Un MIB SNMP per la gestione di client peer-to-peer

Fabio Dianda
dianda@sec.di.unipi.it

January 26, 2002

0.1 Breve introduzione al p2p

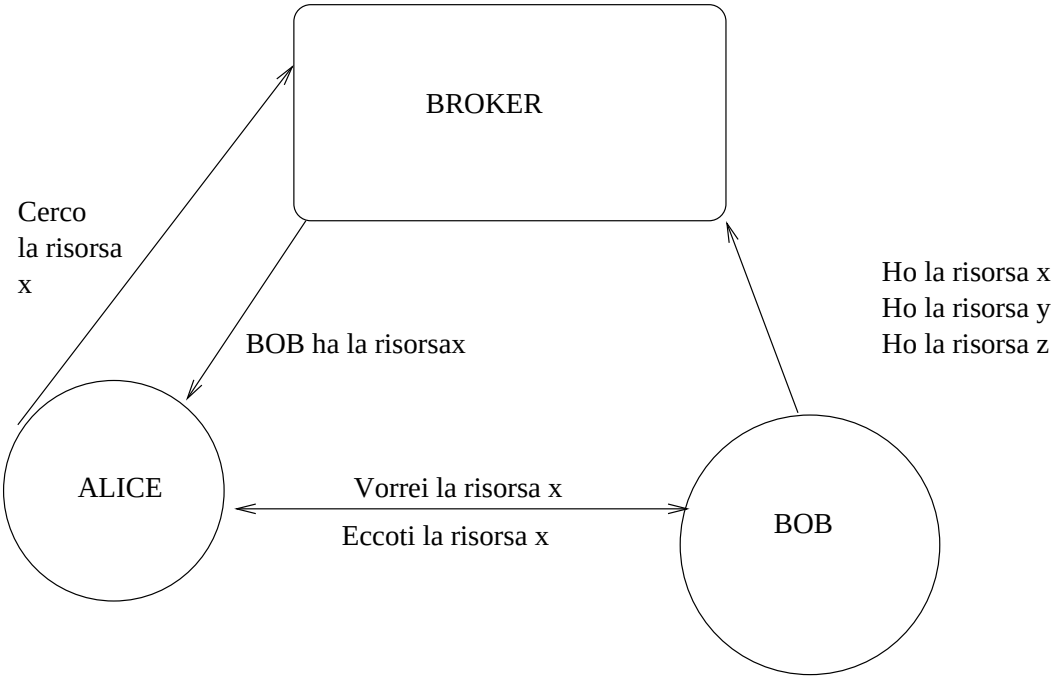
Il paradigma di comunicazione peer-to-peer si sta imponendo come una delle tecnologie più interessanti in questo inizio di secolo nell' ambito delle reti di comunicazioni, ma la sua origine non risale agli ultimi tempi: già i primi nodi della rete ARPANET sfruttavano questo meccanismo, in quanto ognuno era "pari" dell' altro, infatti bisogna ricordare che uno degli scopi esplicitamente richiesti dal dipartimento della difesa americano (DOD) era quello di poter comunicare con un nodo indipendentemente dalla situazione degli altri (es: bombardamento, attacco militare ecc.). Successivamente, con l' esplosione commerciale della rete Internet, questo modello di comunicazione e' stato stato sostituito dal modello client-server che si e' imposto in virtù di una certa semplicità: il client invia richieste ad un server che risponde dopo aver effettuato eventualmente una serie di elaborazioni. Gli unici cambiamenti degni di nota da segnalare nel corso degli ultimi anni sono stati l' avvento di un unico programma (browser) in grado di fare le funzioni di più client, e l' introduzione di elaboratori sempre più potenti sul lato server (cluster in particolare).

Il ritorno alla ribalta del p2p viene generalmente indicato con l' avvento, a partire dal gennaio 1999, di Napster, un semplice programma in grado di facilitare lo scambio di brani musicali appartenenti ad una comunità di utenti, questo suo scopo ha pero' provocato la reazione delle industrie discografiche che non hanno tardato a citare in giudizio l' inventore del software, lo studente Shawn Fleming e provocare la chiusura dell' azienda da lui fondata per "gestire" la comunità; il posto di Napster e' pero' stato preso da altri programmi, in particolare Freeheaven e Freenet, avendo come scopo anche quello di garantire il massimo anonimato, hanno portato all' equazione *(SBAGLIATA)* $P2P = illegale$.

