliora i meccanismi nativi di comunicazione e concorrenza del SO per creare componenti di programmazione riusabili

(es. Java sockets, Java threads)



API che permettono di avere un supporto locale unificato per i diversi sistemi

Esempi:

JVM, .NET, …

Distribution MW

Livello che fornisce modelli di programmazione distribuita che automatizzano ed estendono gli strumenti di programmazione di rete forniti dal livello sottostante

Facilitano comunicazione e coordinamento dei diversi nodi che partecipano all'applicazione con

- modello delle risorse
- API di comunicazione
- altre funzionalità accessorie per la comunicazione
  supporto di naming, di discovery, …

Esempi: RMI, CORBA, DCOM, …

Common MW Services

Servizi indipendenti dal dominio applicativo, che forniscono funzionalità di uso comune (spesso orientate ai componenti)

Esempi di common services:

eventi, logging, streaming, sicurezza, fault tolerance, …

Esempi:

Enterprise Java Beans, CORBAServices, .NET Web services

Domain Specific MW Services

Servizi specifici modellati "verticalmente" sulle esigenze di un particolare dominio applicativo

Esempi:

gruppi di lavoro all'interno di *OMG* stanno definendo e standardizzando funzionalita' per domini specifici (e-commerce, healthcare, telecom, command and control, process automation)

Syngo di Siemens Medical Engineering Group

Boeing Bold Stroke basato su CORBA

Contesti di utilizzo

Contesti tradizionali:

- Applicazioni fortemente sincrone → RPC-based Middleware
- Applicazioni per transazioni distribuite → Transaction Processing (TP) Monitors
- Applicazioni sincrone OO → Distributed Object Brockers
- Applicazioni asincrone → Message Oriented Middleware (MOM)

Contesti piu' recenti:

- Applicazioni per ambienti dinamici → Adaptive & Reflective Middleware
- Applicazioni multimediali → QoS & Multimedia Middleware

RPC-based Middleware

RPC come strumenti per comunicazione C/S

- Interface Definition Language IDL per l'accordo
- Cliente bloccato in modo sincrono in attesa della risposta dal Server
- Gestione eterogeneità dei dati
- Uso di Stub/Skeleton per la Trasparenza
- Binding spesso statico (o poco dinamico)

Modello un po' rigido, poco scalabile e replicabile con QoS

Il server deve essere presente e prevedere i processi necessari in modo esplicito

Non si tiene conto di eventuali ottimizzazioni nell'uso delle risorse

TP Monitors

Ottimizzazione delle connessioni con database e gestione di transazioni distribuite

- **comunicazione RPC-based**
- supporto per la gestione delle **transazioni**
- funzionalita' di interfacciamento ai database (**query**)
- monitoraggio e ottimizzazione delle risorse
- ambiente runtime che fornisce i servizi alle applicazioni

Esempi: **CICS** (**IBM**), **Tuxedo** (**BEA**), **MTS** (**Microsoft**)...

Distributed Object Brokers

Distribuzione di dati e codice attraverso invocazione di metodi tra oggetti remoti

Uso di oggetti come **framework** e di un **broker** come intermediario nella gestione delle operazioni

- **il modello ad oggetti** semplifica il progetto
- **il broker** fornisce i servizi base e altri addizionali
- **alcune operazioni** si possono fare in modo **automatico**
- **l'integrazione** di sistemi è **facilitata e incentivata**

Esempi: **CORBA, .NET e COM, Java RMI**

MOM Middleware

Distribuzione di dati e codice attraverso scambio di messaggi

Scambio di **messaggi non tipati** sia **sincrono** che **asincrono** con **strumenti ad-hoc (code e gestori)**

- **ampia autonomia** tra i componenti
- **asincronicità e persistenza** delle azioni
- **gestori (broker)** con politiche diverse e QoS diverso
- facilità nel **multicast, broadcast, publish / subscribe**

