Appunti delle lezioni di Gestione di Sistemi e Reti
2006-2007

Franco Sirovich

2. Network Monitoring

Per monitoring intendiamo *tenere sotto osservazione*. Il termine monitoring è oramai entrato nel linguaggio comune e viene usato "all'italiana", ad es. monitorare, monitoraggio... Nell'ambito dei sistemi di gestione per *monitoring* si intende osservare ed analizzare stato e comportamento di tutto il sistema sottoposto a gestione e quindi:

- Applicazioni
- Sistemi finali (end system)
- Sistemi intermedi
- Sotto-reti

Un sistema di monitoring deve essere progettato con cura, e la sua progettazione coinvolge vari aspetti:

- *Accesso alla informazione da monitorare*. Le informazioni da tenere sotto controllo devono essere accuratamente definite e l'accesso a tali informazione deve essere progettato.
- *Meccanismi di monitoring*. I meccanismi per ottenere l'informazione desiderata devono essere realizzati in modo efficiente e accurato.
- *Utilizzo della informazione di monitoring*. Il monitoring non è fine a se stesso ma è finalizzato a realizzare le cinque aree funzionali della gestione. L'utilizzo della informazione di monitoring deve essere realizzato in maniera efficiente ed efficace.

In questo capitolo consideriamo solo i primi due aspetti del progetto di un sistema di monitoring. Sottolineiamo che il system monitoring coinvolge tutti le cinque aree funzionali descritte precedentemente: fault management, accounting management, configuration and name management, performance management e security management; il monitoring non è funzionale solamente al fault management! Tratteremo ora solo *performance monitoring*, *fault monitoring* e *accounting monitoring* perché sono le più importanti aree del monitoring.

2.1. Architettura di monitoring

Prima di definire l'architettura di un sistema di monitoring conviene esaminare il tipo di informazione che utile tenere sotto osservazione.

2.1.1. Monitoring information

L'informazione di monitoring può essere in primo luogo classificata in una elle seguenti tre classi.

Statica: è quella informazione che caratterizza la configurazione del sistema sotto gestione e i suoi componenti. È una informazione che cambia molto raramente, ma deve poter essere aggiornata facilmente quando è necessario perché deve sempre corrispondere alla configurazione reale del sistema. Esempi tipici sono: tipo e identificazione delle schede di rete di una macchina, le applicazioni installate sulla macchina e la loro configurazione, gli utenti registrati ad utilizzare la macchina e le sue applicazioni, ecc.

Dinamica: è quella informazione che caratterizza gli eventi che avvengono nel sistema. È una informazione che cambia molto più frequentemente della informazione statica perché in genere le informazioni che sottoponiamo a monitoring sono informazioni che cambiano frequentemente, o possono cambiare e desideriamo essere informati del cambiamento; in questo caso anche quando l'informazione non cambia è significativa. Esempi tipici sono: l'invio di un pacchetto da una scheda di rete, il cambio di stato di una macchina a stati finiti, il login di un utente registrato, ecc.

Statistica: è quella informazione che caratterizza l'evoluzione l'informazione dinamica. Cambia velocemente se l'informazione dinamica è molto "instabile". Esempi tipici sono: il numero di pacchetti inviati da una scheda di rete nell'unità di tempo (e misurato in un certo intervallo di tempo), la media e la varianza del numero di pacchetti inviati nell'unità di tempo.

Questi tre tipi di informazione vanno a costituire tre database che concettualmente contengono tutta la informazione di monitoring. Un esempio è fornito dal monitoring di un sistema di misura real-time descritto nella prossima figura.

Il database statico è costituito da due componenti: il *configuration database* che contiene informazioni che descrivono le macchine e gli elementi di rete che costituiscono l'infrastruttura del sistema, e il *sensor database* che descrive i sensori usati per ottenere misure real-time che costituiscono la finalità del sistema sottoposto a monitoring. Il database dinamico contiene informazione raccolta sullo stato delle macchine e degli elementi di rete, e sugli eventi individuati dai sensori di misura. Il database statistico contiene aggregati statistici utili a caratterizzare le informazioni contenute nel database dinamico. La figura mostra le relazioni fra questi tre livelli di database.

La natura della informazione di monitoring è importante perché ha implicazioni su dove può essere *generata* o *raccolta*, e su dove può essere *conservata*.

L'informazione statica è in genere generata dall'elemento che essa descrive. Ad es. un router mantiene la sua informazione di configurazione. Quando l'informazione di configurazione è generata dall'elemento stesso, è preferibile che il sistema di monitoring possa accedere direttamente a tale informazione, piuttosto che tenerne una copia. In tal modo si evitano tutti i problemi di "allineamento" fra la configurazione reale e quella contenuta nel sistema di monitoring. Per poter accedere all'informazione di configurazione contenuta nell'elemento è necessario che l'elemento possieda il necessario software di agent che interagisce con il monitor, oppure che sia possibile realizzare un proxy attraverso il quale il monitor può accedere a questa informazione.

Anche l'informazione dinamica è spesso raccolta e immagazzinata dall'elemento responsabile per gli eventi descritti, e, come nel caso precedente, resa disponibile al monitor tramite opportuno

software di agent. A differenza del caso di informazione statica, nel caso di informazione dinamica l'evento che siamo interessati a monitorare potrebbe essere osservato dall'esterno e quindi l'informazione raccolta e resa disponibile da un elemento diverso da quelb che la genera. Con il termine *remote monitor* si intende in genere un elemento che osserva e raccoglie informazioni su eventi esterni a lui, ad es. il traffico su una LAN. Il remote monitor non dovrebbe ovviamente in alcun modo influenzare la informazione monitorata altrimenti la misura effettuata è alterata.

L'informazione statistica può essere generata da qualunque elemento che abbia accesso alla informazione dinamica. Vi sono situazioni in cui rendere l'informazione dinamica disponibile al generatore dell'informazione statistica può essere troppo costoso, se l'informazione dinamica è di grandi dimensioni. Infatti, l'informazione statistica rappresenta una "sintesi" di grandi quantità di informazioni dinamiche. Dove collocare quindi la generazione dell'informazione statistica è un problema da esaminare con attenzione.

2.1.2. Configurazioni di monitoring

Un sistema di monitoring può essere scomposto in quattro componenti fondamentali.

