Misura di tempi di latenza in reti industrial Ethernet

Gianluca Cena, Ivan Cibrario Bertolotti,
Adriano Valenzano

Le reti Ethernet costituiscono da tempo la soluzione più diffusa per le comunicazioni nell'ambito dei sistemi di automazione d'ufficio. È opinione comune, tuttavia, che tale tipo di reti possa essere adottato con successo anche a livello di cella e di campo nei sistemi di automazione di fabbrica, per l'interconnessione di controllori in logica programmabile, attuatori e sensori. Nell'ottica di un utilizzo di Ethernet per supportare lo scambio di dati in tempo reale negli impianti di produzione, è fondamentale disporre di strumenti e metodologie per valutare e misurare le prestazioni effettivamente consentite da tale tecnologia di comunicazione. Questo articolo descrive alcune esperienze relative allo sviluppo di uno strumento per l'analisi del traffico in tempo reale in reti Ethernet che si basa su una piattaforma Linux convenzionale non in tempo reale. Tale strumento ha consentito di valutare, con un livello di accuratezza adeguato, diversi indici di prestazioni inerenti la comunicazione nei sistemi reali e, in particolare, le latenze di trasmissione dei messaggi.

Nello scorso decennio la popolarità delle reti Ethernet è cresciuta senza soluzione di continuità, al punto che esse sono oggi la tecnologia standard per interconnettere dispositivi nel contesto dell'automazione d'ufficio. A causa della tecnica di accesso al mezzo trasmissivo utilizzata, tuttavia, Ethernet è stata a lungo ritenuta inadeguata per l'automazione di fabbrica ed il controllo dei processi.

Il meccanismo *Carrier Sense Multiple Access with Collision Detection* (Csma/CD) [1] è, infatti, in grado di rilevare ogni collisione che si verifica sul bus. In tale evenienza, la contesa viene risolta costringendo i nodi coinvolti a ritrasmettere i loro messaggi dopo un intervallo di tempo di auto-sospensione casuale, la cui ampiezza viene accresciuta in caso di collisioni ripetute (algoritmo di *exponential back-off*). Ciò implica che, almeno da un punto di vista teorico, non vi sia alcuna garanzia che un dato messaggio venga trasmesso entro un tempo massimo stabilito a priori, situazione spesso inaccettabile in quasi tutti i sistemi di controllo distribuiti. Da un punto di vista pratico, tuttavia, lo scenario si è profondamente modificato negli ultimi anni, grazie all'introduzione di reti Ethernet operanti a 100 Mb/s e di *switch full-duplex*, che virtualmente evitano il verificarsi di collisioni (a meno che non insorgano fenomeni di saturazione delle code interne agli *switch* stessi).

Al giorno d'oggi è opinione diffusa che le reti Ethernet possano essere impiegate con successo anche negli ambienti di produzione automatizzati, sia a livello di cella sia a livello di campo. Velocità di trasmissione elevate e l'uso di switch in luogo di *hub*, tuttavia, non sono sufficienti a garantire un comportamento deterministico della rete, a meno che questa (e, in particolare, la schedulazione dei messaggi) sia stata progettata in modo adeguato a prevenire la saturazione delle code.

In questo contesto è di fondamentale importanza essere in grado di effettuare misurazioni sulle effettive prestazioni della rete, sia in fase di progetto sia quando il sistema è operativo (per scopi di diagnostica). Sono state definite, pertanto, opportune metodologie di test e realizzati diversi strumenti per la misura degli indici di prestazione di reti Ethernet, che si basano in genere su hardware specializzato ed hanno costi non trascurabili. In tale ambito, la disponibilità di strumenti a basso costo basati su hardware standard e software *open source* è sicuramente auspicabile, soprattutto per il progetto e lo sviluppo di piccoli impianti, dove i progettisti sono vincolati da limitazioni di budget che rendono difficile l'uso di strumenti professionali.

Questo articolo presenta un'attività di ricerca, attualmente in corso di svolgimento presso l'Istituto Ieiit-Cnr, che ha lo scopo di definire una metodologia per il test di reti *industrial Ethernet*. A tal fine è stato sviluppato uno strumento prototipale, che si basa su una piattaforma Linux convenzionale senza particolari caratteristiche di tempo reale. Lo strumento ha struttura molto semplice e, aspetto ancora più importante, può essere facilmente espanso e personalizzato per misurare ulteriori indici di prestazione oltre a quelli presi in considerazione nel presente lavoro.

Il comportamento e le principali limitazioni legate all'uso dello strumento realizzato sono state analizzate innanzitutto da un punto di vista prettamente teorico, e quindi validate per mezzo di un caso di studio reale semplice e ben definito. I ri-

G. Cena, I. Cibrario Bertolotti, A. Valenzano – Ieiit/Cnr, Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni/Consiglio Nazionale delle Ricerche, Torino

sultati confermano che è possibile raggiungere un livello ragionevole di accuratezza, sufficiente in molti casi pratici, a costi del tutto trascurabili.
L'articolo è strutturato nel modo seguente: il prossimo paragrafo descrive le modalità con cui è possibile misurare le prestazioni di una rete Ethernet ed introduce la struttura del nostro sistema di misura, mentre il terzo paragrafo ne analizza il comportamento ed evidenzia le principali limitazioni. L'ultimo paragrafo, infine, riporta i risultati ottenuti nell'analisi di uno switch industriale commerciale condotta con il nostro strumento.

