

Reti di Calcolatori II

SNMP (parte I)

Giorgio Ventre
COMICS LAB
Dipartimento di Informatica e Sistemistica
Università di Napoli Federico II

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

Nota di Copyright

Quest'insieme di trasparenze è stato ideato e realizzato dai ricercatori del Gruppo di Ricerca sull'Informatica Distribuita del Dipartimento di Informatica e Sistemistica dell'Università di Napoli e del Laboratorio Nazionale per la Informatica e la Telematica Multimediali. Esse possono essere impiegate liberamente per fini didattici esclusivamente senza fini di lucro, a meno di un esplicito consenso scritto degli Autori. Nell'uso dovrà essere esplicitamente riportata la fonte e gli Autori. Gli Autori non sono responsabili per eventuali imprecisioni contenute in tali trasparenze né per eventuali problemi, danni o malfunzionamenti derivanti dal loro uso o applicazione.

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

Obiettivi

- | Rimettere le prove pratiche in prospettiva teorica
- | Familiarizzarsi con il linguaggio utilizzato in ambito SNMP
- | Dettagliare le particolarità del protocollo SNMP

Obiettivi di SNMP

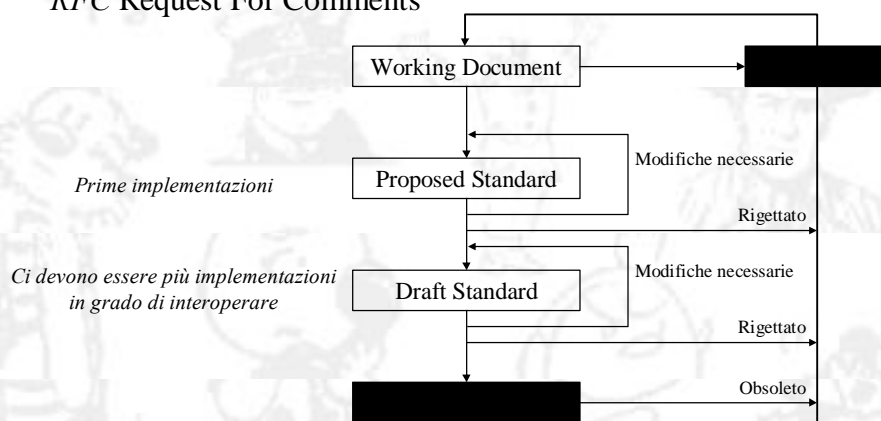
- | Semplicità (bassi costi di implementazione): numero limitato di primitive, meccanismi elementari, in particolare lato Agent
- | Ubiquità (stessa tecnologia per una stampante o un Cray)
- | Estendibilità (possibilità di definire nuove MIB)
- | Rappresentazione dei dati indipendente dalla piattaforma
- | Robustezza (scelta di UDP, connectionless protocol)
- | Indipendenza da altri servizi di rete (DNS, NIS...)
- | Standard internazionale (“vendor-independent”)

Storia

- » 1988: Simple Network Management Protocol (SNMP) - proposed
 - » 1990: Simple Network Management Protocol - standard
 - » 1991: Management Information Base II - standard
 - » 1993: SNMPv2 (Party/Context) - historical
 - » 1996: SNMPv2 (Communities) - draft/experimental
 - » 1998: SNMPv3 (User-based) - draft
-
- | SNMPv1 molto diffusa in ambito reti dati
 - | SNMPv2 ha portato alcuni miglioramenti, ma non ha raggiunto gli obiettivi e non è diventato uno standard
 - | SNMPv3 accettato, ma ancora poco usato

Processo di standardizzazione IETF

RFC Request For Comments

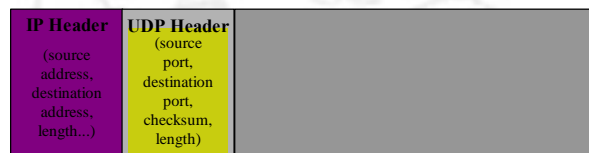


RFC relative ad SNMP

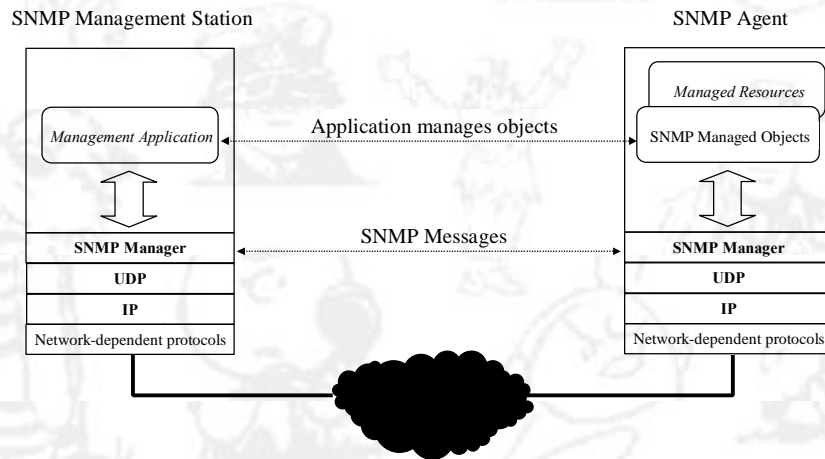
- | RFC 1155: SMI
- | RFC 1157: specifica formale di SNMPv1
- | RFC 1213: definizione MIB II
- | RFC 1215 (non standard): definizione trap
- | RFC 1441: SNMPv2
- | RFC 2271-2275: SNMPv3
- | tante altre RFC che definiscono altre MIB