Esempi: **MQSeries (IBM), MSMQ (Microsoft), JMS (SUN)**

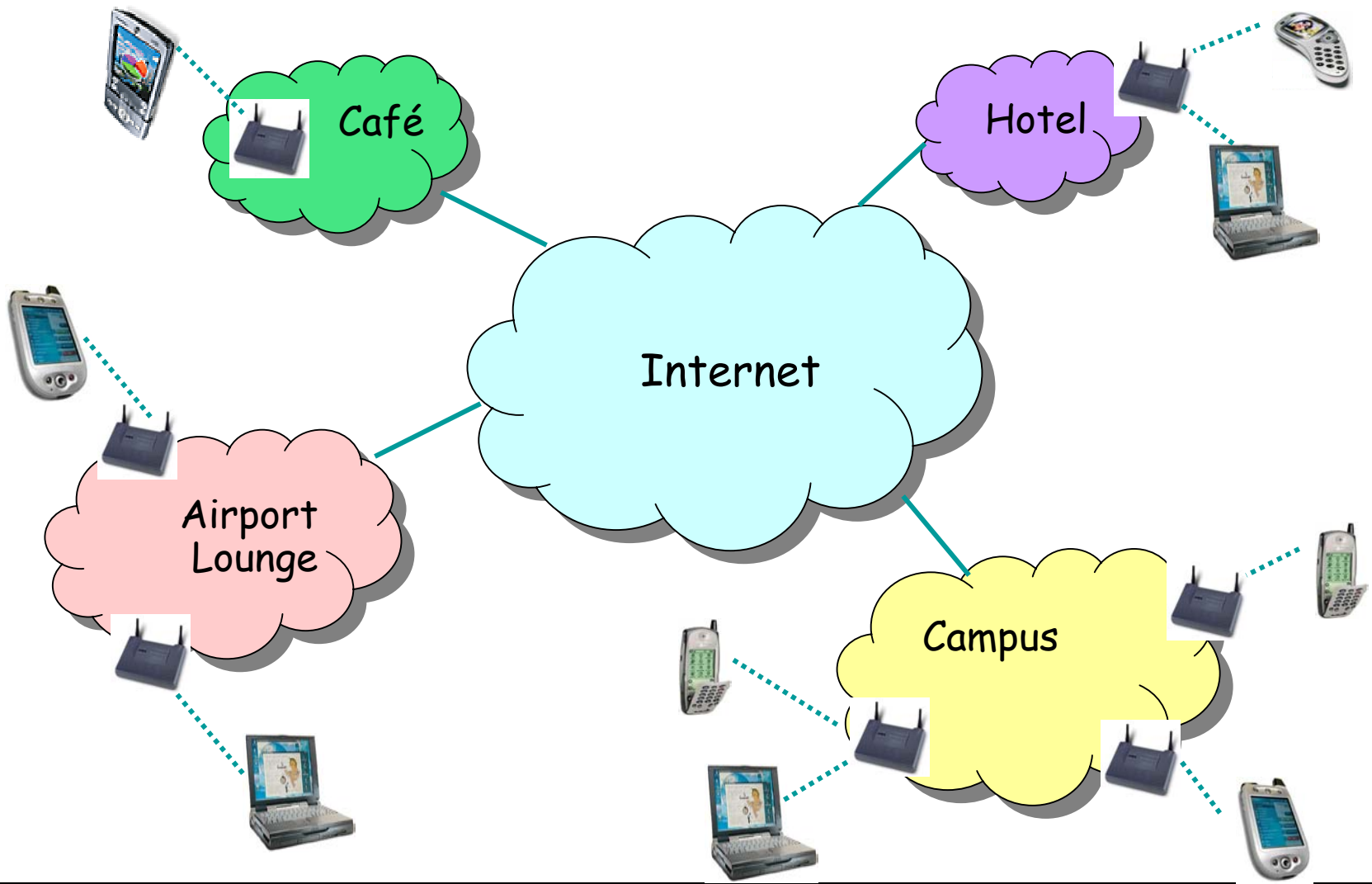
Seconda Parte:

Direzioni di ricerca sui Middleware

Aree di sviluppo dei middleware

- Middleware per **mobile computing**
- Middleware per **wireless sensor networks**
- Middleware per **smart environments**

Scenario 1: Hot-spots WiFi



MW per mobile computing

Obiettivi:

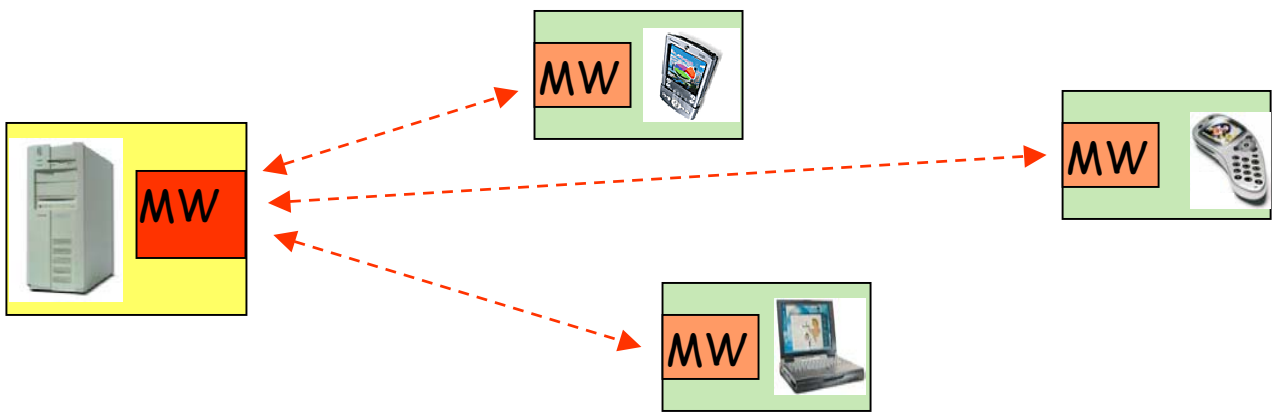
- facilitare la portabilita' delle applicazioni su piattaforme diverse
- ottimizzare l'uso delle risorse (rete in primis)
- migliorare la fruibilita' delle applicazioni adattandone le modalita' di delivery

Funzionalita':

- Transcodifica dei contenuti (statica e/o dinamica)
- Algoritmi di compressione, semplificazione delle transazioni
- Mantenimento della sessione
  (roaming con caching e predizione dei cammini)

MW per mobile computing: approcci

- **Device side**
  - small memory footprint, requisiti computazionali minimi
  - basati su common cross-platform development environments (es. J2ME, Qualcomm's Binary Runtime Environment for Wireless)
- **Infrastructure side**
  - supporto alle applicazioni su lato server
  - utilizzo di intermediari (es. application gateway, mobile proxy)



MW per mobile computing: progetti in corso

- **Progetti in ambito accademico**

  X-Middle (Univ. College London),
  Lime (Poli. Milano), SOMA (Univ. Bologna), …

- **Progetti in ambito non accademico**

  T-Spaces (IBM), Java Spaces (Sun), …

Nascita di consorzi il cui obiettivo e' la *standardizzazione* di diversi aspetti del service management in questi scenari
(es. Parlay, IPDR, PayCircle e JPay, …)

Scenario 2: Wireless Sensor Networks (WSNs)

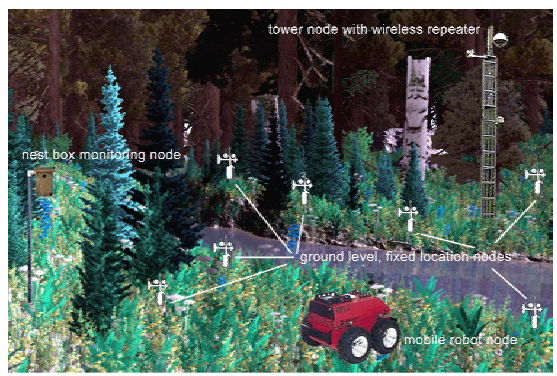
Architecture and Conservation
e.g. structural control