Misura delle prestazioni delle reti Ethernet

La misura delle prestazioni offerte da una rete è sicuramente un argomento di grande interesse ed uno dei temi più trattati, anche di recente, nella letteratura specializzata. Nella maggior parte dei casi, tuttavia, il termine "tempo reale" viene riferito alla capacità di soddisfare in modo adeguato la trasmissione di flussi di informazione multimediali.
L'*Internet Engineering Task Force* (Ietf) ha standardizzato tanto la terminologia quanto la metodologia per la valutazione delle prestazioni di una rete, considerando aspetti legati sia ai dispositivi per l'interconnessione di reti (*router*) [2], [3] sia a quelli per il *Lan switching* [4], [5]. In particolare è stato introdotto il concetto di *system under test* (Sut) per modellare la rete e gli apparati di comunicazione, e si sono definite linee guida non ambigue per individuare quali sono gli indici prestazionali significativi e come devono essere misurati. Nel seguito si tenterà, per quanto possibile, di seguire tali indicazioni, ma occorre tener presente che esse riguardano principalmente lo scambio di informazioni a livello Tcp/Ip, mentre in questo lavoro l'attenzione è focalizzata a livello inferiore, ovvero sull'interfaccia Ethernet (in questo modo è possibile valutare separatamente il contributo della pila protocollare e del sistema di trasmissione, ottenendo risultati meno generali e quindi più significativi).
Altri lavori apparsi in letteratura, come ad esempio [6], [7] e [8], trattano principalmente di sistemi di misura passiva per reti. In questo caso il traffico viene solamente osservato, e l'attenzione è posta sulle tecniche per prevenire interferenze fra il sistema di misura ed il sistema sotto osservazione. Ciò è utile per generare file di log che possono poi essere analizzati fuori linea.

Il sistema di test
Il sistema di test sviluppato è basato su PC standard operanti su piattaforma Linux. Nel nostro caso il PC utilizza un processore Amd Athlon XP 2100+ a 1733 Mhz e 1 GB di memoria Ram. Il sistema operativo utilizzato è una distribuzione standard Debian Linux (versione del kernel 2.4.24). Come mostrato nella figura 1, il sistema di test deve essere dotato di due schede Ethernet distinte (nel nostro caso, 3Com 3C905C-TX e Realtek Rtl 8139 onboard), usate rispettivamente per spedire e per ricevere i messaggi di test. I due cavi dello strumento (sonde) devono essere collegati alle due porte del Sut che si intendono analizzare. Il programma di test genera un traffico rappresentato da un pattern programmabile di messaggi, trasmesso tramite la prima interfaccia Ethernet (Nic1), e, al tempo stesso, cattura i messaggi in arrivo sulla seconda interfaccia (Nic2).
Ogni pacchetto trasmesso è individuato da un identificatore unico (codificato come un intero su 4 byte, più che adeguato per l'uso con sequenze di trasmissione la cui lunghezza può arrivare al miliardo di pacchetti), in modo tale da poter essere identificato in modo univoco in fase di ricezione. Ogni evento rilevante per la comunicazione ed osservabile a livello utente, inoltre, viene etichettato con un opportuno *timestamp*. In questo modo è possibile collezionare dati statistici inerenti diverse grandezze temporali che si riferiscono al tragitto dei pacchetti all'interno del Sut.
Il sistema di test può essere utilizzato per misurare molti parametri, che includono i ritardi di trasmissione, la percentuale di pacchetti persi, la banda effettivamente disponibile, e così via. Nella fase iniziale del progetto, tuttavia, ci si è limitati a misurare le latenze di trasmissione dei messaggi, dal momento che tale parametro costituisce certamente uno degli aspetti più significativi per caratterizzare una rete che connette dispositivi in sistemi di controllo distribuiti, ed influenza direttamente il comportamento in tempo reale dell'intero sistema.

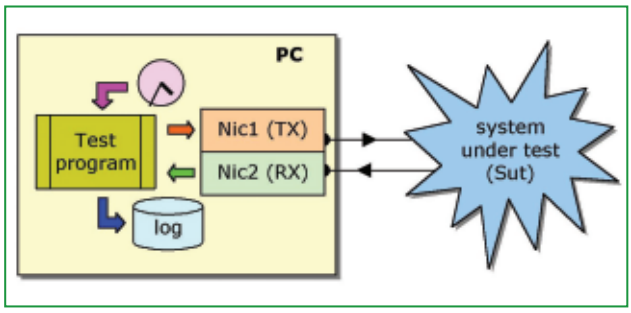

Figura 1 - Struttura del sistema di test

Misurazione dei tempi in Linux
Uno dei principali problemi da risolvere in questo tipo di misure è quello di disporre di un mezzo che permetta di rilevare valori di tempo assoluti. A tal fine si è deciso di utilizzare la funzione standard *gettimeofday()*, che restituisce il tempo trascorso (espresso in secondi e microsecondi) da un istante fisso e noto a priori.
Poiché il programma di test è stato sviluppato per operare su piattaforma Linux convenzionale, il processo di misura può essere interrotto occasionalmente da routine di servizio di interrupt e task del kernel a priorità più elevate, che influenzano i tempi misurati in modo imprevedibile. È stato, pertanto, effettuato un primo set di esperimenti volto a valutare quanto gli eventi legati all'hardware del dispositivo ed al kernel siano in grado di influenzare il comportamento (e la precisione) del sistema di test.
Il breve frammento di codice riportato di seguito è stato scritto con lo scopo di misurare la distribuzione statistica della durata di tali eventi.

```
gettimeofday(&old, NULL);
for(i=0; i<N_SAMPLES; i++) {
       while(1) {
              gettimeofday(&new, NULL);
              if((step = TimeDiff(&new, &old)) > 0)
                     break;
              }
       if(step>=N_TOD_STEP)
              step=N_TOD_STEP-1;
       tod_count[step]++;
       old = new;
       }
```