Il protocollo SNMP (1)

- | Messaggi scambiati tra Agent e Manager chiamati *PDU* (Protocol Data Unit)
- | Viaggiano sul protocollo UDP
- | I dati richiesti o forniti vengono codificati secondo le BER (Basic Encoding Rules) per renderli indipendenti dalla rappresentazione interna di ciascun apparato



Il protocollo SNMP (2)



Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

SNMP PDU

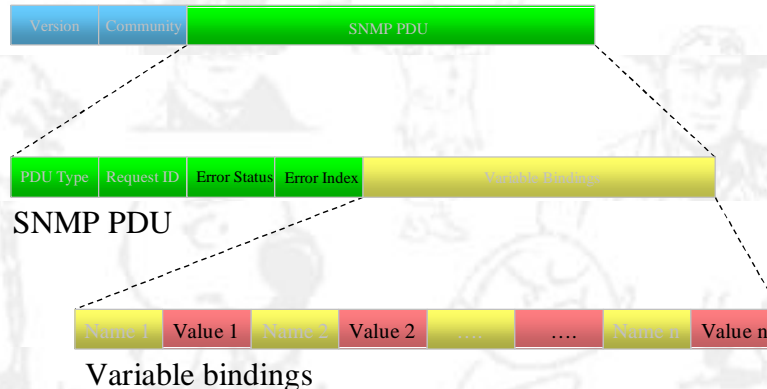
- | I tipi di messaggio si chiamano *primitive*
- | 3 primitive Manager → Agent
 - » GetRequest
 - » GetNextRequest
 - » SetRequest
- | 1 primitiva Agent → Manager
 - » Trap

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

Struttura della PDU

SNMP Message



Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

Campi della PDU (1)

- | **Version:** vale 1 per SNMPv1
- | **Community:** “password” di autenticazione in chiaro
- | **PDU Type:** non è un vero campo del messaggio, ma una “meta-informazione” di tipo ASN.1 del messaggio...
- | **Request ID:** ID univoco per ogni richiesta; permette di appaiare le risposte (in caso di perdita di sequenza)
- | **Error Status:** vale 0 nelle richieste, e nelle risposte fornisce un esito della richiesta:

noError (0)	noSuchName (2)	readOnly (4)
tooBig (1)	badValue (3)	genErr (5)

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

Campi della PDU (2)

- | **Error Index:** vale 0 nelle richieste; nelle risposte, è non-nullo in caso di errore, e dà l'indice della *prima variabile* che ha causato l'errore
- | **Variable Bindings** (*varbinds*): lista di coppie nome/valore. Nelle GetRequest e GetNextRequest il valore è nullo.
 - » è quindi possibile richiedere più variabili in contemporanea
 - » viene eseguita la stessa operazione (get, get-next, set) su tutte le variabili
 - » permette di limitare il numero di richieste da fare e gli scambi manager-agent
 - » utilizzato molto per la scansione delle tabelle...

Struttura della Trap

SNMP Message



SNMP PDU



Variable bindings

Campi della trap

- | **Enterprise:** basato sulla variabile sysObjectID, permette di individuare il tipo di apparato/sottosistema originario
- | **Agent Address:** indirizzo IP dell'agent generatore
- | **Generic Trap:** tipo di trap tra un insieme predefinito
 - coldStart (0) linkDown (2) authenticationFailure (4)
 - warmStart (1) linkUp (3) egpNeighborLoss (6)
 - enterpriseSpecific (6) [per le trap proprietarie -- vedi campo successivo]
- | **Specific Trap:** tipo di trap non standard
- | **Timestamp:** indica il momento in cui è stata inviata la trap
- | **Variable bindings:** come per le altre PDU

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

Identificazione di una variabile

- | Ogni variabile è identificata dalla sua “posizione” nell'albero di naming di una MIB.
- | L'identificativo di una variabile e' rappresentato dall'OBJECT IDENTIFIER
- | SNMPv1 permette solo la gestione di variabili scalari. Le uniche strutture dati composte permesse sono le tabelle, che vengono rappresentate come variabili indicizzate.

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

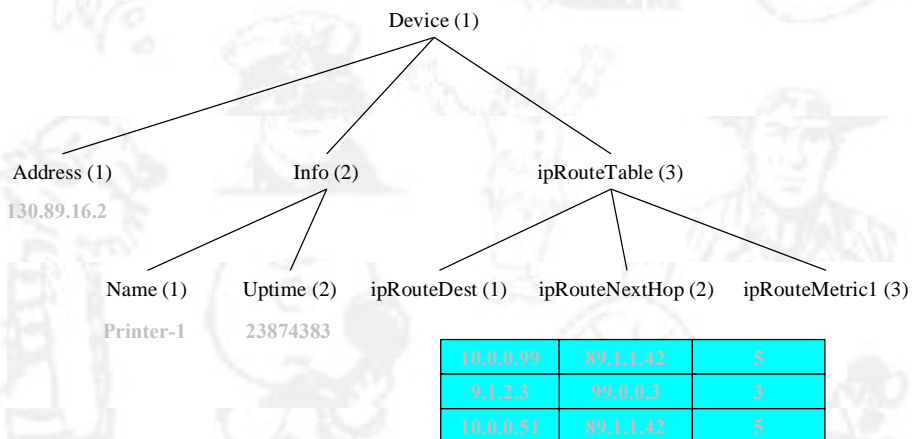
Che cos'è una MIB

- | La MIB **non è** un database...
- | E' una *definizione* dell'informazione gestita ("specificata" dell'interfaccia)
- | Si rappresenta come una lista *ordinata* di valori (vedi il meccanismo della getNextRequest)
- | L'agent non è l'entità gestita (managed device), ma il *componente software* che conosce le informazioni locali da gestire e le trasforma in forma compatibile con SNMP
- | La MIB non è l'entità gestita, ma solo la sua *modellazione*