In realtà il p2p e' uno strumento molto interessante nel campo delle applicazioni distribuite perché permette una migliore scalabalità e disponibilità di servizio rispetto al modello client server, in caso di potenziamento dei servizi offerti il costo per quanto riguarda il server e' infatti notevole (elaboratori paralleli, dischi ridondanti, connessioni in fibra ottica da più ISP e relativa gestione del routing in modo ottimale (AS number, NAT/PAT)ecc.) ; con l' aumento dell' attenzione da parte dei produttori i client p2p hanno iniziato a "scambiarsi" , mettere a disposizione risorse diverse come ad esempio spazi su disco e cicli di CPU, arrivando a formare talvolta un unico grande elaboratore distribuito per i nodi che partecipano alla comunicazione. Lo schema di funzione dello scambio tra peer e' molto semplice: un nodo produttore fornisce una risorsa, un altro nodo (consumatore) richiederà la risorsa direttamente ad esso, per localizzarla sono possibili due approcci, o si fa' uso di un nodo centrale (broker) che indica chi in quel momento ha una copia della risorsa cercata (modello centralizzato), oppure si interrogano gli stessi peer che si preoccuperanno di trasmettere ad altri peer una richiesta che non sono in grado di soddisfare (modello decentralizzato).

Principali modelli di client p2p [1]

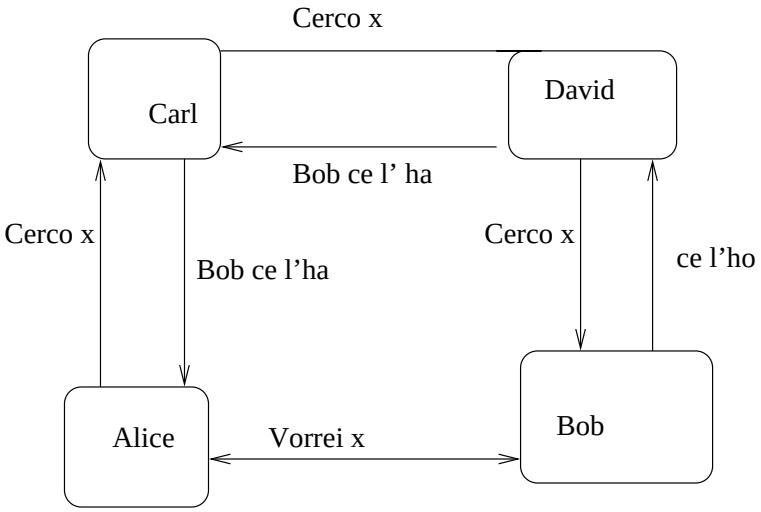
Modello Broker mediated content sharing (es: Napster)



Questi modello sfrutta la presenza di un nodo centrale a cui i client fanno richieste per individuare i peer in possesso della risorsa cercata, si tratta quindi di un modello ibrido: la comunicazione tra broker e alice e' sostanzialmente client/server ma quella tra alice e bob e' peer to peer. La presenza del broker ha da un lato il vantaggio di permettere tempi di ricerca più efficienti dall' altro costituisce un "single point of failure" che nel management di una rete risulta un punto debole (in particolare nella gestione della sicurezza). Restano comunque innegabili i vantaggi sulla qualità (velocità ed accuratezza) delle risposte, mentre alcune riflessioni meriterebbe la fase di autenticazione tra peer e broker al momento della segnalazione di una risorsa ,un falso peer potrebbe segnalare risorse che non ha per causare problemi agli altri utenti.