Military
e.g. battlefield support

Logistics and transportation
e.g. tracking goods

Environment science
e.g. monitoring volcanoes

Agricolture
e.g. monitoring crop growth



MW per WSNs

State-of-the-art

- Sviluppo e ottimizzazione dei protocolli di comunicazione (livelli fisico, data link, rete e trasporto)
- Strategie di data gathering e in-network processing

⇨ **soluzioni ad-hoc, interazione diretta con SO embedded o hardware**

Ancora pochissimo a livello middleware…

Obiettivi del middleware:

- astrazioni standard e portabili del sistema di sensori
- ambiente runtime per l'esecuzione di applicazioni sensing-based e interfacciamento con componenti esterni per resource intensive functions
- esecuzione concorrente di piu' applicazioni sulla stessa WSN
- gestione efficiente e adattativa delle risorse (alimentazione e computation/communication/sensing – CCS – capabilities)

MW per WSNs: resource management

Il middleware deve operare il tuning delle strategie di power saving, e determinare lo scheduling delle applicazioni sulle risorse corrispondenti. Servono:

Informazioni:

- dalla WSN: CCS capabilities ed energia residua dei nodi, connettivita'
- dall'applicazione: struttura dell'applicazione, data-control flow, requisiti di risorse, e vincoli di prestazioni.

Modelli di costo (tempo, energia) per CCS operations:

per tradurre i requisiti di performance application-level in parametri system-level. Tali modelli di costo dovrebbero essere parametrizzati dal carico di lavoro delle operazioni, e dalle condizioni della regione operativa.

Metriche di ottimizzazione:

per valutare la bonta' dell'allocazione e dell'adattamento delle risorse

Algoritmi efficienti:

per determinare le risorse hw necessarie per le applicazioni e definirne lo scheduling.

MW per WSNs: progetti in corso

- **Progetti in ambito accademico**

  Milan (Univ. Rochester), MANNA (Univ. Minas Gerais), SINA (Univ. Delaware), NEST (Vanderbilt Univ.), QUASAR (Univ. California Irvine), TinyLIME (Poli. Milano), …

- **Progetti in ambito non accademico**

  Crossbow, Telecom Italia Labs, …

Scenario 3: Smart environments

Home entertainment
e.g. Interactive rooms

Home automation
e.g. Smart home

Health care
e.g.Children/Elderly assistant

MW per smart environments

Obiettivo:

Supportare applicazioni context/user-aware
per ambienti pervasivi

Concetti chiave:

- *Context-awareness:*
  dipendenza del comportamento dal contesto
- *Proactivity:*
  anticipazione delle azioni
- *Human-Computer interaction:*
  passaggio/integrazione dalle tradizionali interfacce WIMP (windows/icons/mouse/pointer) a interfacce piu' naturali (vocali, gestuali, …).

MW per smart environments: approcci

Per supportare ***context-awareness***:

- *Bandwidth advising*:

  bandwidth monitor + predictor

- *Location sensing/tracking*:

  algoritmi di pattern matching e triangolazione (signal strength WiFi)

  active badge (infrared), RFID e RFID tag

Per supportare la ***proactivity***:

- *Activity inferencing*:

  coordinazione di informazioni eterogenee: locazione, movimento, scheduling eventi (ambiente e utente), profili di preferenza, ecc.

Per supportare la ***HCI***:

- gestione dell'in/output consistente rispetto alle diverse periferiche
- scelta "intelligente" delle interfacce da cui prelevare/verso cui indirizzare il flusso di informazioni

Middleware per smart environments: progetti in corso

- Home networking standard:
  Sun Jini, Microsoft UPnP (service/resource discovery)

- Progetti in ambito accademico:
  Aura (Carnegie Mellon Univ.), iRoom (Stanford Univ.),
  Smart Phone (Rutgers Univ.), Portolano (Univ. of Washington), …

- Progetti in ambito non accademico:
  Cooltown (HP), Easyliving (Microsoft),
  Microservers (Nokia), Personal Server (Intel)

- Soluzioni sviluppate in ambito industriale:
  LonWorks (home/building/utility/industrial automation)
  HAVi (Home Audio Video interoperability)

Terza Parte:

SOMA

SOMA

- **SOMA** (Secure and Open Mobile Agents)