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

MIB di Esempio



Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

Object Identifier (1)

- | E' l'identificativo univoco di un nodo dell'albero
- | Si esprime con la concatenazione degli identificativi dei nodi su tutta la gerarchia (fino alla radice)
- | In notazione simbolica, si usa il punto come separatore
- | Ad esempio: la variabile upTime ha object identifier:
1.2.2
- | Per accedere al valore della variabile, bisogna indicare che si accede ad una istanza dell'oggetto (*Instance Identifier*)
- | per questo va aggiunto uno zero finale: **1.2.2.0**
- | Per le tabelle... ne ripareremo!

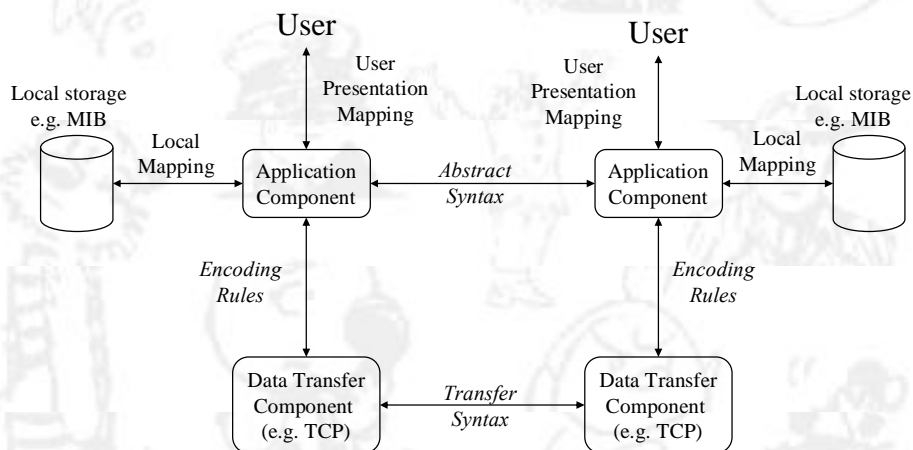
Object Identifier (2)

- | Se ogni nodo dell'albero è numerato in maniera univoca sotto ad uno stesso padre, allora ogni nodo può essere individuato univocamente
- | Si definisce una relazione di ordine di tipo lessicografico, basato sui valori numerici dell'identificativo dei nodi
- | La comparazione parte dalla radice, livello per livello
- | In questo modo:
 - » $1.2.2 > 1.1$
 - » $1.3 > 1.2.2$
 - » $1.5.10 > 1.5.2$
 - » $1.1.0 > 1.1$

SMI : obiettivi

- | Per consentire l'*interoperabilità* delle varie implementazioni agent e manager, è necessario definire senza alcuna ambiguità:
 - » la struttura della MIB (in modo che sia univoco il mapping tra un object identifier e la semantica associata)
 - » la rappresentazione binaria dei valori scambiati
- | Il primo obiettivo si raggiunge definendo un linguaggio o *notazione astratta* di descrizione della struttura della informazione gestita
- | Il secondo obiettivo si raggiunge definendo le modalità di codifica e trasmissione dell'informazione nei messaggio SNMP (BER -- Basic Encoding Rules)

Abstract Syntax/Transfer Syntax



ASN.1

- | OSI specifica gli oggetti astratti con una notazione definita nella raccomandazione X.208: *ASN.1* (Abstract Syntax Notation One)
- | ASN.1 serve a definire i *tipi astratti* e i *valori*
- | Un tipo è un *insieme di valori* (per certi tipi è un insieme *finito*, per altri è *infinito*)
- | SNMP utilizza un sottinsieme di ASN.1

Alcune Strutture ASN.1

- | **Simple Type**: un tipo semplice è definito specificando direttamente l'insieme dei suoi valori. Sono tipi atomici. Tutti gli altri tipi sono definiti in base ai tipi semplici.
- | **Structured Type**: un tipo strutturato è costituito da componenti e permette di costruire tipi di dati complessi.
- | **Macro**: notazione per estendere i tipi ASN.1 in maniera arbitraria.
- | **Module**: insieme di definizioni utilizzabili da altri moduli.

