ETHERNET REAL-TIME PER CONTROLLO MOTORI

30 luglio 2007

Master in Tecnologia del Software Libero e Open Source

Project Work Finale

Anno Accademico 2005/2006

Autore: Roberto A. Foglietta <me@roberto.foglietta.name>

Relatori: Alessandro Rubini e Tullio Facchinetti dell'Universtità di Pavia

Committente: Paolo Costanzo per Cori Officine Meccaniche

Sommario

Lo scopo di questa tesi è quello di studiare e verificare la possibilità di pilotare dei motori brushless attraverso reti Ethernet real-time con chiusura di ciclo di controllo in un tempo dell'ordine di un centinaio di microsecondi; cioè molto breve rispetto ai tempi di reazione tipici di un sistema operativo desktop.

Il principale vincolo di questo studio è l'utilizzo di elaboratori, schede, cavi e switch/hub di rete comuni cioè quel tipo di materiale facilmente reperibile in commercio. L'uso di prodotti di massa per l'automazione industriale ha il beneficio di ridurre sensibilmente i costi degli impianti stessi ma è uno degli elementi caratteristici che implica la necessità di uno studio dei limiti operativi di questi prodotti e che permetta di verificarne l'adeguatezza e le loro eventuali modalità di impiego.

Il sistema operativo scelto per questo progetto è GNU/Linux perché disponibile sotto una licenza che ne permette liberamente la modifica e la distribuzione. Inoltre perché fra i sistemi operativi rilasciati sotto una licenza con queste caratteristiche GNU/Linux è quello più diffuso in ambito embedded, server e desktop quindi quello su cui è più facile trovare personale già preparato.

Il software sviluppato in seno a questo progetto è stato rilasciato in licenza GNU/GPL mentre per le condizioni di licenza di questo documento si veda l'apposita sezione.

Dedica

Dedico questo lavoro di tesi a Cristina che per molti mesi si è battuta come una tigre per contendersi la mia attenzione altrimenti concentrata sul mio lavoro di consulente, lo studio per il master e l'attività di divulgazione dell'informatica libera.

Ringraziamenti

Ringrazio, senza dimenticare tutti quelli che mi hanno aiutato e accompagnato in questa piccola avventura e che elencarli sarebbe troppo lungo, il mio relatore Alessandro per aver accettato di buon grado il mio personalissimo modo di affrontare e sviluppare questo studio. Ovviamente anche i mie genitori con i quali ancora convivo ogni volta che non sono ingaggiato per lavorare fuori Genova!

Indice

1 Descrizione del progetto

Controllo numerico per movimentazione macchine utensili con diversi motori impiegati per pilotare diversi assi in maniera tale che, non soltanto la posizione finale, ma anche la traiettoria sia prevedibile con una certa precisione.

Il principale vincolo di questo progetto è l'utilizzo di elaboratori, schede, cavi e switch/hub di rete comuni cioè quel tipo di materiale facilmente reperibile in commercio. L'uso di prodotti di massa per l'automazione industriale ha il beneficio di ridurre sensibilmente i costi degli impianti stessi ma è uno degli elementi caratteristici che implica la necessità di uno studio dei limiti operativi di questi prodotti e che permetta di verificarne l'adeguatezza e le loro eventuali modalità di impiego.

1.1 Specifiche dei motori

Tratto dal materiale informativo del produttore Phase Motion Control:

> I motori UL-T sono motori Brushless trifase, sincroni con eccitazione a magneti permanenti, e necessitano di comando e controllo elettronico PWM. Sono caratterizzati da elevatissima densità di coppia, grande campo di variazione della velocità e ampia banda passante. L'avvolgimento statorico porta incorporato una terna di sensori di temperatura PTC a soglia con intervento a 130 °C e un sensore di temperatura lineare KTY84-130. I motori sono equipaggiati con un sensore di posizione (resolver o encoder) utilizzato per rilevare la velocità e la posizione del rotore. Tipi di sensore disponibili:
>
> - Resolver 2 poli
> - Encoder assoluto monogiro con interfaccia EnDat, risoluzione 2^{16} conteggi/giro
> - Encoder assoluto multigiro con interfaccia EnDat, risoluzione 2^{16} conteggi/giro x 4096 giri
>
> I motori in esecuzione normale hanno grado di protezione IP 65, sono adatti per ambienti con temperature fra -20 °C e +80 °C e hanno un livello di rumorosità ridotto.