  Universita' di Bologna

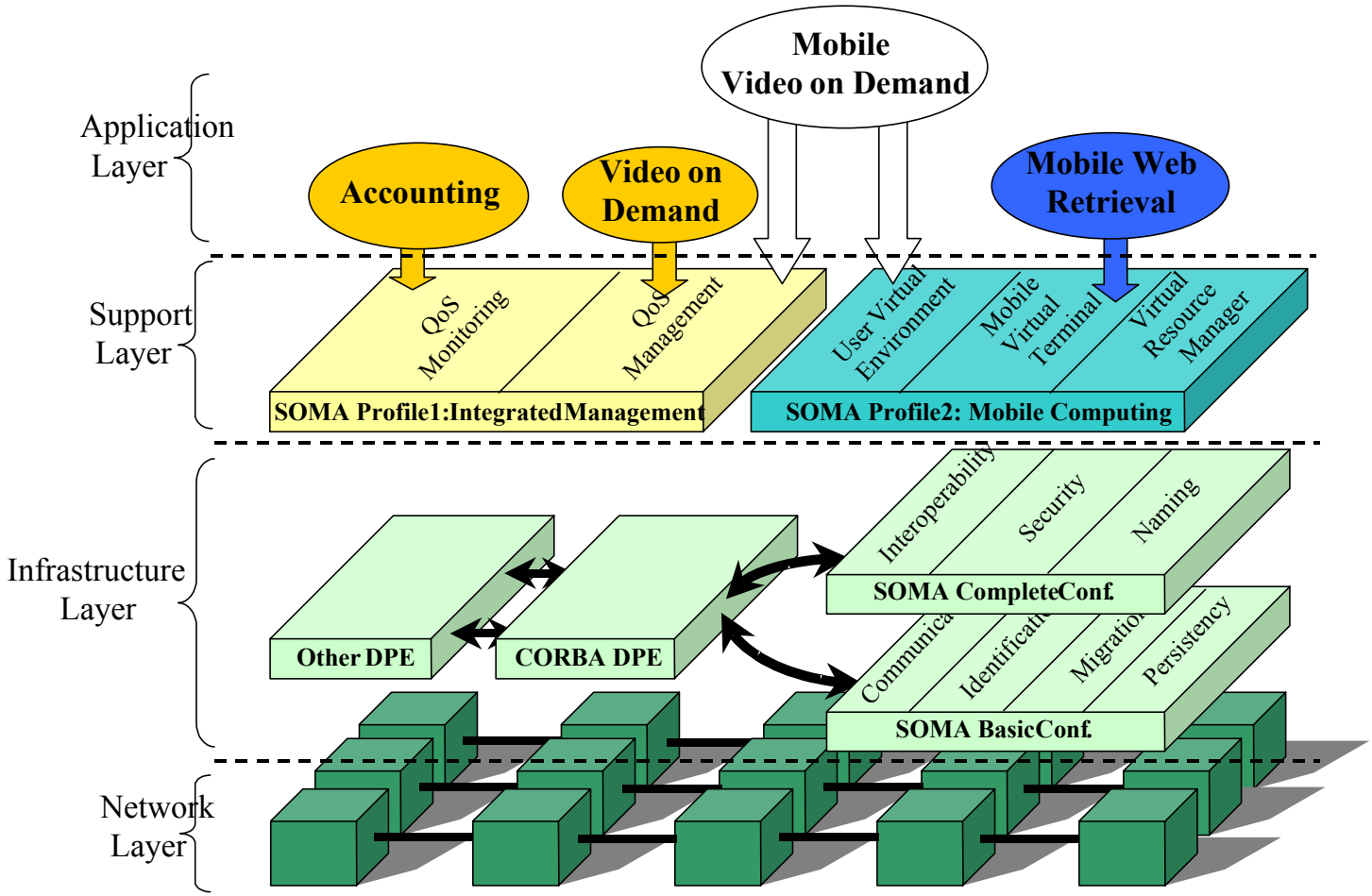
  Middleware per supportare applicazioni per ambienti mobili

Il middleware SOMA

- ***Agenti Mobili***:
  entità computazionali che ***operano per conto*** di un responsabile (***principal***) e che possono decidere ***autonomamente*** di ***migrare*** (***codice + stato***) a tempo di esecuzione verso un nuovo ambiente ospitante, per proseguire là le proprie operazioni

- **SOMA** è un middleware basato sulla tecnologia degli ***agenti mobili***, per la progettazione, l'implementazione e il supporto run-time di ***servizi distribuiti*** in ambienti ***globali***, ***aperti*** e ***non fidati***

Architettura a livelli di SOMA



Livelli di SOMA (1)

Servizi di infrastruttura:

- ***Mobilità***:
  permette ***riallocazione*** di risorse di rete e componenti di servizio
- ***Comunicazione***:
  - comunicazione ***locale*** per mezzo di ***oggetti condivisi*** (*blackboard*, *spazi di tuple*)
  - comunicazione ***remota*** tramite ***scambio di messaggi*** (*asincrono non-bloccante*)
- ***Persistenza***:
  supporta la ***sospensione di agenti*** in attesa di ***eventi asincroni*** esterni (riconnessione utente/terminale, disponibilità risorse, ...)
- ***Servizio di Nomi***:
  - assegna dinamicamente ***GUID*** ad ogni entità del sistema (***basic identification***).
  - permette di ***ritrovare*** ogni entità ***mobile*** del sistema, attraverso l'integrazione di ***differenti sistemi di nomi*** - *proprietario*, *DNS*, ***directory*** (*CORBA Directory*, *LDAP*), ***discovery*** (*Jini*, *SLP*)
- ***Sicurezza***

Livelli di SOMA (2)

Servizi di supporto alle applicazioni:

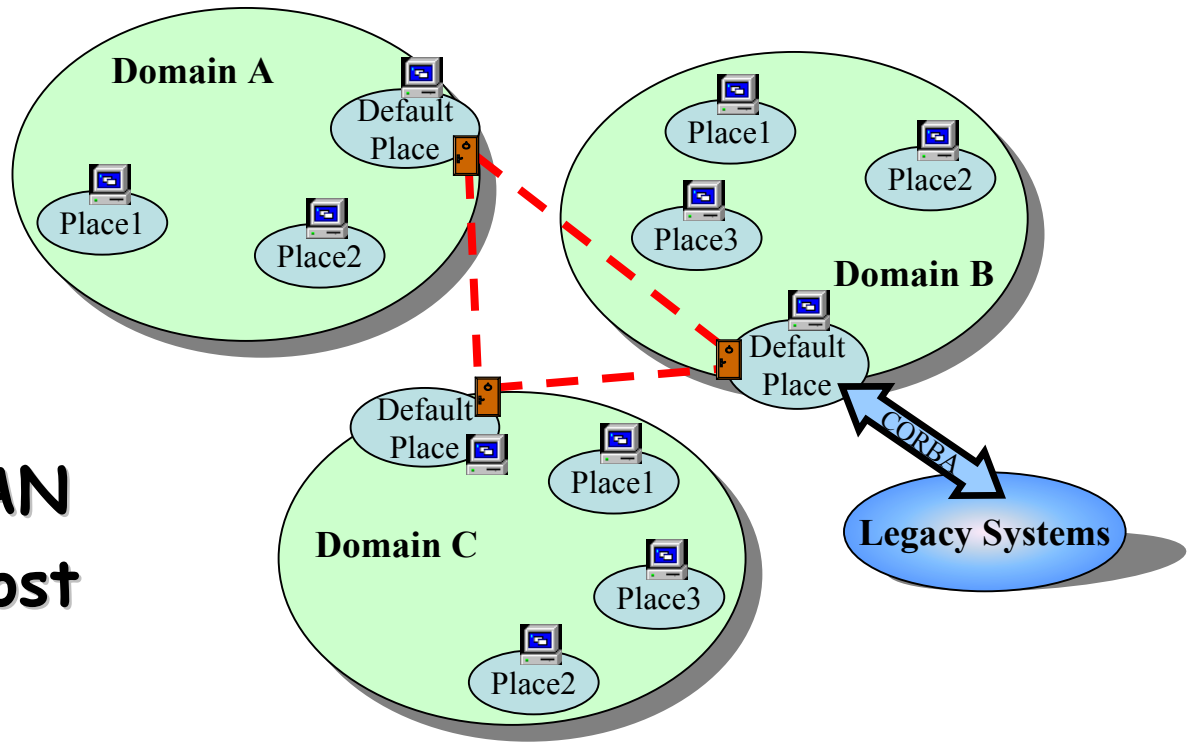
- ***Management Integrato*** di Risorse e Servizi
  - ***Monitoraggio*** delle risorse (kernel + application, on-line, Java-based)
  - ***Gestione*** **QoS** (negoziazione, controllo, riconfigurazione, adattamento)
- Supporto al ***Mobile Computing***
  - Mobilità ***utente*** (*User Virtual Environment*)
  - Mobilità del ***terminale*** (*Mobile Virtual Terminal*)
  - Mobilità delle ***risorse e componenti di servizio*** (*Virtual Resource Manager*)

Applicazioni:

- ***Accounting di servizi***
- ***Video-on-Demand***
- ***Mobile web retrieval***

Astrazioni di Località

Clustering a due livelli di località per la ***modellazione*** degli scenari di esecuzione in un sistema distribuito.



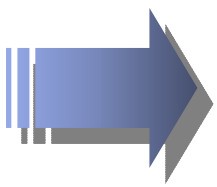
Dominio — **LAN**

Place — **Host**

Monitoraggio

Obiettivo:

una ***monitoring API uniforme (MAPI)*** basata su Java per il monitoraggio locale e distribuito



trasparenza rispetto alla piattaforma osservata

indicatori sintetici vs. indicatori dettagliati e analitici

minima interferenza

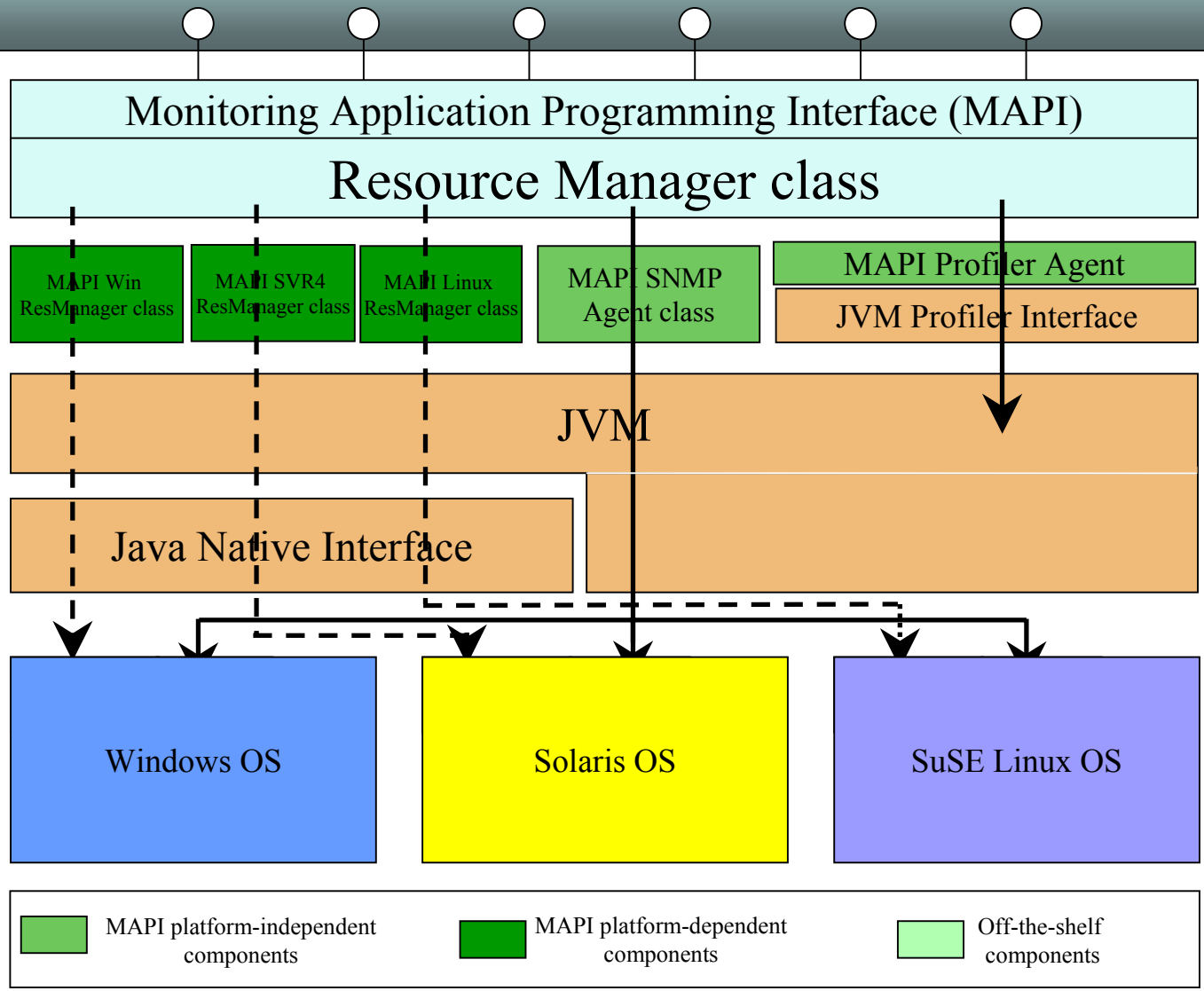
MAPI sfrutta:

1) soluzioni ***JVMPI***-based, per permettere monitoraggio di ***Java thread***

2) integrazione con ***SNMPv3*** oppure ***JNI*** & ***native monitoring*** per ***estendere visibilità di JVM*** allo stato di ***kernel*** e delle ***applicazioni esterne*** alla JVM

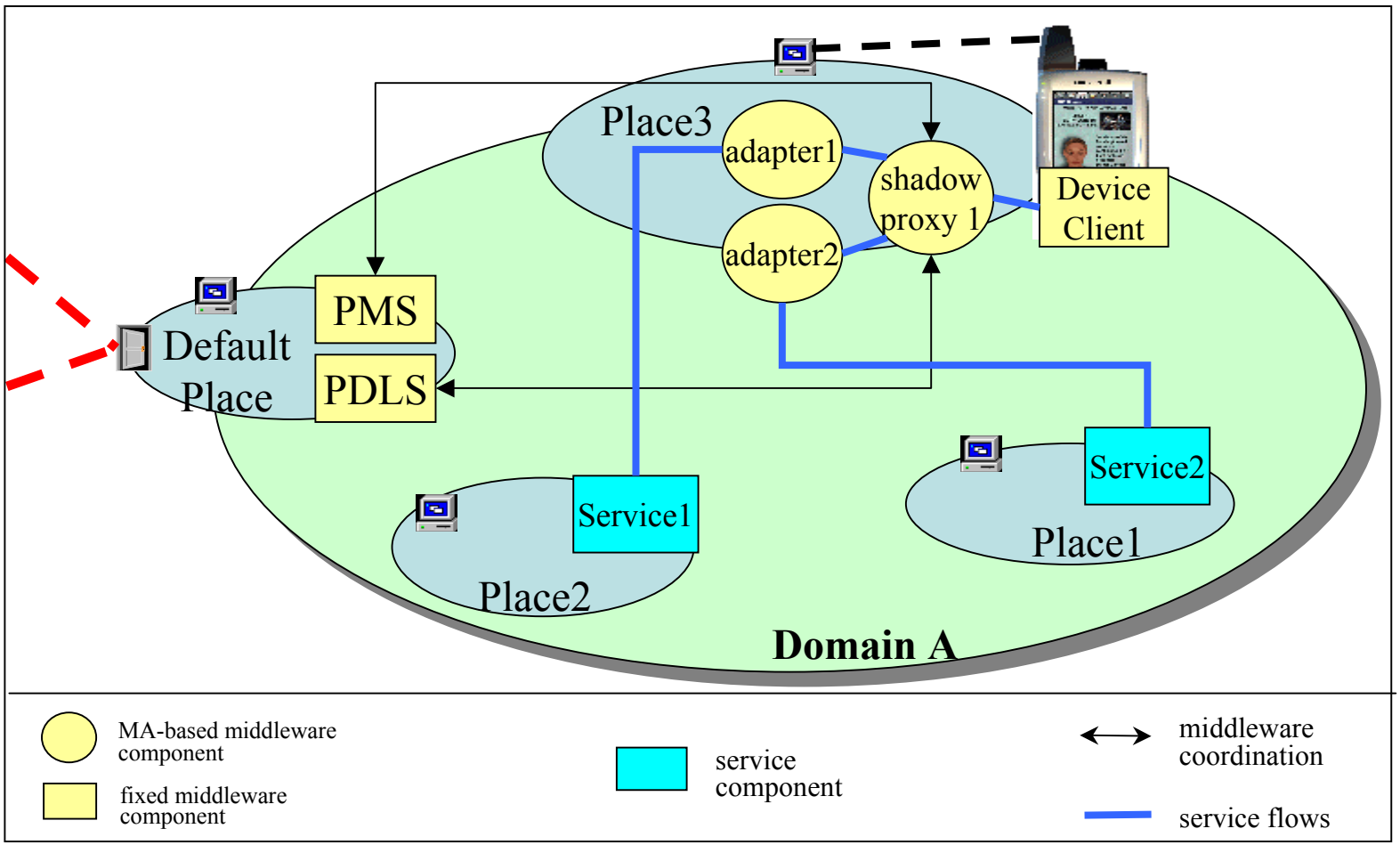
trasparenza attraverso l'integrazione con differenti librerie ***platform-specific***