ASN.1: Modulo

```
<module reference> DEFINITIONS ::=
BEGIN
  EXPORTS -- exported definitions
  IMPORTS -- imported definitions from other modules
  AssignmentList -- value assignments and Macro definitions
END
```

Tipi ASN.1 (1)

- | I tipi ASN.1 utilizzati in SNMP sono di due classi:
 - » UNIVERSAL (definiti nella X.208):
 - INTEGER (2)
 - OCTET STRING (4)
 - NULL (5)
 - OBJECT IDENTIFIER (6)
 - SEQUENCE, SEQUENCE OF (16)
 - » APPLICATION (definite nelle RFC):
 - *IpAddress*: indirizzo a 32 bit
 - *Counter*: intero positivo (32 bit) che può solo essere incrementato
 - *Gauge*: intero positivo (32 bit) che può crescere e decrescere
 - *TimeTicks*: intero positivo che conta il tempo in centesimi di secondo
 - *Opaque*: dati binari senza formato esplicito

Tipi ASN.1 (2)

- | Non c'è tipo “enumerativo”
- | Una lista di possibili valori viene fatta con il tipo INTEGER
- | Il valore 0 non viene usato (per convenzione)
- | Si possono usare solo i valori elencati (in caso si deve quindi prevedere un valore speciale “altro”)
- | Ad esempio:

```
ifType OBJECT-TYPE SYNTAX INTEGER {  
    other(1),          -- none of the following  
    regular1822(2),  
    ...  
    ethernet-csmacd(6),  
    ...  
    fddi(15), lapb(16), sdlc(17), ds1(18), ... }
```

Notazione ASN.1 (1)

- | Alcune definizioni chiavi sono fatte nella RFC 1155, nonché la 1212 (e.g. le macros), che vengono riutilizzate (attraverso la clausola IMPORT) nelle altre definizioni di MIB

- | Definizione di nodi dell'albero:

```
internet OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) 1 }  
mgmt     OBJECT IDENTIFIER ::= { internet 2 }  
private  OBJECT IDENTIFIER ::= { internet 4 }  
ecc.
```

- | Definizione di tipi:

```
IpAddress ::= [APPLICATION 0] IMPLICIT OCTET STRING (SIZE(4))  
Counter   ::= [APPLICATION 1] IMPLICIT INTEGER (0..4294967295)
```

Notazione ASN.1 (2)

Esempio della sysDescr dalla MIB II:

```
mib-2 OBJECT IDENTIFIER ::= { mgmt 1 } -- definizione della MIB II
system OBJECT IDENTIFIER ::= { mib-2 1 }
sysDescr OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..255))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "A textual description of the entity. This value
        should include the full name and version
        identification of the system's hardware type,
        software operating-system, and networking
        software. It is mandatory that this only contain
        printable ASCII characters."
    ::= { system 1 }
```

Notazione ASN.1 (3)

- | Quando si definisce un nodo:
 - » **SYNTAX**: un tipo ASN.1 (e.g. INTEGER, DisplayString)
 - » **ACCESS**: tipo di accesso consentito (read-write, read-only, write-only, not-accessible)
 - » **STATUS**: supporto richiesto alle implementazioni (mandatory, optional, deprecated, obsolete)
 - » **DESCRIPTION**: serve a fini di documentazione

BER

- | OSI definisce la modalità di codifica binaria nella raccomandazione X.209: *BER* (Basic Encoding Rules)
- | La codifica definita dalle BER si basa sulla struttura *type-length-value* (TLV)
 - » *Type*: indica il tipo, e se l'encoding è primitivo o strutturato.
 - » *Length*: indica la lunghezza della rappresentazione
 - » *Value*: la rappresentazione del valore del tipo ASN.1 come sequenza di ottetti.
- | La struttura è ricorsiva: per ogni valore ASN.1 costituito da uno o più componenti, il campo V della sua TLV consiste, a sua volta, di una o più TLV.

Metodi di Codifica

- | **Primitive, Definite-Length Encoding**
 - » Utilizzato per tipi semplici.
- | **Constructed, Definite-Length Encoding**
 - » Utilizzato per tipi strutturati (p.e. SEQUENCE, SEQUENCE-OF,...)
- | **Constructed, Indefinite-Length Encoding**
 - » Utilizzato per dati strutturati o stringhe la cui lunghezza non è nota a priori.

Esempi di Codifica

ASN.1		Esadecimale
INTEGER	<i>Primitive, Defined Length</i>	
Valore: 123	→	02 01 7B
SEQUENCE (INTEGER, INTEGER)	<i>Constructed, Defined Length</i>	
Valore: (3,42)	→	10 06 02 01 03 02 01 2A
DisplayString	<i>Constructed, Undefined Length</i>	
Valore: "Test User 1"	→	13 80 54 65 73 74 20 55 73 65 72 20 31 00 00

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

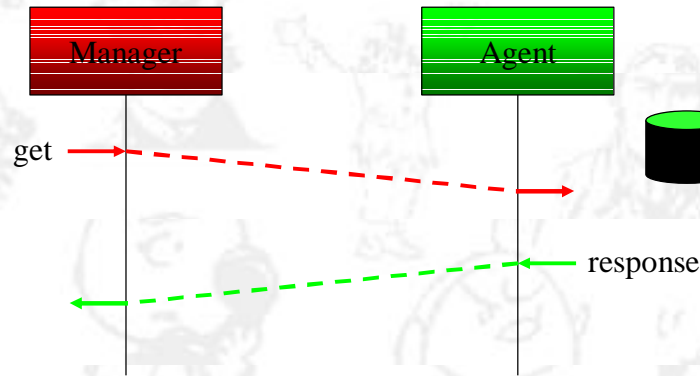
Object Identifier Revisited

- | Esiste una notazione alternativa per l'identificazione dei nodi: la notazione *simbolica*
- | Si basa sull'identificazione simbolica dei nodi nelle definizioni ASN.1 dei moduli di MIB dell'SMI
- | è possibile:
 - » sostituire ogni numero di nodo con la notazione simbolica corrispondente: *iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0*
 - » fare a meno di tutto il *path* dalla radice e scrivere semplicemente: *sysDescr.0*
- | Attenzione che non è quello che viene trasmesso in rete (bensì la codifica fatta secondo le Basic Encoding Rules...)