Applicazione di monitoring: include le funzioni direttamente visibili all'utente del sistema di monitoring, quindi performance monitoring, fault monitoring, accounting monitoring. *Funzione di manager*: il modulo che fornisce le funzioni di base di raccolta delle informazioni di monitoring dagli altri elementi del sistema. *Funzione di agent*: il modulo che raccoglie e registra le informazioni di monitoring da uno o più elementi del sistema sottoposto a monitoring e le comunica al manager. Agent e manager sono i due moduli che permettono lo scambio di informazione di monitoring sulla rete. *Managed objects*: l'informazione di management che rappresenta le risorse e le loro attività.

La figura 2.2 (a) mostra la interazione fra i quattro moduli.

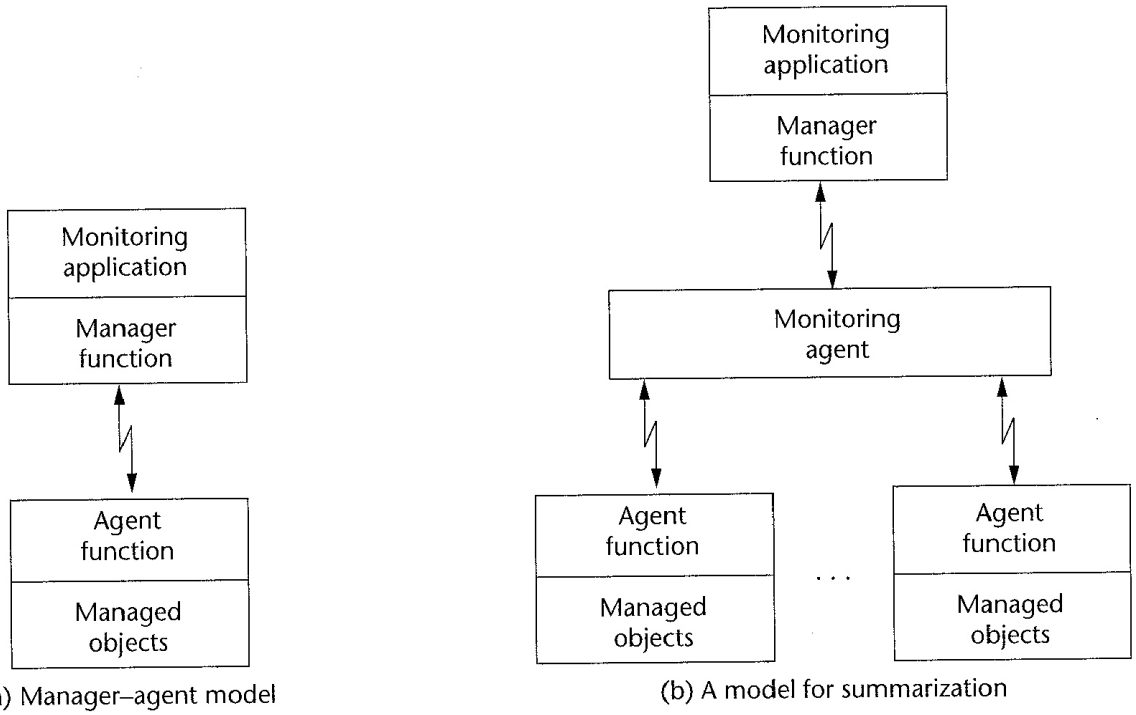


FIGURE 2.2 Functional architecture for network monitoring

Il calcolo delle informazioni statistiche è una funzione della applicazione di monitoring, ma può richiedere la introduzione di una modulo addizionale per evitare che grandi quantità di informazioni

vengano trasferite sulla rete. Il *monitoring agent* è un modulo che genera sommari e analisi statistici di informazioni dinamica di monitoring. Se questo modulo non è all'interno della applicazione di monitoring ed è invece remoto, allora si comporta da agent e comunica tale informazione al manager (caso (b) nella figura di cui sopra). Come si vede abbiamo ancora trasmissione di dati su rete, ma il monitoring agent può essere collocato più vicino agli elementi da monitorare.

2.1.3. Polling e Event Reporting

Nello scenario del monitoring, l'agent deve fornire informazioni al manager: ci sono due tecniche per realizzare questo scambio di informazione che vengono indicate con i nomi di *polling* e di *event reporting*.

La tecnica del *polling* consiste nell'organizzare il trasferimento dell'informazione con una interazione di tipo *richiesta-risposta*, nella quale il manager chiede informazioni all'agent, e l'agent risponde (se il manager ha le necessarie autorizzazioni). Quindi l'agent ha il ruolo di *listener* e attende che il manager lo interroghi. Le richieste che il manager può fare possono essere di diverso tipo. Può essere una richiesta per ottenere una o più informazioni ben identificate, e le informazioni richieste possono essere di tipo statico, dinamico oppure statistico. Naturalmente le informazioni richieste devono essere nella MIB dell'agent altrimenti la risposta restituirà solo un codice di errore. Un'altra possibilità è che il manager richieda informazioni che soddisfano particolari criteri, e quindi l'agent deve essere in grado di eseguire una ricerca all'interno della sua MIB. Infine, il manager potrebbe richiedere informazioni riguardo alla MIB contenuta nell'agent. Non tutte queste possibilità sono sempre presenti in una architettura di gestione che impieghi la tecnica del polling, ma in genere tutte permettono di richiedere informazioni specifiche identificate.

Quando invece si adotta la tecnica dell'*event reporting* è l'agent prende l'iniziativa di inviare informazione al manager, che ha il ruolo di listener e deve essere preparato e in attesa di ricevere queste informazioni che l'agent può autonomamente decidere di inviare in certe situazioni. Anche nel caso di event reporting la informazione fornita può essere di tipo diverso. L'agent può generare un *report periodico* contenente informazioni di management specifiche. L'informazione fornita e il periodo di generazione del report possono essere predefiniti nell'agent, oppure potrebbero essere configurabile (settabile) dal manager. (Si noti a questo proposito che operazioni di modifica della configurazione dell'elemento o dell'agent non possono essere realizzate se non con la tecnica del polling.) L'agent può inviare informazione al manager quando avviene un evento significativo (ad es. un cambio di stato), oppure ancora quando avviene un evento inusuale (ad es. un fault). L'event reporting fornisce informazione appena l'evento si verifica, mentre con il polling il manager viene a sapere che l'evento è avvenuto solo quando effettua una richiesta che gli permette di individuare l'occorrenza dell'evento. L'event reporting è più efficiente quando l'informazione cambia poco frequentemente, se l'agent è in grado di fornire l'informazione solo al momento che questa si modifica (e la comunicazione è affidabile).