Modello pure peer-to-peer content sharing

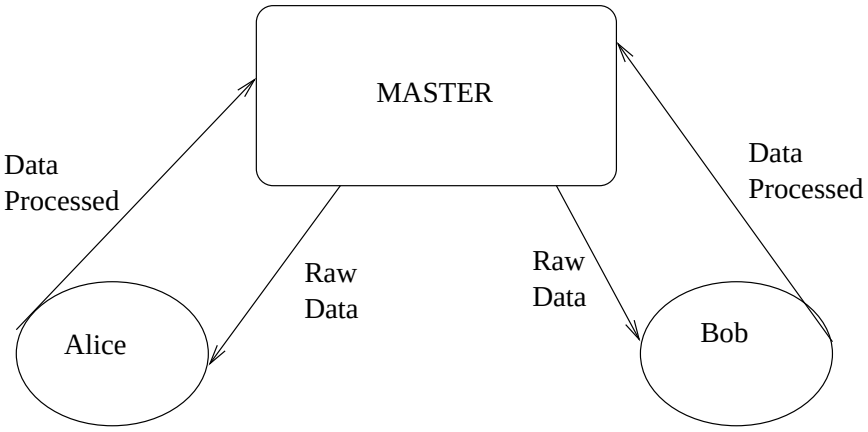
[1] Questa schematizzazione e' incompleta, serve per inquadrare meglio il caso studio



In ogni nodo di questo modello sono memorizzare esclusivamente informazioni circa le risorse locali, la mancanza di un indice globale può portare a risposte approssimative, ognuno deve segnalare al proprio vicino la disponibilità di una certa risorsa. la ricerca inoltre provoca una sorta di "flooding" perché sfrutta la catena dei peer per localizzare quanto cercato, con inevitabili tempi di attesa piuttosto lunghi. Una soluzione a questo problema e' ad esempio quella fornita dal client KaZaA, in cui non tutti i nodi partecipano alla ricerca.

Un altro esempio interessante e' il client Freenet, in cui la fase di ricerca provoca l' instaurazione di una sorta di circuito virtuale tra peer, composto cioè dal peer che ricevono la richiesta, ma non avendola, la ritramsmettono ad un loro vicino, una volta individuata questa arriverà al mittente seguendo il cammino inverso, in questo modo e' possibile fornire una dose di anonimato, i nodi infatti non sanno chi sarà l' effettivo consumatore della richiesta che stanno soddisfando, sia in caso di ritrasmissione della richiesta sia in caso copia della risorsa.

Master Slave cycle sharing



Questo e' uno dei modelli usati da più anni, e' sfruttato ad esempio per l' elaborazione dei dati provenienti dalla sonda SETI (progetto SETI@HOME) e testare la "sicurezza" di alcuni cifrari di uso comune (progetti del sito distributed.net), in questo caso i peer mettono a disposizione risorse fisiche dei loro elaboratori per elaborare i dati forniti dal master, al termine dell' operazione, che può essere eseguita in momenti di inattività delle macchine, i dati vengono rispediti al master. Si tratta di un modello particolare in cui il master ha un compito fondamentale ed abbastanza complesso, resta a suo carico il bilanciamento dei lavori e una forma di ridondanza qualora un nodo non consegni un lavoro "in tempo" ; i risultati concreti di questo modello pero' si sono dimostrati affidabili ed hanno portato a risultati impensabili da ottenere con un unico elaboratore.

0.2 Considerazioni sulle richieste del caso studio

Nelle richieste si parla di un generico client, cercherò di essere il più generico possibile e di individuare, magari a parole, alcuni aspetti particolari dei modelli sopra esposti; prima di iniziare un' analisi delle caratteristiche da monitorare vorrei indicare alcune questioni particolarmente interessanti che non hanno ancora una risposta ben definita. Innanzitutto la sicurezza in un sistema p2p e' una cosa abbastanza diversa da quella in un sistema client-server, infatti un security manager, per essere rigoroso, dovrebbe considerare "untrusted" ogni peer, ma avrebbe difficoltà a monitorare le attività "security-sensitive" in quanto molte delle tecnologie fin qui usate , come ad esempio firewall e SSL, sono di utilità relativa. Nel caso del firewall infatti la porta su cui avviene la comunicazione tra peer dovrebbe essere lasciata aperta ed eventualmente "filtrata" con un apposito modulo, che dovrebbe essere scritto appositamente per il protocollo p2p in questione (protocollo a livello application ma raramente standardizzato IETF), inoltre come indicato in [1] i produttori di firewall sono combattuti dall' eterno dilemma complessità del controllo-degrado di prestazioni quindi appare abbastanza difficile trovare un prodotto in grado di fornire funzionalità più elaborate, che non siano una semplice statistica delle porte in uso, per un certo client.