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

SNMP GetRequest (1)



Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

SNMP GetRequest (2)

- | Permette di richiedere il valore di una o più variabili di MIB
- | Possibili errori:
 - » *noSuchName*: l'oggetto non esiste o non è una foglia dell'albero
 - » *tooBig*: la risposta non sta nella PDU di risposta
 - » *genErr*: errore generico, tutti gli altri casi

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

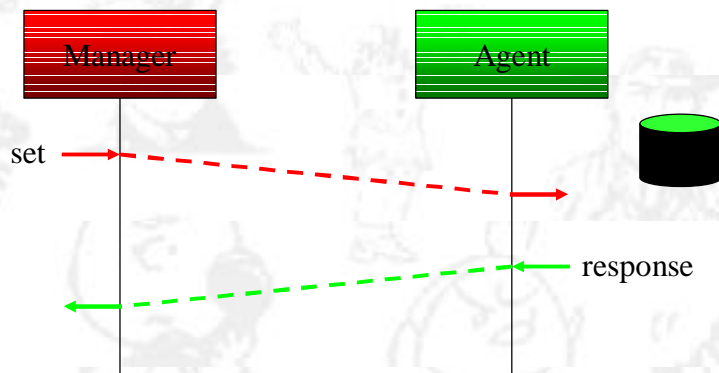
Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

SNMP GetRequest (3)

Esempi:

- » get(1.1.0)
⇒ response (1.1.0 → 130.89.16.2)
- » get(1.2.0)
⇒ response (Error-Status = noSuchName)
- » get(1.1)
⇒ response(Error-Status = noSuchName)
- » get(1.1.0; 1.2.2.0)
⇒ response(1.1.0 → 130.89.16.2; 1.2.2.0 → 23874383)

SNMP SetRequest (1)



SNMP SetRequest (2)

- | Per assegnare un valore a istanze di variabili già esistenti, o per comandare azioni
- | Possibili errori:
 - » *noSuchName*: l'oggetto non esiste o non è una foglia dell'albero
 - » *badValue*: valore fornito inadeguato per il tipo
 - » *tooBig*: la risposta non sta nella PDU di risposta
 - » *genErr*: errore generico, tutti gli altri casi

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

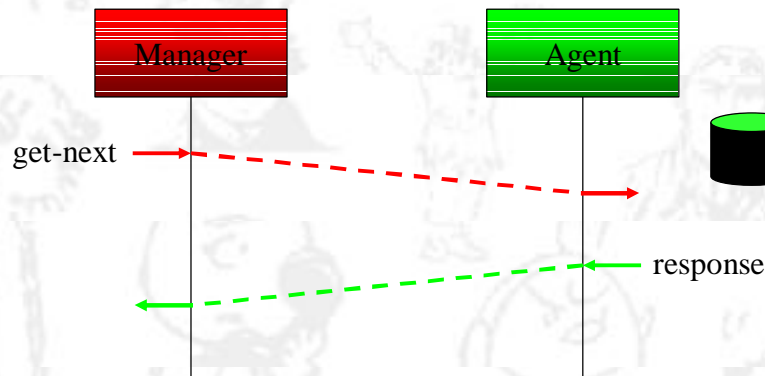
SNMP SetRequest (3)

- | Esempi:
 - » `set(1.2.1.0 → my-printer)`
⇒ `response(noError, 1.2.1.0 → my-printer)`
 - » `set(1.2.1.0 → my-printer, 1.2.2.0 → 0)`
⇒ `response(noSuchName, error-index=2)`
- UpTime non e' scrivibile...*

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

SNMP GetNextRequest (1)



Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

SNMP GetNextRequest (2)

- Restituisce il nome e il valore della **successiva** istanza di variabile del MIB; è usata per accedere alla struttura del MIB, e per navigare tabelle.
- Possibili errori:
 - » **noSuchName**: il MIB è stato scandito interamente
 - » **tooBig**: la risposta non sta nella PDU di risposta
 - » **genErr**: errore generico, tutti gli altri casi

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

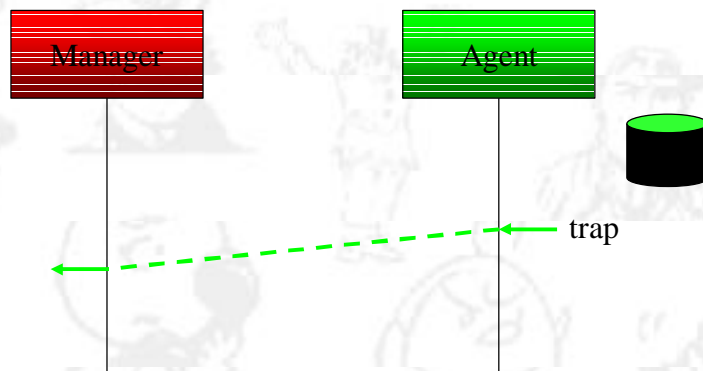
Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

SNMP GetNextRequest (3)

Esempi:

- » get-next(1.1.0)
⇒ response (1.2.1.0 → printer-1)
- » get-next(1.2.1.0)
⇒ response (1.2.2.0 → 23874383)
- » get-next(1)
⇒ response(1.1.0 → 130.89.16.2)
- » get-next(1.4)
⇒ response(Error-Status = noSuchName)