In pratica viene effettuato un ciclo nel quale viene valutato il tempo intercorrente fra chiamate successive alla *gettimeofday()*, scartando i valori nulli (che indicano che la durata del ciclo cade sotto la risoluzione permessa dal programma pari ad 1 µs). L'istogramma di tali valori è prima memorizzato in un vettore e quindi analizzato fuori linea. Tale approccio (mantenere, cioè, il ciclo di misura il più semplice ed efficiente possibile ed effettuare tutti i calcoli fuori linea a test concluso) è stato adottato durante lo sviluppo dell'intero programma, in modo da minimizzare l'*overhead* introdotto dal sistema di misura.
Nella figura 2 sono mostrati i diagrammi di dispersione relativi a due diverse condizioni operative e, in particolare, a: un computer sostanzialmente "scarico", dove il sistema grafico *X Window System* (X11) è abilitato; un computer "scarico" con il sistema X11 disabilitato. Tali diagrammi permettono di avere un'indicazione dei principali eventi che influenzano il comportamento del programma di test, quali ad esempio accessi concorrenti al disco, paginazione, sistema grafico X11 ecc., e di quanto ciascuno di essi incide sul processore.
La durata delle "interferenze" è solitamente dell'ordine dei microsecondi, ma può arrivare anche ad alcuni millisecondi; questi ultimi eventi, tuttavia, sono relativamente rari, come dimostra il picco prominente negli istogrammi in corrispondenza del valore di ritardo pari a 1 µs. Nella maggior parte dei casi, quindi, non esiste effetto apprezzabile sulle misure temporali effettuate dal programma di test, e la risoluzione ottenibile raramente peggiora in modo significativo. Ciò significa che *gettimeofday()* è in grado di offrire "statisticamente" una risoluzione pari a 1 µs.

Struttura del programma di test
Le misure sono state effettuate per mezzo di un programma basato su un singolo *thread*. Inizialmente è stata considerata (ed implementata) anche una soluzione basata su due thread, uno per gestire la trasmissione e l'altro per la ricezione. L'accuratezza delle misure si è rivelata peggiore nel secondo caso, in quanto la risoluzione standard dei timer in Linux (10 ms) è decisamente inferiore alla risoluzione del clock.
Il programma consta di un singolo ciclo che determina il tempo corrente per mezzo di una chiamata a *gettimeofday()* e quindi effettua le azioni appropriate (si veda la figura 3). In particolare, viene inviato un nuovo pacchetto non appena il tempo corrente risulta essere maggiore del tempo di generazione previsto per il prossimo messaggio. A tale scopo è stata utilizzata una chiamata alla funzione *send()*. Ogni qualvolta viene spedito un pacchetto, il tempo di trasmissione viene memorizzato in un buffer circolare insieme all'identificatore univoco del pacchetto. Il contatore che gestisce gli identificatori, inoltre, viene incrementato di una unità ed il tempo di generazione del prossimo messaggio viene aggiornato.
Viene poi effettuata una chiamata non bloccante alla funzione *recv()*, che controlla se è stato ricevuto o meno un pacchetto e, in caso di ricezione, vengono aggiornati i relativi dati statistici. L'identificatore presente nel pacchetto è usato per determinare l'istante di trasmissione di ogni pacchetto ricevuto (tale informazione viene reperita nel buffer circolare).
Nella prima fase del progetto si è ipotizzato che l'ordine di trasmissione dei pacchetti non possa essere modificato dalla rete (questo implica che nel Sut non sono ammesse connessioni ridondate). In questo modo è stato possibile mantenere il codice di gestione del buffer circolare semplice e molto veloce.
Dal momento che il corpo del ciclo viene eseguito in un tempo molto breve quando il sistema è inattivo (cioè quando nell'iterazione corrente non viene né trasmesso né ricevuto alcun messaggio), l'accuratezza della misura dei tempi è