Componente di monitoraggio



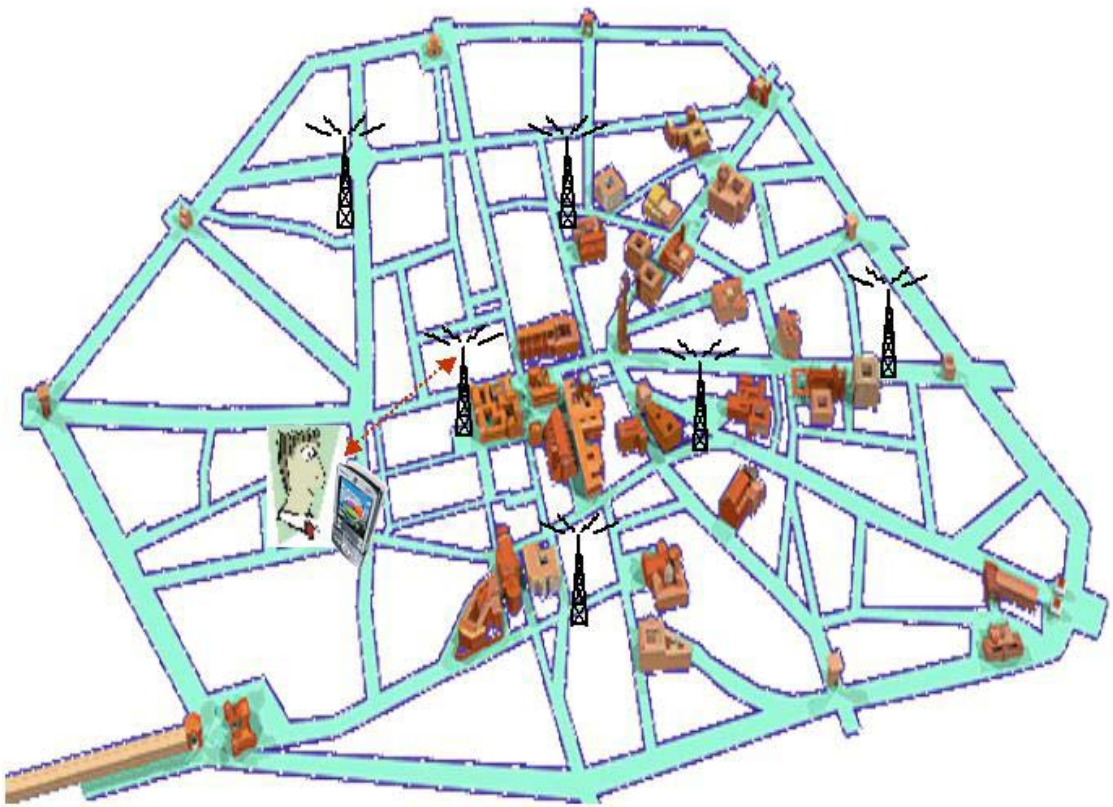
QoS Management

Adattamento dinamico dei contenuti alle capacita' del dispositivo



Case Study: Accounting di un servizio per terminali mobili

Servizio di informazioni turistiche con adattamento dei contenuti



Costi di servizio:

- **connettività wireless**

per tener conto del consumo delle risorse di comunicazione (durata dell'accesso, traffico generato)

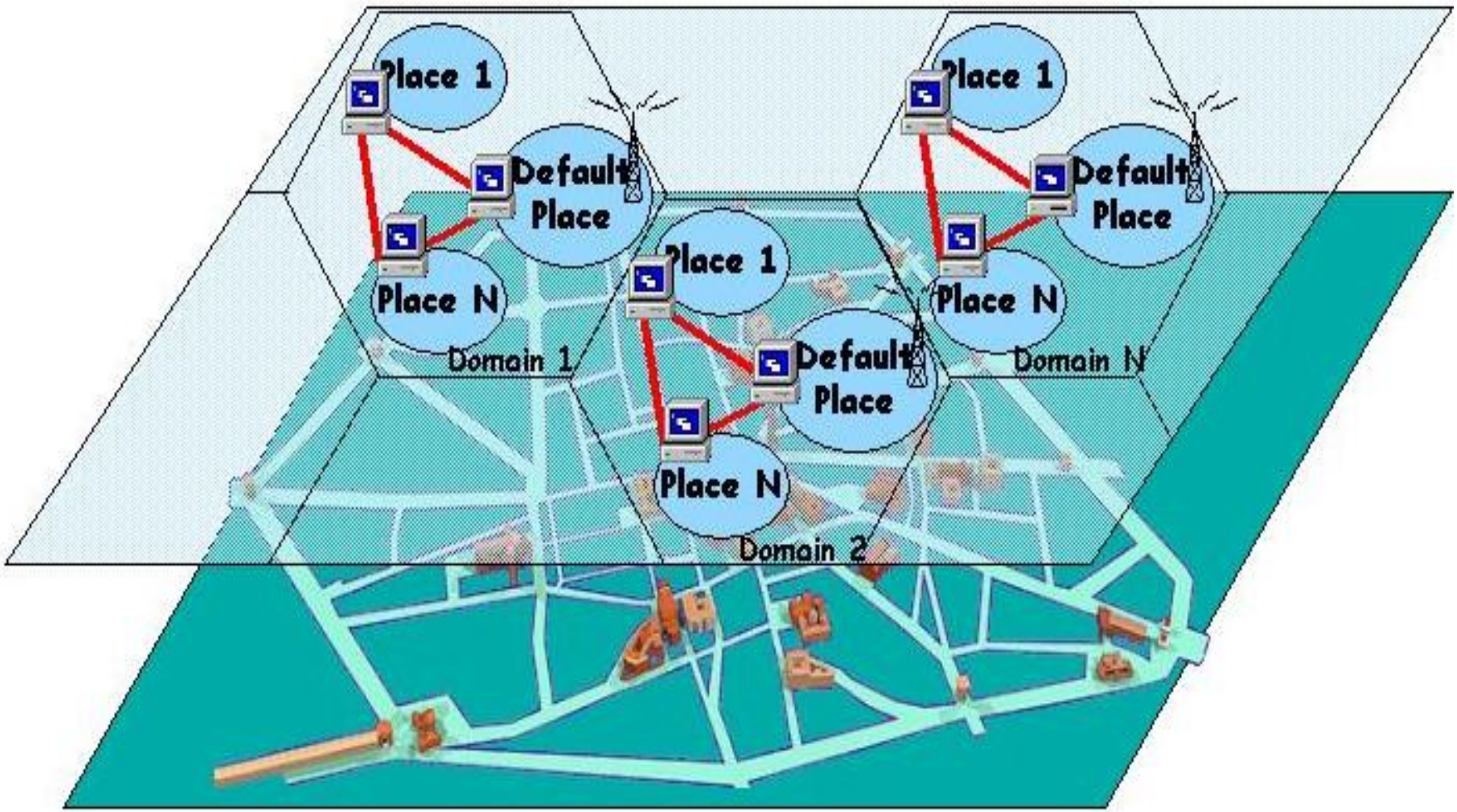
- **adattamento**

per tener conto delle risorse coinvolte nell'adattamento dinamico della QoS (CPU e memoria usate per scalare i contenuti)