In genere vengono usati ambedue i metodi, in situazione diverse, anche se con enfasi diverse in diverse architetture di gestione, perché ambedue i metodi hanno i loro pro e contro (come discuteremo meglio nel seguito) e nessuno dei due è decisamente superiore all'altro. La scelta della tecnica da adottare in una certa situazione dipende da vari fattori che è opportuno considerare per comprendere appieno la funzione di queste tecniche all'interno di una architettura di gestione:

- Il traffico di rete generato
- La robustezza in situazioni critiche
- Il ritardo temporale nella notifica dell'informazione al manager
- L'ammontare di elaborazione richiesta negli elementi gestiti
- Costo relativo fra trasferimento affidabile e trasferimento non affidabile

- Le applicazioni di gestione supportate
- Come trattare il caso in cui il dispositivo che deve fare la notifica "muore" prima di inviare l'event report

Come si vede dalla lista, vi sono molti aspetti del problema della gestione che si devono tenere presenti e quindi non esiste la soluzione ottima in tutte le situazioni.

2.2. Performance monitoring

La misura e il monitoring delle prestazioni è un requisito assoluto di una buona gestione di un sistema, anche se spesso questo obiettivo è trascurato, tipicamente in favore del monitoring dei fault del sistema. Questa "disattenzione" è probabilmente dovuta, oltre ad una carenza culturale, alle difficoltà che si incontrano nel misurare le prestazione con accuratezza e in modi indiscutibili.

Il primo problema che si incontra è quello di scegliere gli indicatori da usare per misurare le prestazioni del sistema, e il problema della selezione ha molte origini:

- Sono stati proposti nella letteratura e nella pratica molti indicatori: è quindi non realistico usarli tutti e la scelta deve essere effettuata su un grade numero di proposte.
- Il significato di molti di questi indicatori non è chiaro e/o è difficile da comprendere. I gestori e gli utilizzatori sono restii ad usare indicatori che non comprendono bene come indichino la prestazioni del sistema.
- Gli indicatori di prestazioni devono essere misurati dal software di gestione, e molti indicatori sono supportati solo da alcuni costruttori.
- Molti indicatori supportano malamente il confronto e quindi sono utili solo per indicare una tendenza nelle prestazioni del sistema.
- È sovente il caso che anche in presenza di misure accurate dell'indicatore la interpretazione della misura da parte del sistema di gestione sia carente, e induca in errori i gestori del sistema.
- Sovente il calcolo di questi indicatori è (considerato) oneroso e quindi non vengono adottati perché troppo costosi (o ritenuti tali).

I numerosi indicatori proposti cadono un due grandi classi:

- *Orientati al servizio* (service-oriented): sono indicatori che intendono misurare la qualità del servizio offerto dal sistema ai suoi utilizzatori, che possono essere utenti umani, ma anche altri sistemi o componenti di un sistema più complessivo. Fra questi indicatori orientati al servizio i più significativi sono la *disponibilità*, il *tempo di risposta*, l'*accuratezza*.
- *Orientati all'efficienza* (efficiency-oriented): questi indicatori intendono misurare aspetti delle prestazioni del sistema che sono di interesse per chi realizza il servizio e non necessariamente (o raramente) sono percepiti come qualità da parte degli utilizzatori del servizio. Fra questi indicatori i più significativi sono il *throughput*, e il *coefficiente di utilizzazione*.

Gestire bene un sistema significa mantenere i livelli di servizio stabiliti per incontrare la soddisfazione degli utenti. Gli indicatori orientati al servizio sono dunque quelli da monitorare a più alta priorità. Ma i gestori sono anche interessati a fornire il servizio pattuito al minimo costo e quindi gli indicatori orientati all'efficienza devono essere considerati per evitare che il costo della fornitura del servizio non diventi eccessivo.

2.2.1.1. Disponibilità (availability)

La *disponibilità* di un sistema è definita come la percentuale di tempo che il sistema (componente,

applicazione) è *disponibile* per i suoi utilizzatori. Si noti subito che ha senso solo se associata alla ampiezza dell'intervallo di tempo in cui deveessere misurata, intervallo che necessariamente deve essere abbastanza lungo perché sia significativa.

Quando la disponibilità può essere solo parziale (non solo tutto o niente) si dovrebbe parlare di *disponibilità funzionale*, in modo da tenere conto delle funzionalità limitate che il sistema non completamente funzionante ha comunque offerto ai suoi utenti. Le limitazioni possono essere di tipo prestazionale (ad es., maggior ritardo nella risposta, maggiore lentezza, minor spazio disco a disposizione) oppure di tipo più precisamente funzionali (una certa funzione del sistema non è disponibile ma le altre sono disponibili). La disponibilità funzionale è ancora più difficile da definire, calcolare e comprendere di quella "binaria", e quando il livello di funzionalità è oggetto di un contratto, è spesso fonte di contenziosi.

Per certe applicazioni la disponibilità è una caratteristica di prestazione essenziale; ad es. un sistema di prenotazione aerea, un sistema bancario, un sistema di compravendita di azioni in borsa: ogni minuto di non disponibilità del sistema comporta danni monetari elevatissimi.

La disponibilità si basa sulla *affidabilità* dei componenti del sistema. L'affidabilità di un componente è definita come la *probabilità* che il componente si comporti come prescritto nelle condizioni di funzionamento prescritte.

La disponibilità è spesso espressa in termini di *tempo medio fra due fallimenti* (*Mean Time Between Failures, MTBF)*, e di *tempo medio per riparare* (*Mean Time To Repair, MTTR*) il fallimento, nel seguente modo:

$$A = \frac{MTBF}{(MTBF + MTTR)}$$

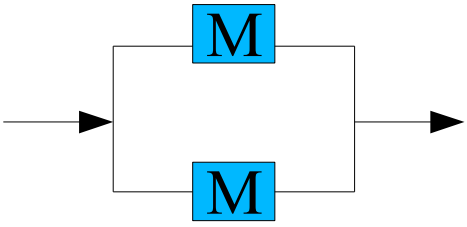
La disponibilità dipende dalla disponibilità dei componenti ma anche dalla organizzazione del sistema. Ad es., se abbiamo due modem collegati da una linea di comunicazione, come nella seguente figura,

se indichiamo con M e L la disponibilità dei modem e del link, la disponibilità del sistema complessivo è

$$A=M*L*M$$

quindi minore di quella di ciascun componente, perché i tre fattori sono tutti minori di 1.