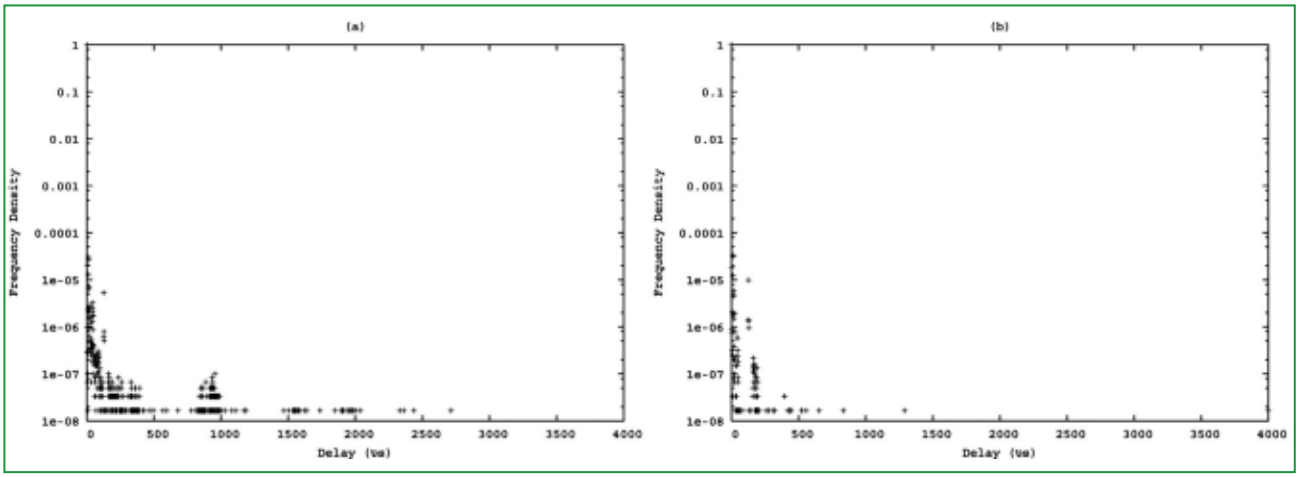

Figura 2 - Misurazione delle interferenze

dell'ordine di alcuni microsecondi.
Obiettivo del lavoro è stato anche quello di misurare le prestazioni della rete a livello fisico (o, perlomeno, a livello Mac). Al fine di evitare che le latenze introdotte dallo *stack* protocollare Tcp/Ip influenzassero le misure sul Sut, si è deciso di operare direttamente a livello Ethernet utilizzando i cosiddetti *raw socket*. A differenza dei socket convenzionali, questi permettono ai programmi operanti in modo utente di accedere direttamente al livello data-link del protocollo di comunicazione. In particolare si sono utilizzati pacchetti Ethernet il cui campo *ethertype* è settato al valore 6006 (riservato in origine per uso privato da Dec).

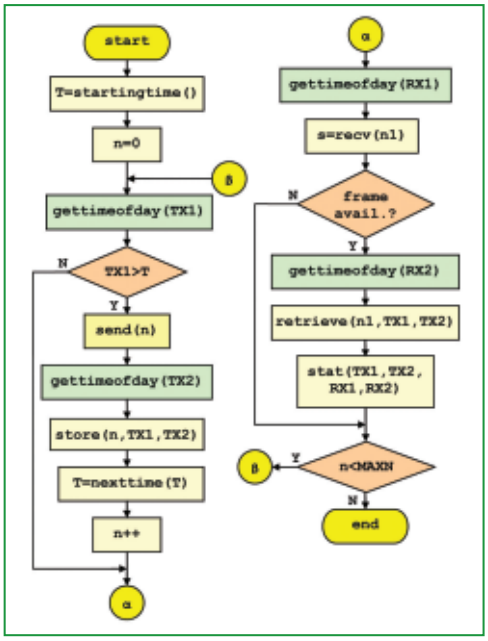

Figura 3 - Programma di test

Pattern di generazione dei messaggi
Il *pattern* di generazione implementato è di tipo periodico, tuttavia, nel caso in cui il tempo di trasmissione previsto non possa essere rispettato a causa dei *jitter* introdotti dalla precedente iterazione (ciò può essere ad esempio dovuto ad un traffico in rete troppo elevato o ad un'interferenza del kernel o dell'hardware), la trasmissione del messaggio è posticipata all'inizio del periodo successivo.
Il sistema può così generare un traffico quanto più possibile periodico (il traffico generato medio si mantiene costante anche nel breve-medio termine), ad eccezione degli intervalli di tempo in cui le trasmissioni pianificate vengono posticipate.
Occorre, infine, notare come il blocco di generazione possa essere facilmente esteso in modo tale da permettere la generazione di traffico *burst* o distribuito in modo *poissoniano*.

Tecnica di misura

La struttura complessiva dello strumento di test sviluppato è adatta per la valutazione di numerosi parametri. Il primo insieme di misure effettuate sulla rete riguarda la latenza dei messaggi, ovvero il tempo che intercorre fra la trasmissione di un pacchetto e la sua ricezione. I diversi ritardi che compongono la latenza totale per lo strumento di test sono indicati nella figura 4.

Analisi del sistema
I controller Ethernet utilizzati, sebbene economici, sono alquanto sofisticati. In particolare, essi possono operare come bus master Pci, il che implica che i messaggi non vengano effettivamente copiati dalla memoria del kernel al Nic (e viceversa). Al contrario, i Nic possono accedere direttamente ai messaggi parcheggiati nella memoria kernel in strutture dati apposite (Sk_Buff) organizzate come un buffer circolare. In questo modo è possibile ridurre in misura significativa le latenze all'interno del Nic e, grazie alle funzionalità avanzate di cui sono dotati i Nic moderni, non è neppure necessario effettuare operazioni di copia per incapsulare o estrarre i dati utente nei/dai pacchetti.
Nella chiamata a *send()*, il messaggio viene prelevato dalla memoria utente ed accodato in un buffer di trasmissione (Tx_Sk_Buff) localizzato nella memoria kernel. Da qui, esso viene trasferito allo *shift register* di output del Nic1 per mezzo di operazioni in Dma. Tutti i bit costituenti il pacchetto, inclusi gli 8 byte di preambolo, sono quindi spediti serialmente sulla porta di uscita e, dopo un tempo che dipende dai ritardi introdotti dal Sut (in particolare dal ritardo di propagazione nei conduttori e nei *transceiver*, nonché dal tempo speso negli switch) giungono alla porta di ingresso della sezione di ricezione. Qui il pacchetto viene dapprima copiato (tramite operazioni in Dma) dallo shift register di ingresso del Nic2 in un buffer di ricezione adatto (Rx_Sk_Buff) localizzato nella memoria kernel, quindi viene generato un interrupt.
La routine di servizio dell'interrupt notifica l'arrivo del pacchetto allo stack protocollare che, a sua volta, effettua il *demultiplexing* sulla base del tipo del pacchetto (il campo ethertype viene utilizzato a questo scopo) e trasferisce il messaggio nella coda di ricezione del socket cui esso è destinato.
La chiamata successiva (non bloccante) a *recv()*, effettuata dal programma di test, può quindi avere successo ed il messaggio è copiato nella memoria utente.