Nel caso del protocollo SSL bisogna notare che la potenza computazionale richiesta per una sessione e' abbastanza notevole per un PC (architettura utilizzata maggiormente

dai peer), nel client server erano possibile l' installazione di schede apposite per gestire tali connessioni (SSL accelerator) ma appare francamente improbabile che gli utenti si dotino di tale schede visto il loro costo non certo irrisorio; inoltre bisognerebbe riconsiderare il ruolo dei certificati, infatti se prima essi autenticavano un server ai client, ora dovrebbero autenticare i peer, ma le successive relazioni di "trust" di difficile gestione, inoltre l' uso di un certificato per ogni peer appare improponibile, vista anche la scarsa diffusione sul lato client, e un indebolimento delle caratteristiche di anonimato molto ricercate dagli utenti di alcuni client (ad esempio Freenet).

Un' ultima considerazione interessante si può dedurre da [2], il p2p non e' un fenomeno indipendente, le applicazioni distribuite stanno vedendo l' introduzione di nuove tecnologie come web services e grid computing, tutte volte a favorire l' interscambio di risorse tramite architetture SOA (Service Oriented Architecture) in cui al centro della comunicazione c'è un servizio fornito da un entità per mezzo di una rete, gli utilizzatori del servizio possono essere entità dello stesso tipo (come nel p2p) o di tipo diverso (come nei web-services) che possono eventualmente utilizzare un broker per individuare le risorse; tutto questo dovrebbe portare vantaggi agli sviluppatori di applicazioni in termini di tempo di sviluppo soprattutto, mentre non sono convinto che il lavoro dei network manager (security in particolare) verrà favorito nello stesso modo, visto che molte di queste tecnologie veicolano i loro messaggi su protocolli già esistenti (es: SOAP può usare HTTP,FTP,SMTP) rendendo complesso un controllo effettivo del payload della comunicazione.

Affrontiamo adesso il caso studio, per implementare un MIB che riesca a monitorare le connessioni di un client p2p, e' molto importante che ogni nodo debba avere una comunicazione tra agenti e manager quanto più leggera possibile, inoltre una delle richieste e' quella di integrare l' implementazione nel client stesso; il network manager dovrebbe compiere una stima a priori del numero delle connessioni che un client e' in grado di sopportare, tale stima dovrebbe considerare da un lato le risorse fisiche (HW, banda disponibile ecc.) dall' altro dovrebbe considerare quali tipo di richieste possono essere ricevute , solo copia di un contenuto o anche ricerca eventualmente da ritrasmettere; per semplicità consideriamo solo client che utilizzino il modello broker content sharing (es:Napster) che quindi non prevedono la gestione di messaggi di ricerca; stabiliamo quindi una soglia (*maxConnessioniIn*) per indicare il numero massimo di peer che possono essere serviti in un dato momento, di conseguenza e' necessaria una istruzione di trap (t*rapConnessioniIn)* che il manager riceve se questa soglie viene superata.

Accettata una connessione, il peer se ha effettivamente la disponibilità della risorsa, dovrebbe stabilire una connessione verso il richiedente, questa fase può effettivamente essere più pesante dal punto di vista della banda occupata, quindi, sempre dopo una stima delle risorse disponibili e' bene introdurre una soglia *(maxConnessioniOut)* per indicare un numero massimo di connessioni in uscita, da notare come misurare la banda disponibile non sia un problema facile, in questo caso non si può far altro che adottare un approccio empirico, facendo un monitor con strumenti come MRTG o NTOP sulla console del manager. Da notare che il problema del traffico generato dai client p2p, o meglio dall' uso che gli utenti, specie in ambito accademico ne fanno, e' fonte di un acceso dibattito tra i network manager, come testimoniato dal traffico di questo mese su [4]

Per rendere più chiara la struttura del MIB consideriamo due tipi di strutture, una per le connessioni in entrata (richieste e risorse in entrata) l' altra per le connessioni in uscita (copia delle risorse richieste), in questo modo sono facilitate le successive operazioni per calcolare le statistiche richieste; una soluzione alternativa, ma più elaborata da gestire poteva essere quella di mantenere le informazioni in un unica tabella, con un eventuale campo per differenziare le connessioni in entrata da quelle in uscita.