- **contenuti**

… non ancora impementato! ☹

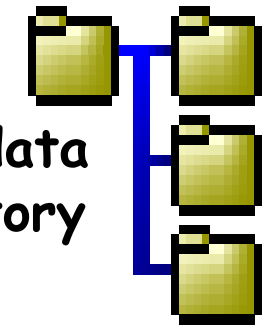
Modellazione delle località



Profili per adattamento e accounting

Device Profiles

Metadata Directory



Pricing Profiles

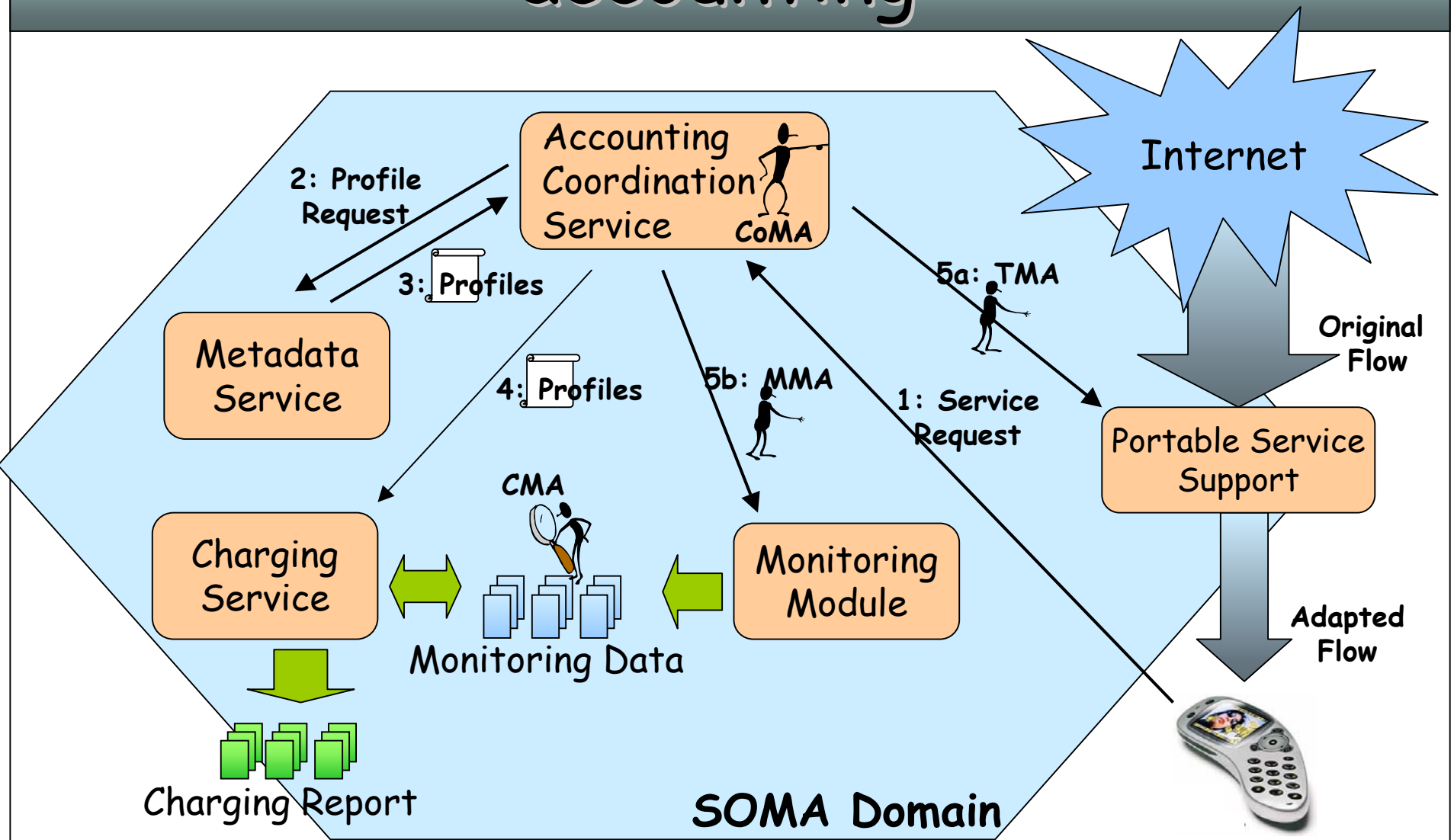
User Profiles

Service Profiles

Profile representation:

Device/User Profiles -> *Composite Capabilities/Preference Profile (CC/PP)*
Service Profiles -> *Web Services Description Language (WSDL)*
Pricing Profiles -> *Resource Description Framework (RDF)*

Interazione fra i componenti di accounting



Conclusioni

- Overview dei concetti di base sui middleware
- Middleware per applicazioni in ambiti innovativi: mobile computing, WSNs, smart environments
- SOMA
- Middleware:
  - sistemi in rapida evoluzione
  - tecnolgia strategica per passare da applicazioni prototipali ad hoc a prodotti commercialmente utilizzabili