Nel caso di sistemi realizzati mediante collegamenti in parallelo che si suddividono i compiti, come quello illustrato nella prossima figura,



il sistema complessivo non è disponibile solo quando *ambedue* i componenti sono indisponibili. Possiamo allora esprimere la disponibilità in termini della probabilità che non avvenga che i due componenti siano ambedue non disponibili, cioè

$$A=1-(1-M)(1-M)=1-(1+M^2-2M)=2M-M^2$$

L'analisi si complica se si vuole tenere conto delle funzionalità offerta nelle situazioni in cui funziona un solo componente, e quindi se vogliamo esprimere la disponibilità funzionale del sistema,

per es. nel caso in cui il fallimento di uno dei due modem provoca una riduzione della banda di comunicazione offerta agli utilizzatori (chi è interessato veda sul libro).

2.2.1.2. Tempo di Risposta (Response Time)

Il *tempo di risposta* è definito come il tempo impiegato dal sistema per (iniziare a) fornire la risposta all'input da parte dell'utente (o del sistema utilizzatore). In un *sistema interattivo* è il tempo che intercorre fra l'istante in cui l'utente ha digitato l'ultimo carattere del suo input e l'istante in cui compare sullo schermo il primo carattere dell'output della transazione interattiva. Per un *sistema server* è in generale il tempo che impiega a rispondere alla richiesta che gli arriva dal client.

Si noti che è molto differente considerare l'applicazione client come parte del sistema di cui si misura il tempo di risposta, oppure considerarla come esterna. Se l'applicazione client è considerata esterna non si considerano:

- I due tempi di risposta dell'applicazione client
- Le due latenze della rete che connette il client al server

L'utilizzatore del server tenderà ovviamente a considerare l'applicazione client come parte del sistema che lui sta utilizzando, mentre il fornitore del server tenderà a non considerare né il client né la rete come parte del servizio offerto. Anche nei casi in cui il fornitore dell'applicazione server fornisce anche il software dell'applicazione client, non accetterà di essere responsabile della latenza della rete che interconnette client e server. Solo il gestore del sistema è responsabile spesso di tutti e tre i componenti: *client*, *server* e *rete di comunicazione*; certamente è suo compito *monitorare* le prestazioni di tutti e tre i componenti, anche (se non soprattutto!) nel caso in cui la rete di comunicazione sia realizzata da un fornitore esterno (ad es., un Service Provider).

Il tempo di risposta è un indicatore che si desidera minimizzare, ma ridurre i tempi di risposta aumenta inevitabilmente i costi del sistema, per le seguenti ragioni.

Potenza di calcolo: per avere tempi di risposta piccoli occorre avere alta potenza di calcolo; esigenza che aumenta quanto maggiore è la quota di tempo di risposta che dipende dal tempo di elaborazione, cioè se l'applicazione è *compute bound*.

Velocità del sistema di input/output: determina la quota di tempo di risposta dovuta all'I/O; la rete di comunicazione è un componente importante di questo tempo di I/O: è critico il ritardo causato dalla comunicazione in rete, soprattutto nelle applicazioni che sono *"network bound"*, cioè nelle quali il tempo di esecuzione del client e del server è trascurabile rispetto alla latenza della rete di comunicazione. Non si dimentichi però che a volte la componente server (e talvolta anche quella client) possono essere *I/O bound* nel senso tradizionale del termine. In questi casi, per ridurre il tempo di risposta del sistema occorre intervenire sulla velocità dell'I/O e/o sulla *latenza* della rete di comunicazione.

Esigenze contrastanti: per migliorare il tempo di risposta dei processi che implementano il sistema sotto monitor si peggiora inevitabilmente il tempo di risposta di altri, perché le risorse di calcolo e di I/O sono quasi sempre condivise. Ci si trova quindi molto spesso a dover negoziare gli interventi fra queste esigenze contrastanti, che bisogna tenere presenti.

Il tempo di risposta di un sistema è un parametro prestazionale molto critico perché un tempo di risposta inadeguato può rendere il sistema totalmente inadatto a certi usi. Alcuni esempi sono i seguenti:

- Un sistema interattivo viene "rifiutato" dagli utenti se il tempo di risposta supera i 2 secondi: l'utente deve ricordare per troppo tempo l'informazione che gli serve per interpretare l'output del sistema, si distrae e non riesce a lavorare né con soddisfazione né con efficacia.

- Se il compito dell'utente richiede molta elaborazione mentale (ad es. applicazioni grafiche) occorre risposta in meno di un secondo, perché altrimenti la perdita del contesto da parte dell'utente rende l'applicazione inusabile.
- L'echo dei caratteri sullo schermo non può richiedere più di pochi decimi di sec.: in caso contrario l'utente crede di aver mal digitato i caratteri, li ripete, e poi cerca di cancellarli, ma anche l'echo del carattere di cancellazione giunge troppo in ritardo perché l'utente capisca "a che punto è" della sua digitazione dell'input.
- Il tempo di risposta di un sistema deve essere *"uniforme"* perché l'utente si adatta alla velocità del sistema e si aspetta l'output entro un certo tempo. Quindi il tempo di risposta deve cambiare poco nell'arco di una giornata e al cambiare della transazione eseguita. In alcuni sistemi interattivi si prolunga artificialmente il tempo di risposta di transazioni particolarmente "leggere" per non creare nell'utente aspettative che non possono essere mantenute nel caso di transazioni più "pesanti".

Il fatto importante che occorre tenere presente è che la diminuzione del tempo di risposta aumenta la produttività dell'utente. Misurata in termini di numero di transazioni all'ora completate. Il peggioramento della produttività è, sfortunatamente, non lineare con l'aumento del tempo di risposta e quindi si presentano i tipici effetti *soglia*, per cui un piccolo peggioramento del tempo di risposta provoca conseguenze enormi mentre un peggioramento della stesso ammontare precedentemente verificatosi non aveva dato luogo a proteste significative. Il tempo di risposta è quindi un essenziale parametro di qualità da considerare nel *progetto* di un sistema transazionale, e per queste stesse ragioni da *monitorare* in un sistema in esercizio: La gestione di un sistema transazionale deve tenere sotto controllo il tempo di risposta effettivo del sistema in uso, perché il suo deterioramento è "sofferto" dagli utenti. Non si dovrebbe dimenticare che il tempo di risposta, misurato in condizioni normali di funzionamento del sistema, dovrebbe anche essere un parametro critico nella accettazione del sistema da parte dell'acquirente: ma questa è un'altra storia!