Variabili considerate:

numConnessioniIn: un contatore a 32 bit per indicare il numero corrente delle connessioni di richiesta di una risorsa

numConnessioniOut: un contatore a 32 bit per indicare il numero delle connessioni attive in un dato momento

byteInviati: numero di byte inviati, attraverso le connessioni di out, fino ad un certo istante

byteRicevuti: numero di byte ricevuti, attraverso attraverso le connessioni di out, fino ad un certo istante

tabConnessioniIn: una tabella per fotografare la situazione delle connessioni in corso, contiene il numConnessioni e informazioni sui peer connessi (indirizzo IP)

tabConnessioniOut: una tabella per fotografare la situazione delle connessioni in corso, contiene il numConnessioni e informazioni sui peer connessi (indirizzo IP)

trapConnessioniIn: evento che notifica ad un manager un superamento della soglia delle connessioni in entrata

trapConnessioniOut: evento che notifica ad un manager il superamento del numero massimo di connessioni di uscita attive in un dato momento

Soglie definite Per mantenere una certa qualità di servizio, e' doveroso stimare a priori alcuni parametri, superati i quali gli agent notifichino un evento al manager con una istruzione di trap apposita

maxNumConnessioniIn: il numero massimo di connessioni attive che si stima possano essere sostenute dal client senza degradare troppo le prestazioni

maxConnessioniOut: il numero massimo di connessioni in uscita che si stima possano essere gestite senza un degrado di prestazioni

Trap notificate *trapConnessioniIn* : questa notifica viene segnalata al manager nel momento in cui il numero delle connessioni in uscita superano la soglia stabilita, si noti che il numero delle connessioni in entrata misura anche il numero delle richieste pervenute, quindi questa trap può segnalare eventuali attacchi di tipo "flooding".

Codici ritornati:

0 se numConnessioniIn $<$ maxConnessioniIn

1 se numConessioniIn $>$ maxConnessioniIn

2 il numero delle Connessioni di in torna sotto la soglia stabilita

trapConnessioniOut: questa notifica viene segnalata al manager per indicare il superamento della soglia di connessioni che il client e' in grado di servire,

Codici ritornati:

0 se numConnessioniOut $<$ maxConnessioniOut

1 se numConessioniOut $>$ maxConnessioniOut

2 il numero delle Connessioni di out torna sotto la soglia stabilita

0.3 P2PMIB

P2P-MIB DEFINITION::=BEGIN

IMPORTS ....

P2PMIB MODULE-IDENTITY

LAST-UPDATE "0201231200Z"
ORGANIZAZION "GapXse"
CONTACT-INFO "Fabio Dianda"
GapXse (GAP XML Signature)
Corso Italia 40
Pisa Italy
dianda@sec.di.unipi.it

::={P2PMIB 1.3.4}

numConnessioniIn OBJECT-TYPE

SYNTAX COUNTER32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"Numero di connessioni in entrata in un dato momento, registra sia richieste che connessioni per ottenere la risorsa richiesta precendentemente"

::{P2PMIB 1}

numConnessioniOut OBJECT-TYPE

SYNTAX COUNTER32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"Numero di connessioni di uscita attive ad un dato momento
Utile a fini statistici "

::{P2PMIB 2}

byteInviati OBJECT-TYPE

SYNTAX COUNTER32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"Numero di byte inviati (in uscita) ai peer connessi"

::{P2PMIB 3}

byteRicevuti OBJECT-TYPE
SYNTAX COUNTER32
MAX-ACCESS read-only
STATUS current
DESCRIPTION "numero di byte ricevuti da un peer , misura sia le richieste
sia le connessioni (forse era meglio distinguere i due casi)