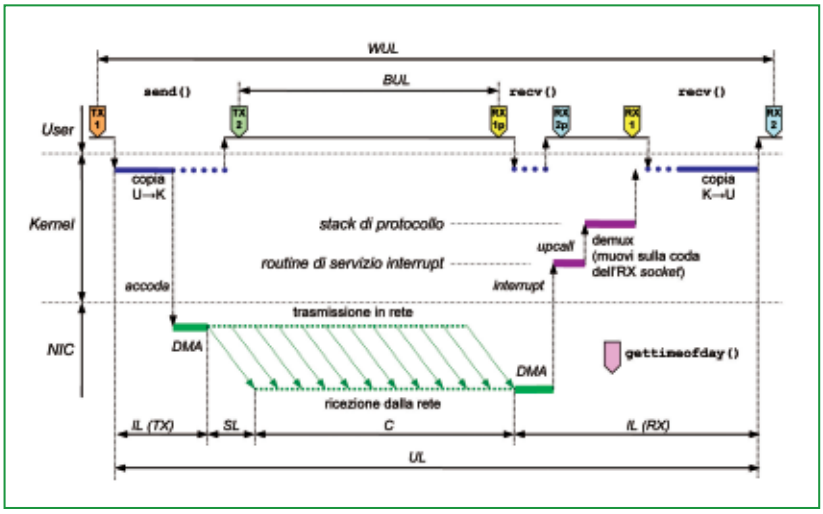

Figura 4 - Definizione di latenza interna

Metodologia di test

Nel caso in oggetto il Sut è una rete Ethernet composta di cablaggi, hub e switch interconnessi. Si è, inoltre, interessati alla latenza del Sut (Sut Latency, SL), che esprime il ritardo intercorrente fra l'istante in cui il primo bit del pacchetto entra nel Sut tramite una porta di trasmissione e l'istante in cui il medesimo bit esce dalla porta di ricezione considerata.

A causa delle prestazioni "limitate" del nostro strumento "economico", le misure sono influenzate anche da ritardi addizionali. Se si indica con UL (user latency) la latenza osservata dai programmi utente (cioè, a livello socket), il ritardo sofferto da un pacchetto trasmesso dal sistema di test è dato da:

$$UL = C + IL + SL \quad (1)$$

dove IL (internal latency) è la latenza interna dello strumento di test (che include i ritardi causati dal ciclo principale del programma utente, il tempo trascorso in modalità kernel e le latenze tipiche dell'hardware) mentre C è il tempo impiegato per trasmettere serialmente sul bus tutti i bit costituenti il pacchetto. In pratica IL comprende quattro termini:

$$IL = T_{PROC} + T_{COPY} + T_{LOOP} + T_{INT} \quad (2)$$

dove T_{PROC} è la parte fissa del tempo di elaborazione relativo alla trasmissione/ricezione del pacchetto, T_{COPY} è il tempo richiesto per copiare i buffer in memoria (dalla memoria utente al kernel e quindi al controller di rete per mezzo di trasferimenti in Dma, e viceversa) e dipende dalla dimensione dei dati utente, T_{LOOP} esprime l'incertezza legata al fatto che il socket di ricezione viene controllato con una cadenza limitata per determinare l'arrivo dei pacchetti, mentre T_{INT} è legato alle interferenze (perlopiù non predicibili) introdotte dall'hardware e dai task del kernel.

C, al contrario, è una grandezza strettamente deterministica è può essere calcolata a partire dalla dimensione P dei dati utente (in byte) e il bit rate utilizzato sulla rete (100 Mb/s):

$$C = (8 + \text{Max}[\,64\,,\,18 + P\,]) \cdot 8 \cdot 0{,}01\ \mu s = \text{Max}[\,5{,}76\ \mu s\,,\,2{,}08\ \mu s + P \cdot 0{,}08\ \mu s/\text{byte}\,] \quad (3)$$

È bene ricordare, infatti, che i pacchetti Ethernet sono preceduti da un preambolo su 64 bit e che i pacchetti più corti di 512 bit sono "allungati" fino a raggiungere questa dimensione minima.

Il valore di SL per ogni dato Sut può essere valutato in modo semplice allorché UL, IL e C siano conosciuti: UL è misurato di volta in volta variando le condizioni operative del Sut; IL viene determinato in fase di calibrazione del sistema; C è calcolato direttamente a partire dai parametri del test.

Nel fare questo, si è assunto implicitamente che IL non dipenda dal particolare Sut. Tale ipotesi si dimostra corretta con ottima approssimazione a patto che il traffico generato dallo strumento di test sia basso, perché, in tal caso, i pacchetti non vengono mai accodati nel sistema di test (eventualmente lo sono nel Sut).