Il tempo di risposta di una transazione è quasi sempre il risultato di addendi costituiti dai tempi di risposta di sotto-componenti del sistema che si passano "di mano in mano" la elaborazione della transazione. Se il tempo di risposta complessivo è eccessivo, occorre analizzare l'architettura del sistema, individuare gli addendi e individuare quale di questi addendi è quello preponderante o quello che sta degenerando rispetto alla situazione di funzionamento normale. In un sistema client/server in cui ci interessa il tempo di risposta all'utente interattivo abbiamo almeno tre componenti: *client-rete-server*.

Ciascuno di questi componenti potrebbe contribuire con due addendi: tempo di elaborazione dell'input dell'utente e tempo di elaborazione dell'output, cioè della risposta da fornire all'utente. Possiamo indicare come *upwards*, verso l'alto, i tempi di elaborazione dell'input fornito dall'utente, e come *downwards*, verso il basso, i tempi di elaborazione necessari per fornire all'utente la risposta una volta che la risposta sia stata calcolata. I tempi di latenza della rete di comunicazione possono essere visti come una forma di elaborazione dei dati della transazione.

Ma soprattutto il server può essere costituito da sotto-componenti molti significativi, quali la logica applicativa e il database (uno o più). Se il database è situato su un altro host, abbiamo di nuovo una relazione client/server con una rete (in genere locale) che connette il server applicativo al DBMS. La logica applicativa (così come il DBMS) fanno uso dei servizi del sistema operativo, introducono a loro volta ritardi e che debbono essere monitorati perché possono essere ottimizzati o essere loro stessi causa del degrado.

Nel *monitorare* e *analizzare* il tempo di risposta di un sistema è quindi essenziale chiarire bene quali sono il perimetro del sistema che si sta monitorando e quali sono i suoi componenti. È facile che ci sfugga un componente essenziale che con il suo tempo di risposta condiziona pesantemente il tempo di risposta del sistema: *se un addendo viene ignorato si attribuisce inevitabilmente il suo ritardo agli addendi di cui si ha conoscenza!*

2.2.1.3. Accuratezza (accuracy)

Un sistema di calcolo o di comunicazione deve comportarsi in modo "corretto", cioè rispettando le specifiche: quando il comportamento del sistema non è conforme alle specifiche si dice che *il comportamento non è corretto*. L'accuratezza misura la percentuale dei risultati corretti sui risultati totali. Una misura alternativa equivalente è naturalmente la *percentuale di errori* nel comportamento del sistema. L'accuratezza si vorrebbe che fosse tendenzialmente pari a 1 (ma è in pratica inferiore a 1), il tasso di errori si vorrebbe che fosse tendenzialmente pari a 0 (ma è in pratica maggiore di 0).

Le specifiche di un sistema dettano vari aspetti del comportamento, rispetto ai quali si potrebbero avere diversi livelli di accuratezza. Nei protocolli di comunicazione l'accuratezza è in primo luogo relativa alla *integrità dei dati ricevuti*, e *codici di verifica degli errori* (e in alcuni casi di *correzione automatica*) sono introdotti allo scopo di rendere particolarmente basso (trascurabile) il *tasso di errore residuo*, cioè il tasso degli *errori non rilevati*, che comportano conseguenze molto più gravi di quelli rilevati.

È possibile misurare dall'esterno (tramite *sonde*) oppure mediante *strumentazione* del software il tasso di errore e per questa via l'accuratezza del servizio di comunicazione: un aumento del tasso di errori indica problemi che vanno risolti subito!

La stessa esigenza è presente in qualunque sistema di calcolo, anche se è molto più difficile calcolare gli errori di un sistema di calcolo che quelli di un sistema di comunicazione (la funzione calcolata dal sistema di comunicazione dovrebbe essere la funzione identità!). Una buona applicazione dovrebbe almeno loggare (e quindi permettere di misurare) le situazioni di errore interno in cui si viene a trovare, che sono certo situazioni in cui l'applicazione certamente non è accurata. Monitorando i file di log si può fare molto di più di quanti si creda (e non si faccia!)

2.2.1.4. Throughput

Il throughput è considerata una misura orientata alle applicazioni, ma ha senso anche per i servizi di comunicazione. Esempi di misura che ricadono in questa classe sono:

- Numero di transazioni eseguite in un certo periodo di tempo
- Numero di sessioni di utente mantenute per un certo periodo di tempo
- Numero di chiamate in un sistema a commutazione di circuito in un certo periodo di tempo
- Numero di byte trasferiti in un certo periodo di tempo

Si deve subito notare che l'attenzione è posta alla misura *per un periodo sostanziale di tempo*, anche se poi l'unità di misura è generalmente "al secondo": in altre parole è una misura *media*, non *istantanea*. Quando sistema ha throughput diverso se misurato nel breve periodo o nel lungo periodo, si distingue quest'ultimo aggiungendo l'aggettivo *sostenuto* (*sustained*). Ci possono essere fenomeni di esaurimento di risorse che si manifestano in un lungo periodo e che quindi abbassano il throughput sostenuto rispetto a quello a breve; un esempio potrebbe essere il rallentamento introdotto dalla garbage collection di un sistema realizzato in Java.

È importante distinguere il throughput dalla latenza o dal tempo di risposta: se un sistema ha un tempo di risposta di un millisecondo NON È VERO che necessariamente abbia un throughput di 1000 transazioni al secondo. Infatti, quando si parla di throughput si vuole misurare la capacità del sistema di sopportare un carico alto e non coordinato in modo da permettere al sistema di reagire meglio. Nell'esempio precedente, il sistema può avere bisogno di eseguire delle operazioni (fra le quali potrebbe esserci la garbage collection) dopo che ha fornito la risposta, oppure potrebbe essere rallentato dalla richiesta contemporanea di due transazioni, a causa della necessità di assicurare mutua esclusione su alcune risorse del sistema. Quindi potremmo avere situazioni in cui potrebbe fornire 1000 transazioni al secondo ma solo se avesse memoria infinita, oppure se le transazioni fossero

coordinate in modo da presentarsi esattamente serializzate. Ma non possiamo fornire memoria infinita né coordinare tutti gli utenti del sistema in modo che si presentino uno alla volta! Il modo "corretto" di misurare il throughput di un sistema transazionale è quello di "aggredire" il sistema con un numero elevatissimo di transazioni lanciate contemporaneamente e verificare sia il tempo medio di risposta che il tempo necessario a completare il "burst" di transazioni: il tempo di risposta così misurato sarebbe più accurato di quello misurato con il sistema scarico, e il numero di transazioni al secondo misurato sarebbe quello che potremmo davvero avere in condizioni di carico alto del sistema.