Un utile documento [1] per la selezione e l'uso di questi motori, insieme a molte altre informazioni, sono disponibili sul sito `http://www.phase.it` del produttore.

Il seguente elenco di termini fornisce informazioni circa alcune definizioni tecniche che altrimenti risulterebbero oscure a coloro i quali non sono intimi del settore dei motori elettrici:

brushless traducibile in "senza spazzole". Per mettere a fuoco le peculiarità di questo tipo di motori è bene fare il contrasto con i motori che invece delle spazzole necessitano. Concettualmente è possibile dividere un motore in due parti: la carcassa con la bobina principale, che si chiama statore, e il rotore con la bobina secondaria. La mutua interazione fra il campo magnetico generato da queste due bobine produce la rotazione dell'albero. L'alimentazione della bobina sul rotore è fornita attraverso delle spazzole di materiale conduttore che strisciano lungo una guida circolare solidale allo statore. L'impiego di tale sistema di alimentazione avrebbe diversi effetti indesiderati fra quali: dissipazione di energia cinetica, produzione di calore, logorio delle spazzole e della guida, relativamente alta resistenza elettrica e ampie fluttuazioni della stessa.

trifase si riferisce al tipo di alimentazione in corrente alternata trifase che è una fornitura tipica di energia per impianti di media potenza, cioè utenze non domestiche oltre i 6 KW/h e tensioni a partire da 380 V.

sincroni Questa tipologia di motori arriva fino a 10 KW di potenza e ha come vantaggi un elevato rapporto potenza/peso, affidabilità, bassa inerzia del rotore e produzione di calore solo sullo statore cioè nella parte dove è più facile indurne la dissipazione e fare il controllo della temperatura. Prende il nome dal fatto che la rotazione deve essere sincrona rispetto alla frequenza della corrente alternata di alimentazione. Quindi questo tipo di motori gira a velocità pressoché costante e deve essere avviato in maniera assistita. Ad esempio in un motore trifase sul cui rotore vi siano 20 elementi magnetico/induttivi la velocità minima non nulla è quella che prevede un ventesimo di giro per 50 Hz cioè 150 giri/minuto. Le velocità possibili sono tutte multiple di quest'ultima. Però un motore con controllo elettronico opera correttamente anche a bassissima velocità. La velocità minima ottenibile è definita solamente dalla risoluzione del sensore di posizionamento utilizzato. In generale la velocità minima a cui la rotazione è perfettamente uniforme è quella in cui la frequenza dell'encoder supera la banda passante del sistema.

mag. permanenti una delle due bobine può essere sostituita da magneti permanenti ma il campo magnetico di un magnete permanente rispetto a quello generato da una bobina è molto più basso a parità di volume occupato dal dispositivo. Quindi per ottenere gli stessi effetti occorre che l'altra bobina sia in grado di generare un campo magnetico variabile di maggiore intensità e la distanza fra bobina e i magneti deve essere la minima possibile. Se i magneti permanenti sono fissati al rotore piuttosto che al telaio la necessità di avere delle spazzole di alimentazione cessa completamente.