SNMP Trap (1)

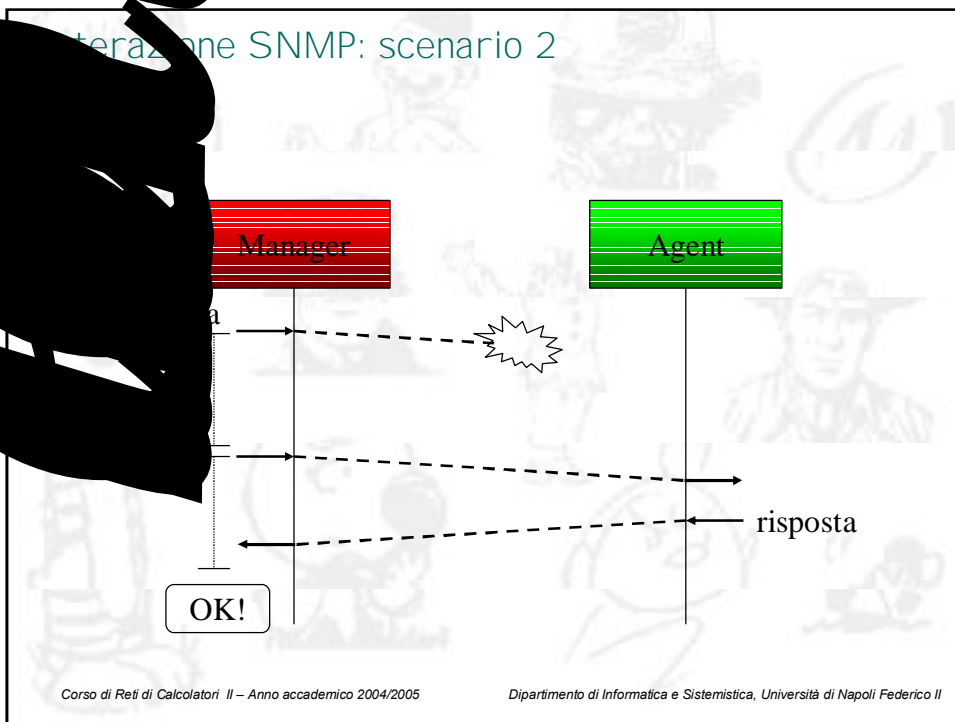
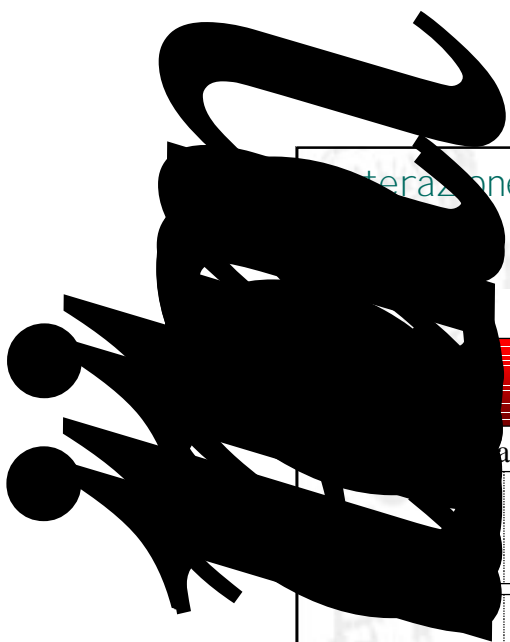
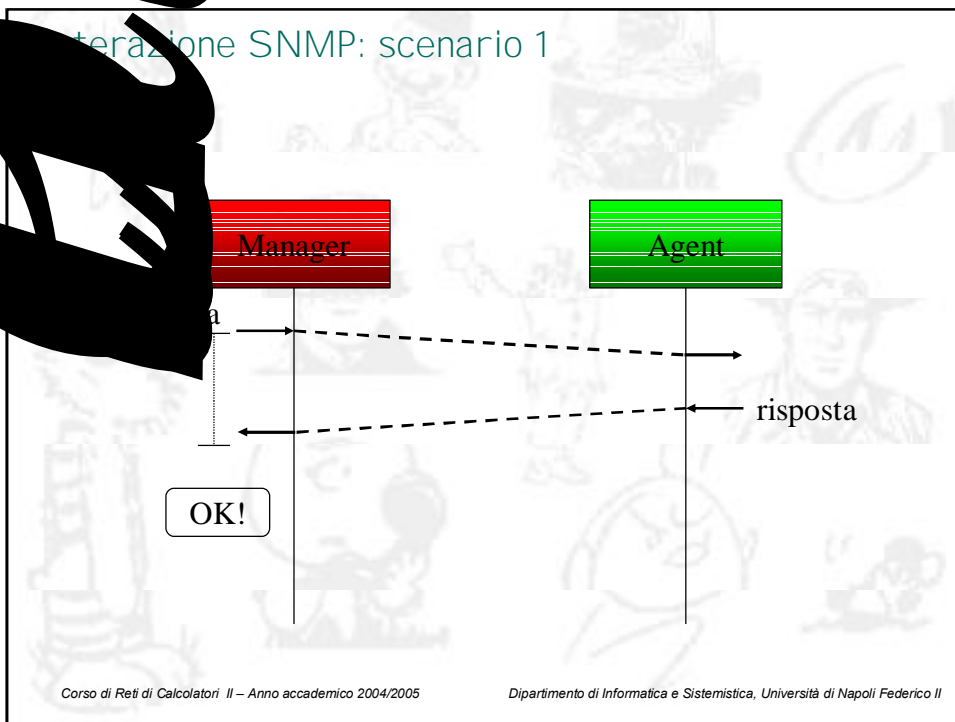