In conclusione, siamo interessati al throughput in condizioni di carico "normale" del sistema (fra le quali ci sono le condizioni di alto carico) e per un periodo sufficientemente lungo per evidenziare il comportamento a regime del sistema. Solo in questo contesto e quando il carico offerto al sistema è alto ha senso misurare il throughput: se il sistema non viene caricato ovviamente eseguirà poche transazioni al secondo (perché nessuno le invoca!) e anche se risponde velocemente soddisfa pochi utenti! Quindi, anche il tempo di risposta andrebbe misurato in condizioni di carico normale e sostenuto, ma spesso ci si dimentica di questa esigenza.

Nelle reti di comunicazione la latenza della rete (tempo di risposta) è profondamente diversa dalla sua capacità (throughput): I protocolli che vogliono ottimizzare il throughput hanno sviluppato tecniche apposite, quali la sliding window e la bufferizzazione in spedizione e in ricezione, proprio per avere un throughput il più possibile indipendente dalla latenza di rete. Le tecniche per aumentare il throughput introducono però ritardi addizionali (soprattutto la bufferizzazione): (ri)studiarsi TCP, nel quale sono offerti alle applicazioni controlli (servizio PUSH) che permettono di disabilitare le ottimizzazioni rivolte alla massimizzazione del throughput.

La conclusione fondamentale di tutte queste considerazioni è che monitorare il throughput (e il tempo di risposta) è essenziale quando questa prestazione è di interesse per gli utenti.

2.2.1.5. Utilizzazione (utilization)

È una misura più fine del throughput, che misura una funzionalità a livello utente, perché consiste nella percentuale di tempo per cui una risorsa è in uso (o la quantità di uso della risorsa) in un certo periodo di tempo. È quindi una misura di prestazione *interna* al sistema di cui l'utilizzatore non ha visibilità, ma che è essenziale per la buona gestione del sistema. Dato che è una misura percentuale è spesso evidenziato il periodo di misura, anche perché la utilizzazione di una risorsa può subire ampie variazioni nel tempo.

Il fattore di utilizzazione è una misura molto importante perché spesso le prestazioni del componente/risorsa degradano in modo esponenziale con il crescere dell'utilizzazione o, peggio ancora, quando l'utilizzazione supera una soglia critica.

Il fattore di utilizzazione deve essere monitorato, perché cambia nel tempo, e occorre verificare che il fattore di utilizzazione delle varie risorse del sistema sia "equilibrato", cioè non vi siano risorse che sono molto più utilizzate di altre. Le risorse poco utilizzate sono ovviamente sovradimensionate e quindi "sprecate", ma non è questo il problema più grave.

Alla crescita dell'uso del sistema, l'utilizzo delle risorse crescerà, quando il sistema è ben progettato, in modo lineare (cioè proporzionale al fattore di utilizzo in corso): se una risorsa è utilizzata al 90%, è molto probabile che arrivi al 100% e mandi quindi in congestione il sistema, anche se vi sono altre risorse a bassa utilizzazione.

A tale riguardo, è molto istruttivo l'esempio riportato sul libro. Il Link3 di 60Kbps ha il 15% della capacità di banda complessiva del sistema, ma sopporta il 25% del carico totale. Quindi, anche se ha una capacità di 60Kbps è più "critico" dei due link di 40Kbps!

2.2.2. Performance Monitoring

La funzione di *performance monitoring* è costituita da tre componenti:

- Misura delle prestazioni (performance measurement)
- Analisi delle prestazioni (performance analysis)
- Generazione di traffico sintetico

Misura delle prestazioni: consiste nella raccolta di statistiche sul traffico e dei tempi di risposta del/nel sistema. Questa misura è spesso ottenuta da moduli presenti all'interno dei dispositivi o dei componenti dei quali si vogliono misurare le prestazioni.

Tutte queste misure "costano" in termini di richieste di elaborazione e memoria dei componenti coinvolti. Per questa ragione, misure ottenute mediante sonde (probe) esterne sono da preferire, anche se aggiungono componenti al sistema, perché non sovraccaricano (quasi completamente) i sistemi sotto misura.

Analisi delle prestazioni: consiste in funzionalità/software per ridurre e presentare i dati raccolti dalle misure. Infatti, i dati raccolti sono di grandi dimensioni e la loro stessa visualizzazione è un problema. Inoltre, i dati raccolti vanno "interpretati" per essere utili ai gestori: si rende quindi necessario disporre di software di predizione dell'andamento delle prestazioni al mutare della dimensione e del tipo di carico sul sistema. Per fare una buona analisi delle prestazione, che sia uno strumento per *pianificare la crescita* del sistema (*capacity planning*) occorrono opportuni *modelli* del sistema.

Generazione di traffico sintetico: permette di osservare il comportamento del sistema sotto carico controllato. Questi strumenti sono essenziali nella accettazione di un sistema al momento della consegna, ma anche nel corso della gestione per:

- Verificare il comportamento del sistema (mettendolo off-line) in condizioni di traffico anomalo
- Verificare il comportamento del sistema al crescere del carico, iniettando traffico addizionale (molto pericoloso se il sistema non è off-line o in condizioni di funzionamento non critiche, ad es. di notte...)

Gli strumenti di generazione di carico sintetico sono di una importanza pari alla facilità con cui ci si dimentica della loro utilità!

2.2.3. Misure esaustive vs statistiche

Raccogliere in modo esaustivo tutti i dati di misura di un sistema può essere inaccettabile, quando il carico del sistema è elevato e i dati da raccogliere sono estremamente numerosi. Un buon esempio è fornito dalla necessità di costruire una matrice che misuri accuratamente il numero totale di pacchetti inviati, in un certo periodo di tempo, da ogni nodo di una rete ad ogni altro nodo della rete. Può non essere possibile raggiungere questo obiettivo perché manca potenza di calcolo, oppure memoria RAM, oppure ancora memoria disco per immagazzinare i risultati.