Il principale problema nell'uso dello strumento realizzato è l'impossibilità di determinare con esattezza l'istante in cui i pacchetti sono trasmessi o ricevuti, dal momento che queste azioni non possono essere registrate con accuratezza nel caso in cui si utilizzi un sistema software e per di più non in tempo reale. È possibile, invece, misurare in modo semplice le finestre temporali entro le quali hanno luogo la trasmissione o la ricezione di un pacchetto. In particolare, ogni chiamata a *send()* è preceduta e seguita da chiamate a *gettimeofday()*, che forniscono un intervallo temporale [TX1, TX2] nel quale la trasmissione ha sicuramente inizio (TX-window). In modo analogo è possibile misurare un intervallo [RX1, RX2] che include *recv()* nel quale avviene con certezza la ricezione del pacchetto (RX-window).

Per ogni dato pacchetto k scambiato in rete, la differenza fra $RX2_k$ (misurato subito dopo la chiamata a *recv()* che legge il pacchetto k) e $TX1_k$ (che precede la chiamata a *send()* che trasmette il pacchetto k) fornisce un limite superiore alla latenza del pacchetto stesso (worst-case user latency, Wul_k).

Analogamente, il tempo che trascorre fra gli istanti $TX2_k$ e $RX1p_k$ (misurato, quest'ultimo, prima dell'ultima chiamata *recv()* che non ha avuto esito positivo, ovvero quella che precede la lettura del pacchetto k) fornisce un'indicazione significativa e "ragionevole" del limite inferiore della latenza (best-case user latency Bul_k). Definiamo tale grandezza "ragionevole" poiché non è possibile conoscere il tempo esatto impiegato dal kernel e dal Nic per iniziare a servire una richiesta di trasmissione pendente o per notificare una condizione di interrupt alla Cpu ed eseguire lo stack protocollare.

Tuttavia, poiché il sistema è basato su Nic ad alte prestazioni ed il codice dello stack di protocollo è altamente ottimizzato, è possibile trascurate tali ritardi ed assumere che Bul rappresenti un limite inferiore effettivo per la latenza. È quindi possibile definire una finestra temporale [Bul_k, Wul_k] per la trasmissione di ogni singolo pacchetto k tale che:

$$Bul_k < UL_k < Wul_k \quad (4)$$

Misurazioni effettuate sul Sut mostrano che Bul_k e Wul_k sono assimilabili a variabili casuali. Un parametro interessante è il loro valor medio (mean user latency, Mul), definito come:

$$Mul_k = \frac{Bul_k + Wul_k}{2} \quad (5)$$

Tale parametro può essere usato per stimare con buona approssimazione la latenza a livello utente UL del pacchetto k. Infatti, dalle prove effettuate si è visto come UL_k e Mul_k differiscano per un offset costante ΔL che, seppur difficilmente valutabile, non dipende in prima approssimazione dalla dimensione dei pacchetti o dal Sut analizzato ma solamente dallo strumento di test:

$$Mul_k \approx UL_k + \Delta L \quad (6)$$

É bene notare che il contributo delle interferenze T_{INT} a Mul_k è in media nullo, dal momento che le interferenze possono "dilatare" Wul_k oppure "accorciare" Bul_k.

Uso dello strumento

Lo strumento deve essere calibrato una sola volta prima dell'utilizzo, dopo di che può essere collegato al Sut per la misura degli indici di prestazione. È bene notare che la procedura di calibrazione può essere eseguita in modo completamente automatico.

Calibrazione
Al fine di calibrare lo strumento di misura è necessario effettuare una serie di misure di latenza, dopo aver collegato insieme le due porte dello strumento per mezzo di un cavo (possibilmente corto). In questo modo le componenti della latenza che dipendono dal Sut (SL) sono ridotte a 0, ed il valore di latenza misurato a livello utente (UL0) corrisponde al tempo impiegato per trasmettere in modo seriale ogni bit del pacchetto (C) più il ritardo interno complessivo dello strumento di test (IL). In pratica, quindi:

$$Bul0_k < IL_k + C_k < Wul0_k \quad (7)$$

Come evidenziato in precedenza, è utile considerare il valore medio aritmetico della latenza interna (mean internal latency, Mil) definito come:

$$Mil_k = \frac{Bul0_k + Wul0_k}{2} - C_k \quad (8)$$

Valutando la media statistica delle latenze a livello utente per un numero molto elevato di pacchetti scambiati, è possibile ottenere una stima soddisfacente della vera latenza interna, che può quindi essere usata per calibrare lo strumento. In particolare, si consideri la media statistica di Mil_k, calcolata per pacchetti della stessa dimensione P:

$$Amil(P) = \frac{1}{N_{tot}} \cdot \sum_{k=1}^{Ntot} Mil_k \quad (9)$$

dove N_{tot} è il numero dei pacchetti considerati in tale misura. Per il teorema del limite centrale, è possibile affermare che, sotto blande condizioni ed a prescindere dalla reale distribuzione dei valori Mil_k, Amil è caratterizzata da una distribuzione normale. Inoltre, per quanto detto prima essa è influenzata dalle interferenze in modo trascurabile.
In generale, Amil dipende dalla dimensione P dei dati utente inclusi nei pacchetti di test e il tasso di generazione di tali pacchetti Γ. Amil è stato misurato per differenti valori di P e, come prevedibile, si è constatato che la dipendenza è esprimibile tramite una legge lineare:

$$Amil(P) = \alpha + \beta \cdot P \quad (10)$$

dove α tiene conto della parte fissa del tempo di esecuzione del programma di test (che include T_{PROC} e T_{LOOP}) mentre β rappresenta l'overhead addizionale per byte dovuto alle operazioni di copia dei dati dalla memoria utente al kernel e viceversa (β riguarda fondamentalmente il termine T_{COPY}). Si è anche osservato che il tasso di generazione Γ è in pratica irrilevante, a condizione che il traffico prodotto dallo strumento di test sia mantenuto sufficientemente basso (meno della metà della banda disponibile).
Per stimare i valori di α e β si è usata una semplice analisi di regressione lineare per interpolare i valori misurati con una linea retta a + b · P con il metodo dei minimi quadrati, dove a e b sono rispettivamente le stime di α e β derivate dai campioni.
La figura 5 mostra gli istogrammi di Mil per diversi valori di P e la corrispondente retta di regressione ottenuta coi minimi quadrati nel caso in cui Γ = 10%.

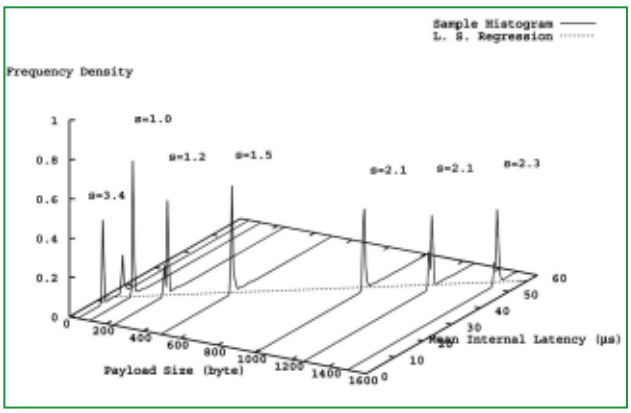

Figura 5 - Istogrammi della latenza interna media e risultati della regressione

La tabella 1 mostra i valori di a e b, insieme alle loro deviazioni standard s_a e s_b, per i due valori di Γ considerati nell'esperimento.

Tabella 1

Γ	a	s_a	b	s_b
10%	11,73	0,376	0,028	6,95E-4
50%	12,69	0,442	0,027	8,17E-4

Parametri della retta interpolante

Infine, è stato calcolato l'intervallo di confidenza per α and β corrispondente ad un livello di confidenza del 95%, nell'ipotesi che a e b seguano una distribuzione t di *Student* con cinque gradi di libertà (si sono effettuate misure per sette diversi valori di P) e usando la varianza residua come una stima di σ^2 delle misure di Amil (si veda la tabella 2).
La grandezza Amil può essere usata per stimare il contributo IL_k al tempo di trasmissione del pacchetto k dovuto alla latenza interna dello strumento di test, data la dimensione

Tabella 2

Γ	α (μs)	β (μs/byte)
10%	11,73±0,97	0,028±0,0017
50%	12,69±1,14	0,027±0,0021

Intervalli di confidenza su α e β

P_k del *payload* del messaggio usato per la misura. In particolare, essi differiscono per un offset approssimativamente costante ΔL, che è lo stesso osservato in sede di valutazione della latenza a livello utente:

$$Amil(P_k) \approx IL_k + \Delta L \quad (11)$$

Misure

Lo strumento s/w è stato usato per misurare la latenza media di un Sut particolarmente semplice, ovvero costituito da un singolo switch, e il cui comportamento è in qualche modo noto a priori. In particolare è stato analizzato un componente di tipo *Hirschmann* Spider 5TX [9], che opera secondo il principio *store* and *forward*, dotato di 5 porte 10/100Base-TX (socket RJ45) con funzionalità di *auto-crossing*, auto-negoziazione e auto-polarità. Più in dettaglio è stata misurata la latenza utente per un numero molto elevato di pacchetti al variare della dimensione del pacchetto e del tasso di generazione. L'indice di prestazione considerato è ancora una volta Mul che, nel caso in oggetto, risulta essere il più significativo. Da Mul è possibile calcolare la latenza media introdotta dal Sut (mean Sut latency, Msl), definita come

$$Msl_k = Mul_k - Amil(P_k) - C_k \quad (12)$$

Msl_k può essere usato per stimare direttamente la latenza SL_k introdotta dal Sut, in quanto il contributo delle interferenze su Msl è in media nullo. Inoltre, anche se si ipotizza di non disporre di alcuna informazione sulle latenze interne dello strumento di test, queste ultime influenzano nello stesso modo sia

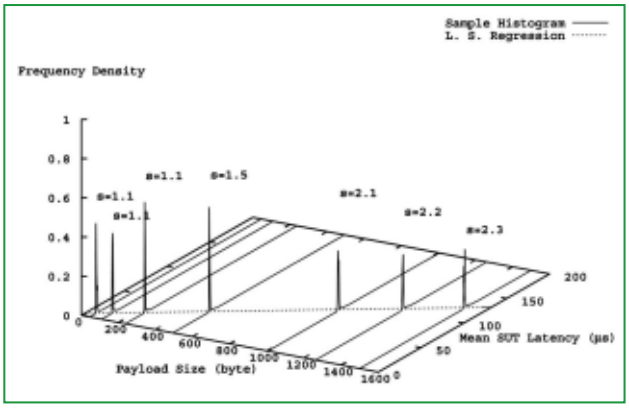

Figura 6 - Istogrammi della latenza media del Sut e risultati della regressione

Mul0 (valutato in fase di calibrazione) sia Mul (misurato quando il Sut viene connesso allo strumento di test). Nel calcolo di Msl_k, quindi, il contributo delle latenze interne viene, di fatto, annullato. La statistica sulla latenza media del Sut è stata ottenuta dalla distribuzione dei valori Msl_k: gli istogrammi di Msl e la corrispondente regressione lineare basata sul metodo dei minimi quadrati sono mostrati nella figura 6 per Γ=10%.