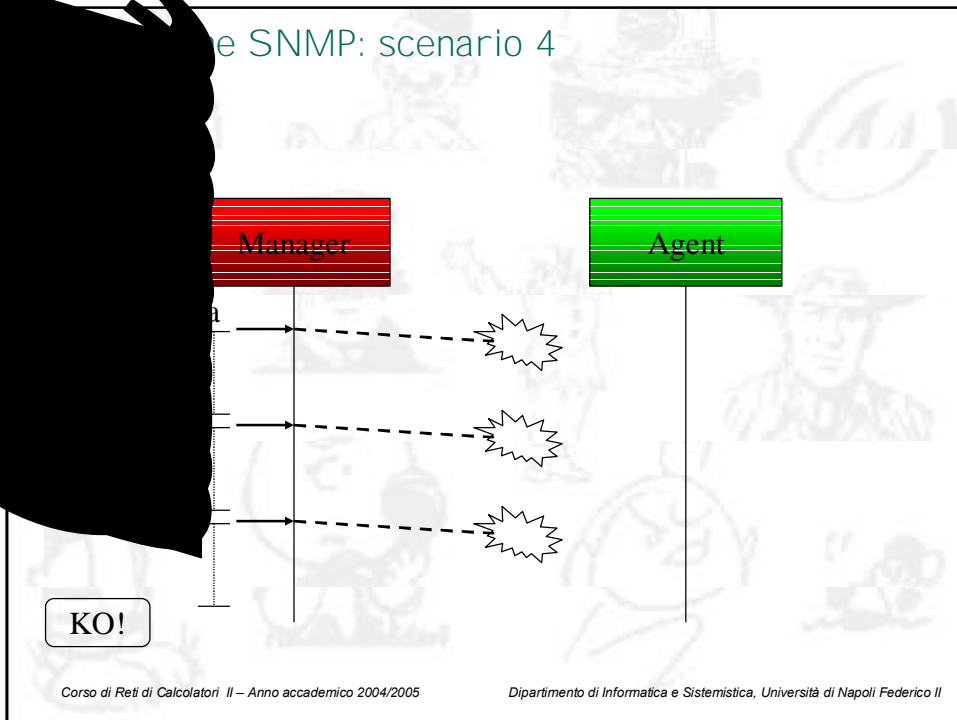
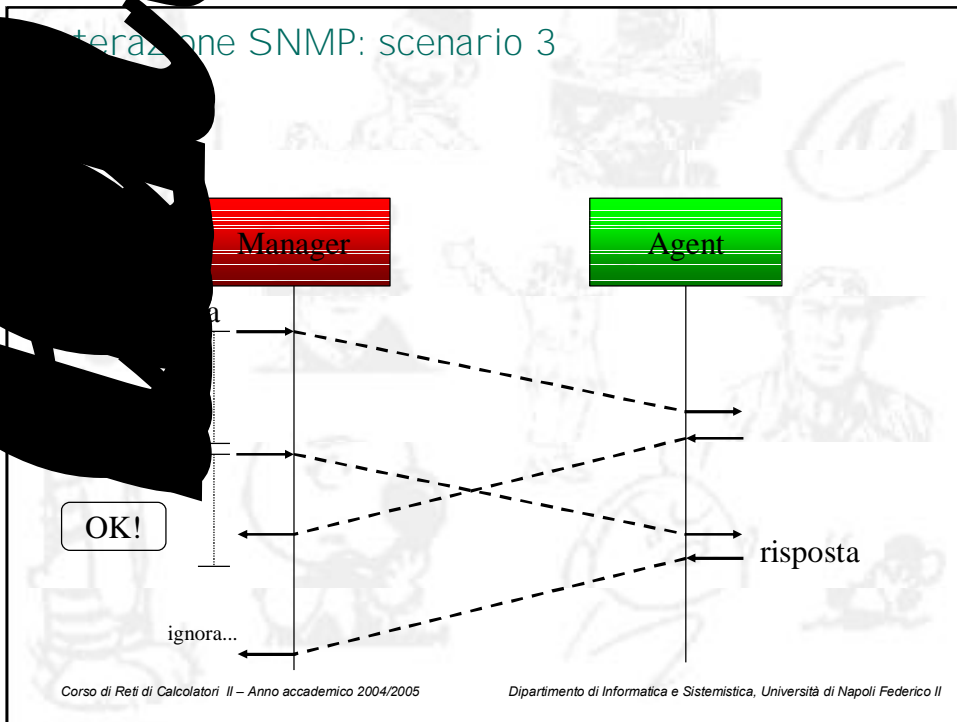
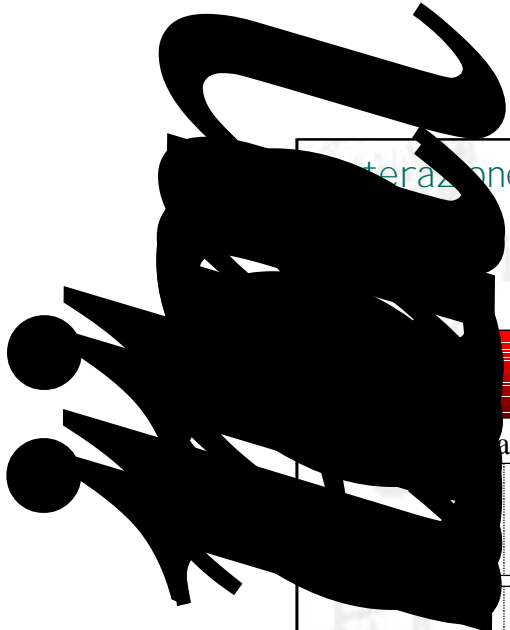
SNMP Trap (2)