tabConnessioniIn OBBJECT-TYPE

SYNTAX SEQUENCE OF connessioni
MAX-ACCESS not-accessible
DESCRIPTION
"Una tabella per monitorare le connessioni in entrata"
::{P2PMIB 4}

```
connessioni  OBJECT-TYPE

        SYNTAX connessioni
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION "Informazioni per identificare un peer “
::{tabConnessioni 5}

connessioni  ::=SEQUENCE {
        srcip Ipaddress
        idrich OCTECT STRING
        }

srcip OBJECT-TYPE
        SYNTAX Ipaddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "Indirizzo Ip di un peer"
::{tabConnessioniIn 1}

idrich OBJECT-TYPE
        SYNTAX OCTECT STRING
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "una stringa per identificare la risorsa richiesta da un peer
                forse si poteva inserire l' hash della strign per migliorare
                il tempo di ricerca nella tabella"
::{tabConnessioniIn 2}




tabConnessioniOut OBBJECT-TYPE

        SYNTAX SEQUENCE OF connessioni
        MAX-ACCESS not-accessible
        DESCRIPTION
              "Una tabella per monitorare le connessioni in uscita"
::{P2PMIB 5}
```

codiceTrapConnessioniIn OBJECT-TYPE
SYNTAX UNSIGNED32 (0,1,2)
MAX-ACCESS read-write
STATUS current
DESCRIPTION "0 se numConnessioniIn < maxConnessioniIn
1 se numConnessioniIn > maxConnessioniIn"
2 se numConnessioniIn torna sotto maxConnessioniIn

::{P2PMIB 6}

codiceTrapConnessioniOut OBJECT-TYPE
SYNTAX UNSIGNED32 (0,1,2)
MAX-ACCESS read-write
STATUS current
DESCRIPTION "0 se numConnessioniOut < maxConnessioniOut
1 se numConnessioniOut > maxConnessioniOut"
2 se numConnessioniOut torna sotto maxConnessioniIn

::{P2PMIB 7}

maxNumeroConnessioniIn OBJECT-TYPE
SYNTAX COUNTER32
MAX-ACCESS read-wite
STATUS current
DESCRIPTION "Soglia di connessioni che un client puo' servire"
::{P2PMIB 8}

maxNumeroConnessioniOut OBJECT-TYPE
SYNTAX COUNTER32
MAX-ACCESS read-wite
STATUS current
DESCRIPTION "Soglia di connessioni che un client puo' servire"
::{P2PMIB 9}

notificaMaxConnessioniIn NOTYFICATION-TYPE
OBJECTS {numConnessioni,maxConnessioni}
STATUS current

```
        DESCRIPTION "Trap inviata quando il numero delle connessioni attive
                in entrata supera una soglia prestabilita"
::{P2PMIB 10}

notificaMaxConnessioniOut NOTYFICATION-TYPE
        OBJECTS {numConnessioni,maxConnessioni}
        STATUS current
        DESCRIPTION "Trap inviata quando il numero delle connessioni attive
                in uscita supera una soglia prestabilita"
::{P2PMIB 11}



END.
```

NOTA: Identificare un peer tramite un indirizzo IP può sembrare abbastanza logico, ma può non essere totalmente sicuro, in teoria sarebbero possibili attacchi di tipo "spoofing", in cui il peer richiedente non fornisce il suo indirizzo, ma quello di una vittima, la successiva connessione del peer produttore al "falso" consumatore potrebbe essere interpretato come attacco, anche se non particolarmente elaborato e perciò facilmente bloccabile.

Conclusioni Con questo lavoro ho cercato di introdurre alcune problematiche relative alla gestione dei client p2p, il lavoro e' certamente migliorabile e specializzabile, infatti non per tutti i client i dati hanno lo stesso significato, si prenda ad esempio la tabella delle connessioni, nel caso di un client tipo Napster in essa si trova l' indirizzo effettivo del peer che ha chiesto e che consumerà la risorsa, nel caso di un client tipo Freenet (Pure p2p) si trova l' indirizzo di un peer che ha chiesto la risorsa, senza sapere se il consumatore sarà lui o un altro.

Bibliography

[1] Claudio Telmon : "Il crepuscolo dei firewall" Commento del 07/06/2001 `http://www.telmon.org`

[2] Andrew Grimshaw: Distributed Computing and the New World Order of Enterprise Networks O' Reilly p2p and web services conference 2001 `http://conferences.oreillynet.com/cs/p2pweb2001/view/e_sess/1808`

[3] Luca Deri : Sistemi Elaborazione Informazione Gestione di Rete

[4] Nanog (North America Network Operation Group) `http://www.nanog.org`

[5] Infoanarchy `http://www.infoanarchy.org`