Anche per Msl si può ipotizzare una dipendenza lineare dal tempo di trasmissione C del pacchetto:

$$Msl(C) = \alpha' + \beta' \cdot C \quad (13)$$

Utilizzando la stessa tecnica adottata nella fase di calibrazione è possibile caratterizzare i valori α' e β' relativi al Sut. I relativi risultati sono riportati in tabella 3.

Tabella 3

Γ	α' (μs)	β' (μs/byte)
10%	4,13±1,09	0,080±0,0020
50%	5,51±2,51	0,080±0,0046

Intervalli di confidenza su α' e β' per il Sut

Il coefficiente di regressione lineare ottenuto dalle misure indica chiaramente come Msl dipenda linearmente dal tempo di trasmissione C del pacchetto, ed il valore trovato di β' è molto prossimo al tempo impiegato per trasmettere un byte in rete (80 ns). Ciò è in perfetto accordo con il funzionamento dello switch, che memorizza prima l'intero pacchetto ricevuto sulla porta di ingresso e lo inoltra quindi sulle porte di uscita.

La parte fissa di Msl (da 4 a 5 μs circa) riflette, infine, il tempo di elaborazione interno dello switch, quantità temporale non valutabile in modo semplice senza ricorrere a misure del tipo descritto in precedenza. L'accuratezza molto elevata ottenuta nella caratterizzazione dello switch è legata al fatto che nell'esperimento effettuato si è valutata statisticamente una grandezza mediando un elevato numero di misure. Anche nel caso in cui lo strumento di test venga utilizzato per misurare la latenza dei singoli pacchetti (come ad esempio quando si analizza la risposta di un sistema di switch in regime transitorio), tuttavia, è lecito aspettarsi una precisione che, nella maggior parte dei casi è dell'ordine dei 5 μs o migliore.

Conclusioni

Ethernet è la tecnologia di comunicazione più diffusa per l'automazione di ufficio, e sta rapidamente diventando popolare anche negli ambienti di automazione di fabbrica. In questo articolo è stato presentato un progetto volto a definire tecniche a bassissimo costo per misurare le prestazioni di una rete Ethernet e per verificare il soddisfacimento dei requisiti di tempo reale nelle comunicazioni che hanno luogo in impianti di produzione automatizzati. In particolare è stato sviluppato

uno strumento s/w che si basa su un PC convenzionale e una distribuzione di sistema operativo Linux standard, e presenta l'unico requisito addizionale di disporre di due porte Ethernet per la connessione al Sut. Lo strumento è stato quindi utilizzato per misurare la latenza di uno switch industriale reperibile sul mercato.
I risultati ottenuti mostrano che l'accuratezza dello strumento è dell'ordine di alcuni microsecondi, un valore più che adeguato in molti sistemi reali, nei quali i requisiti temporali sulle tempistiche degli scambi in rete non sono particolarmente critici. Infine, se si considera che la versione 2,4 del kernel di Linux non è stata ottimizzata per le operazioni in tempo reale, il sistema di test realizzato si comporta meglio di quanto ci si possa attendere, con una latenza dello stack di protocollo che nella macchina considerata è di soli 12 µs, includendo le transizioni fra la modalità utente e quella kernel.

Bibliografia

[1] *Ieee Standard For Information Technology – Telecommunications And Information Exchange Between Systems – Local And Metropolitan Area Networks – Specific Requirements Part 3: Carrier Sense Multiple Access With Collision Detection (Csma/Cd) Access Method And Physical Layer Specifications*, Institute of Electrical and Electronics Engineers, 2002.
[2] Internet Engineering Task Force Rfc 1242, *Benchmarking Terminology for Network Interconnection Devices*, July 1991.
[3] Internet Engineering Task Force Rfc 2544, *Benchmarking Metodology for Network Interconnection Devices*, March 1999.
[4] Internet Engineering Task Force Rfc 2285, *Benchmarking Terminology for Lan Switching Devices*, February 1998.
[5] Internet Engineering Task Force Rfc 2889, *Benchmarking Metodology for Lan Switching Devices*, August 2000.
[6] S. Donnelly, *High Precision Timing in Passive Measurements of Data Networks*, Ph.D. thesis, University of Waikato, June 2002.
[7] J. Micheel, I. Graham and S. Donnelly, "Precision Timestamping of Network Packets", in *Proceedings of the Acm Sigcomm Internet Measurement Workshop*, San Francisco, California, Usa, November 1-2, 2001.
[8] J. Clearly, I. Graham, T. McGregor, M. Pearson, I. Ziedins, J. Curtis, S. Donnelly, J. Martens, S. Martin, "High Precision Traffic Measurement", *Ieee Communications Magazine*, pp. 167-173, March 2002.
[9] *Spider 5tx Description and Operating Instructions*, Hirschmann Electronics, Order No. 943 824-001, http://hus.hirschmann.com.