L'alternativa alla misura *esaustiva* è la misura *statistica*, che considera ogni grandezza da misurare come una *variabile casuale*, le cui caratteristiche devono essere valutate mediante un appropriato campionamento. Occorre però fare attenzione nel valutare le stime statistiche dei risultati ottenuti dal campionamento, perché possiamo essere in presenza di una bassa probabilità dell'evento, ad es. nel caso in cui fossimo interessati ad errori, oppure in presenza di misure non indipendenti fra di loro, ad es. nel caso in cui si sia in presenza di fenomeni di burst: gli errori oppure i burst a cui siamo interessati possono benissimo sfuggire al nostro campionamento.

Per effettuare misure statistiche attendibili occorre quindi conoscere bene il fenomeno a cui siamo

interessati, ma anche la teoria della Probabilità e della Statistica!

2.3. Fault Monitoring (controllo dei guasti)

Gli obbiettivi del controllo dei guasti sono due:

- Identificare i fault il più presto possibile dopo il loro verificarsi, e
- Identificare la causa del fault, per poter prendere azioni correttive.

Un buon sistema di fault monitoring è quindi uno strumento indispensabile per realizzare una alta disponibilità del sistema, disponibilità che è una importante componente della Qualità del Servizio percepita dagli utilizzatori. È però evidente che anche la "robustezza" del sistema (il suo grado di "fallosità") e il tempo necessario a riparare i guasti sono componenti essenziali del tempo di indisponibilità del servizio in un arco temporale lungo (tipicamente un anno).

Fare fault monitoring non è così facile. Si incontrano vari problemi.

2.3.1. Problemi nel fault monitoring

La complessità del sistema da gestire comporta conseguenza che a prima vista possono apparire sorprendenti e che esaminiamo ora in modo sintetico.

Fault non osservabili: vi sono fault che per la loro stessa natura non sono osservabili direttamente: si pensi ad es. un deadlock. Ma ve ne sono anche altri che non sono osservabili direttamente per carenza di "strumentazione", per es. perché il dispositivo non segnala il fault, oppure questo è assai difficilmente misurabile, ad es. la perdita di un interrupt in un sistema operativo.

Fault parzialmente osservabili: Il fault è osservabile, ma non è sufficiente per individuare la causa del malfunzionamento. Ad es., un componente può comportarsi in modo scorretto a causa del malfunzionamento di un altro componente a lui collegato (che invece a sua volta non è osservabile). In questi casi la nostra attenzione viene portata sul componente "sbagliato" e la diagnosi sarà molto difficile.

Incertezza nella osservazione: l'osservazione del guasto può non essere significativa; ad es. un componente può rispondere in ritardo a causa di un suo fault, ma anche a causa di un fault della rete, oppure perché un timer locale misura male il tempo (e quindi "sembra" che il componente non risponda in tempo).

Una volta che il guasto sia stato individuato, si deve passare alla individuazione della sua causa, perché questa è l'unica vera ragione per cui siamo interessati a individuare i guasti. In questa diagnosi possiamo incontrare problemi di varia natura:

Molte cause potenziali: quando sono coinvolte diverse tecnologie, lo stesso malfunzionamento può essere causato da cause di natura molto diversa (che richiedono quindi azioni correttive diverse!). Fare la diagnosi corretta quindi significa individuare la causa effettiva del fault, non una solo plausibile. Un buon esempio è illustrato dalla prossima figura.

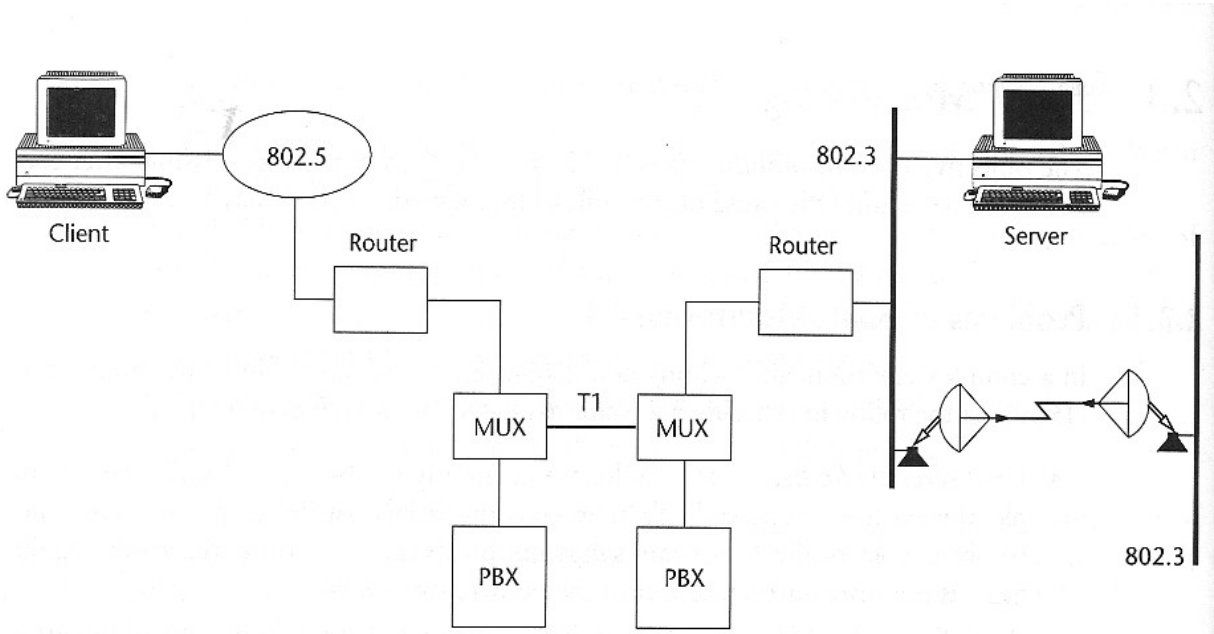


FIGURE 2.8 Heterogeneous network environment

In questo sistema vi sono molti punti di fallimento costituiti da tutti i componenti che devono essere attraversati da un pacchetto cheviaggia dalla macchina client alla macchina server. La perdita di connettività può essere causata dal fallimento di uno qualunque dei link/reti attraversati.

Troppe osservazioni di guasti che sono fra di loro correlate: Ancora usando la figura precedente

come esempio, un guasto del link T1 (un link geografico veloce) si manifesta in moltissimi modi: caduta delle connessioni dati fra le stazioni token ring e quelle Ethernet, caduta delle connessioni voce fra i due PBX (centralini telefonici privati). Ma anche rimanendo nel caso di connettività fra client e server, possiamo rilevare la mancata risposta dell'applicazione server, e la caduta della connessione TCP fra client e server. Quando il link T1 va in fault, vengono segnalati tutti questi "errori" (il problema del PBX difficilmente viene segnalato al gestore della rete di comunicazione...), che però hanno una unica causa. Occorre essere capaci di selezionare il guasto apparente che più facilmente ci porterà alla individuazione della causa: se cerchiamo la causa nel client o nel server difficilmente troveremo la soluzione!