- | Segnala un evento.
- | La ricezione di una trap non è confermata (*unreliable*).
- | Gli Agent possono essere configurati in modo da:
 - » non emettere traps
 - » emettere traps solo verso manager specifici

Impatto dell'uso di UDP

- | UDP è *connectionless* e *unreliable* (non affidabile): l'invio di un messaggio non significa la garanzia della sua consegna
- | ⇒ fatta una richiesta SNMP, la risposta arriva in maniera asincrona, senza garanzia
- | Vanno messi in piedi meccanismi di *timeout* e di *retry* per simulare una interazione affidabile (quello che fa TCP)
- | Ma rimane a carico dello strato SNMP (più leggero di TCP)
- | Il fatto di rimanere su UDP dà *robustezza* al meccanismo (in particolare in condizioni critiche, quali sono quelle di una rete in situazione di "crisi")





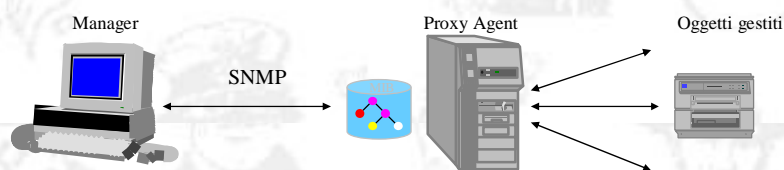
Interazione SNMP: parametri

- | **Timeout: tempo trascorso prima di ritentare**
 - » non deve essere troppo lungo, per consentire un rinvio rapido in caso di non risposta, ne troppo corto per evitare rinvii inutili (vedi scenario 3)
- | **Retry count: numero di tentativi prima di dichiarare fallita la connessione**
 - » non deve essere troppo basso, per dare un minimo di affidabilità, ne troppo alto, per non sovraccaricare la rete di richieste

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

Proxy Management



- | “*proxy*” in inglese = nel nome di qualcuno
- | un proxy agent è un agent SNMP di cui la MIB non rappresenta informazioni di gestione dell'apparato su cui gira
- | traduce un protocollo proprietario in SNMP
- | aggrega le informazioni di basso livello in informazioni di più alto livello

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

Meccanismi di monitoraggio

- | Un manager SNMP monitora lo stato degli apparati attraverso una interrogazione periodica di opportune variabili di stato (*polling*)
- | *Trade-off* sulla scelta della periodicità e delle variabili interrogate (traffico generato in rete vs. qualità dell'info.)
- | Le trap SNMP vengono in aiuto (in maniera *unreliable*) in quanto segnalano i casi eccezionali nel momento in cui accadono
- | *Trap-directed polling*: il manager adegua la sua politica di polling su ricezione di trap (e.g. interrogazione completa + aumento frequenza di polling, ecc.)

SNMPv1: Sicurezza (1)

- | La gestione della sicurezza da parte del protocollo SNMPv1 è estremamente elementare. Approcci più adeguati sono definiti per le versioni successive: SNMPv2 e SNMPv3.
- | Il concetto fondamentale è quello di *community*, a cui sono associati dei diritti di accesso *READ-ONLY* o *READ-WRITE*.
- | La gestione delle *community* non viene specificata dagli standard ed è quindi lasciata all'implementazione. Lo standard specifica solo che la *community* è una stringa che va trasmessa con il messaggio SNMP.
- | La *community* viaggia in chiaro sulla rete...

SNMPv1: Sicurezza (2)

Per limitare i rischi, alcune misure:

- | utilizzare password non banali, e cambiare spesso
- | configurare l'Agent ad inviare le trap di *authenticationFailure*
- | configurare la rete (con filtri o firewalls) per impedire il traffico SNMP al di fuori dai percorsi Manager-Agent

Nella realtà, gli apparati forniscono anche meccanismi di ACL (Access Control List), dove vanno inseriti gli indirizzi dei manager di fiducia.

Conclusione (1)

- | SNMPv1 è un protocollo semplice, basato sul polling (di variabili critiche) e che modella gli oggetti gestiti in termini di variabili scalari o tabelle.
- | La sua semplicità è stata il fattore principale del suo successo. Tuttavia, a lungo andare si è anche rivelata l'inconveniente principale nell'estensione dell'utilizzo di SNMPv1 a realtà diverse dalle LAN.

Conclusione (2)

- | Le limitazioni principali di SNMP si possono identificare in:
 - » Gestione della sicurezza insufficiente
 - » Strutture dati troppo elementari per modellare in maniera semplice realtà complesse
 - » Numero di eventi asincroni definiti troppo ridotto
 - » Primitive insufficienti per garantire prestazioni adeguate
- | Queste limitazioni hanno portato alla specifica di SNMPv2 e SNMPv3