Guasti causati dalla concomitanza di più di un evento, ciascuno dei quali non è sufficiente da solo a causare il malfunzionamento. I guasti dovuti a singola sono effettivamente di gran lunga i più comuni e quindi si è portati a cercare un singolo evento che causi il malfunzionamento. Ma è ovvio che nei pochi causi in cui il guasto è dovuto alla concomitanza di due o più "eventi" non si riesce a fare la diagnosi se non cambiando modo di ragionare. La diagnosi deve spesso essere effettuata ragionando su più osservazioni, sia in condizioni normali che in condizioni anomale. Ma gli eventi che se concomitanti causano il malfunzionamento si possono benissimo osservare in situazioni di funzionamento regolare, il che ragionando nell'ipotesi di singola causa, ci porta ad escludere che possano essere la causa.

Interferenza fra diagnosi e procedure di rimedio locali: le azioni di rimedio locali possono distruggere le prove o le tracce del guasto e quindi rendere impossibile una ulteriore diagnosi se il rimedio si dimostra inefficace. Molto spesso si traggono conclusioni affrettate e le azioni intraprese rendono impossibile risalire alle cause del guasto. Certamente, se il guasto è intermittente ma ricorrente si verificherà di nuovo, ma lo "pagheremo" un'altra volta e di nuovo dovremo affrontare i costi del tempo per eseguire la diagnosi.

Assenza di strumenti automatici di test: gli strumenti di test per isolare (individuare) i guasti sono sono molto utili ma anche molto difficili e costosi e costosi da amministrare: troppo spesso vengono ritenuti superflui o non giustificati, perché è difficile valutare i costi del tempo aggiuntivo speso nella diagnosi.

2.3.2. Funzioni di fault monitoring

Un sistema di fault monitoring deve rispondere a numerosi requisiti:

- *Individuare* e *riportare al manager* i fault. Questo sembra assolutamente il minimo, ma in realtà è molto difficile, in primo luogo perché implica l'abilità di *riconoscere* l'occorrenza di un guasto e distinguerla con certezza da un evento "normale". Ci si deve in molti casi accontentare di meno...
- *Mantenere log* di eventi significativi e di potenziali errori. Questo veramente il minimo che un agente di gestione deve fare per supportare il fault management. I sistemi manager controlleranno questi log periodicamente con uno schema di polling
- Quando un agente di gestione è in grado di riportare errori deve poterlo fare *indipendentemente verso uno o più sistemi di gestione manager*; per non intasare la rete il criterio di "errore" deve essere ragionevolmente "stretto", in modo che il traffico generato e il carico sui manager non sia eccessivo.
- *Anticipare i fault:* si vorrebbe che i fault non si verificassero mai. Quasi sempre i guasti totali sono preceduti da periodi in cui il componente che sta per guastarsi comincia ad avere un comportamento saltuariamente scorretto. E' necessario che l'agent permetta di impostare soglie di allarme, e sia in grado di riportare l'evento di superamento delle soglie. In tal modo il gestore del sistema può intervenire con una diagnosi mirata e intervenire prima che il componente si

guasti totalmente.

Una efficace interfaccia utente è necessaria per il fault monitoring ancora più che per altre funzioni di gestione, in primo luogo perché vi è una grande quantità di eventi e di segnalazioni da esaminare: senza avere la possibilità di filtrare gli eventi rilevati diventa veramente difficile capire quali sono le informazioni utili alla diagnosi. La capacità di imporre soglie sui valori di allarme e di effettuare almeno in parte delle correlazioni fra eventi è quindi particolarmente utile, per ridurre la quantità di eventi e segnalazioni che il gestore deve esaminare. In secondo luogo, la individuazione e la diagnosi degli errori richiede una grande cooperazione fra sistema e operatore, e questa cooperazione deve essere supportata e favorita dalla interfaccia utente.

2.4. Accounting Monitoring (controllo della contabilizzazione)

Contabilizzare richiede in ultima analisi la capacità di tenere traccia dell'uso che gli utenti fanno delle risorse. I requisiti sulla contabilizzazione variano moltissimo a seconda degli ambienti e delle finalità per cui si esegue questa contabilizzazione:

- In alcuni ambienti, tipicamente nelle aziende private, è sufficiente *valutare l'uso complessivo delle risorse* e *in quale proporzione devono essere addebitate* ai vari dipartimenti dell'azienda
- In altri ambienti, il controllo sui costi è più stringente e quindi si richiede di *valutare l'utilizzo per centro di costo, progetto o addirittura per user-id*, in modo che i costi possano essere "ribaltati" con maggiore livello di visibilità e di rendicontazione.
- Negli ambienti in cui invece si ha una vera e propria fornitura di servizio, occorre raccogliere dati che permettano di *fatturare* il costo dell'uso delle risorse al singolo utente del sistema.

Quali siano le risorse di interesse è anche esso un tema che viene sviluppato in modo diverso nei diversi ambienti e a seconda del "servizio" di cui ci si interessa. Esempi di risorse possono essere:

- Servizi di comunicazione
- Hardware
- Software e sistemi
- Servizi

Per ogni tipo di risorsa, e a volte per ogni singola risorsa, occorre raccogliere dati di contabilizzazione che variano molto da organizzazione a organizzazione, e che talvolta possono essere raccolti addirittura utente per utente. Se definiamo come "transazione" il singolo uso del sistema che siamo interessati a contabilizzare, esempi di dati da raccogliere:

- Identificazione dell'utente
- Ricevitore (l'altra entità della transazione)
- Numero di pacchetti, o dimensione della transazione
- Livello di sicurezza richiesto
- Timestamp
- Codici di stato della transazione (errori o comunque esito della transazione)
- Risorse usate nella transazione

E' evidente che i dati da raccogliere possono facilmente raggiungere dimensioni assai rilevanti, e che quindi devono essere periodicamente ridotti di dimensione mediante opportune aggregazioni,

concordate con il sistema di contabilizzazione a cui devono essere trasferiti, oppure tale riduzione deve lasciata al sistema di contabilizzazione a cui comunque vanno trasferiti periodicamente.