PWM acronimo di Pulse Width Modulation, cioè modulazione a larghezza di impulso, significa che la potenza erogata dal motore è controllata dal duty cicle del segnale periodico di controllo. In questo caso il motore è dotato di un'apposita elettronica di controllo che modula la frequenza di alimentazione in maniera da ottenere una variazione continua della velocità, sia in accelerazione che in decelerazione. La frenata di emergenza in caso di mancanza di alimentazione si ottiene mettendo in cortocircuito le fasi dello statore in questo modo il campo indotto dai magneti in rotazione sulla bobina principale trasforma l'energia cinetica rotatoria in corrente circolante che si dissipa nella bobina trasformandosi infine in calore.

densità di coppia s'intende il rapporto fra la coppia generata e il peso del motore. Sebbene in generale la densità sia una misura riferibile al volume quando si parla di motori elettrici il peso è una misura più interessante anche perché a causa dell'omogeneità dei materiali impiegati il rapporto fra peso e volume, cioè la densità di massa, non è sensibilmente migliorabile[1].

[1]Almeno fino a quando non saranno scoperti e prodotti in modo industriale materiali che siano superconduttori anche a temperatura ambiente o relativamente vicina.

banda passante rappresenta l'intervallo compreso fra la minima e massima velocità di rotazione fuori dal quale la coppia cala drasticamente. Si tratta di una quantità che mette in relazione l'accelerazione della rotazione in funzione dell'effetto della variazione della coppia motrice. Un'ampia banda passante permette una movimentazione molto rapida e di effettuare un controllo a contro-reazione della velocità e della posizione. Imporre accelerazioni oltre la banda passante può portare a maggiori ed inutili errori di inseguimento nei transitori e/o a eccitare frequenze di risonanza che innescano vibrazioni meccaniche.

IP65 standard che si riferisce al grado di protezione. In questo caso s'intende che il motore è protetto da infiltrazioni di polvere o in generale particelle solide e da getti accidentali d'acqua o di altri liquidi non corrosivi.

RMS acronimo di Root Mean Square indica il valore efficace di una grandezza elettrica in corrente alternata. E' un parametro che ha lo scopo di semplificare i calcoli ingegneristici evitando di analizzare ogni istante infinitesimo che costituisce la forma d'onda. Il valore efficace di una grandezza si può definire come quel valore che in regime continuo svilupperebbe la stessa potenza.

La velocità nominale del tipo di motore impiegato è di 6000 giri/minuto, i poli magnetici sul rotore sono 20, l'alimentazione è a corrente trifase a 380 V, l'assorbimento nominale di corrente è $3\,A_{rms}$ che corrisponde a una potenza nominale assorbita di $1.14\,KW_{rms}$. La coppia nominale è $3.44 Nm_{rms}$ che moltiplicata per la velocità nominale fornisce $1.08 KW_{rms}$ di potenza meccanica all'albero quindi con un rendimento massimo teorico, calcolato come rapporto fra potenza assorbita e quella erogata, del 94.7%. La coppia di picco è $6.12\,Nm$ da cui si ricava che la potenza di picco assorbita alla velocità nominale è di $6.12\,Nm \cdot 314\,\frac{rad}{s} \cdot \frac{1}{0.947} = 2.03\,KW$.

1.2 Specifiche degli encoder

In generale per quel che riguarda la movimentazione la precisione dell'encoder è di gran lunga più importante di quanto non lo sia la qualità costruttiva dei motori.

Tratto dal materiale informativo del produttore:

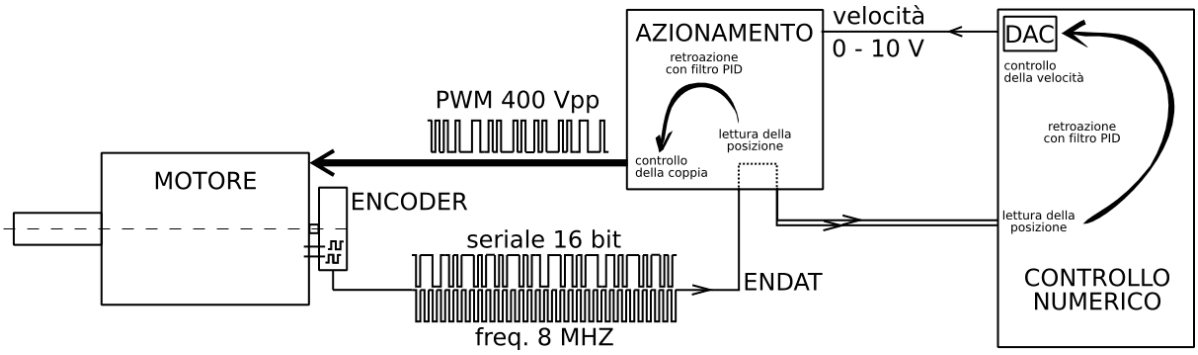
> The EnDat 2.2 interface from HEIDENHAIN is a digital, bidirectional interface for encoders. It is capable both of transmitting position values from incremental and absolute encoders as well as transmitting or updating information stored in the encoder, or saving new information. Thanks to the serial transmission method, only four signal lines are required. The data are transmitted in synchronism with the clock signal from the subsequent electronics. The type of transmission (position values, parameters, diagnostics, etc.) is selected by mode commands that the subsequent electronics send to the encoder.

L'alimentazione è 5 Vcc e la frequenza di lavoro arriva fino a 8 MHz. Entrambe devono essere fornite dall'esterno. Alla velocità nominale di 100 giri al secondo per 2^{16} conteggi a giro ci si aspetta di avere un segnale con banda di 6.5 MHz. Si noti però che la banda di comunicazione raramente supera gli $1 \div 2$ MHz quindi nei sistemi in cui i motori raggiungono alte velocità si limita il numero di conteggi per giro perdendo di risoluzione ma rimanendo nella banda del canale trasmissivo.

1.3 Requisiti delle tempistiche del controllo

Nella figura sottostante viene mostrato lo schema concettuale di un controllo motore basato su doppio anello di controllo: il primo è detto anello di velocità e il secondo anello di posizione. Il primo anello di retroazione, che coinvolge il motore e l'azionamento, permette di regolare la velocità del motore in funzione dell'ingresso analogico $0 \div 10\,V$. Il secondo anello di retroazione, che coinvolge l'azionamento e il controllo numerico, permette di controllare la posizione del motore fornendo all'azionamento un profilo di velocità.

In questo contesto è utile notare che il primo di questi anelli, quello di velocità, è chiuso in un tempo molto più breve del secondo e che l'uso di due anelli di retroazione, seppure sia la soluzione più diffusa, non è quella ottimale. Infatti se calibrare un filtro PID[2] è relativamente facile non lo è altrettanto calibrarne con precisione due in cascata.



L'encoder fornisce all'azionamento le informazioni sulla posizione del rotore. Nello schema a due anelli di retroazione quest'informazione è utilizzata per derivare la velocità e controbilanciare adeguatamente il segnale PWM.

E' possibile sostituire l'anello di velocità con uno di corrente nel quale la posizione fornita dall'encoder serve all'azionamento per determinare le fasi di alimentazione del motore sincrono in maniera da fornire una coppia costante.

In quest'ultimo caso il filtro PID sull'azionamento non è più necessario e l'unico vero anello di retroazione è costituito dall'encoder e dal controllo numerico mentre l'azionamento funge semplicemente da attuatore. Ovviamente il sistema a singolo anello PID deve essere chiuso in un tempo almeno uguale al più veloce dei due anelli nello schema precedente da questo ne consegue che il ciclo di retroazione globale debba chiudersi in $\frac{1}{8\,MHz} = 125\,uS$.

Per l'abitudine di preferire i numeri interi che sono potenze intere della base binaria è stato scelto come requisito temporale 128 uS di ciclo.

[2]Il filtro PID è composto da tre termini dai quali prende il nome per acronimo: proporzionale, integrativo e derivativo. La sua larga diffusione è dovuta alla sua capacità di adattarsi a diverse tipologie di controllo e al fatto che ognuno dei tre termini di cui composto è facilmente relazionabile a una qualche caratteristica del sistema o della funzione di trasferimento che si vuole trattare.

$$v(x) \propto a\varepsilon \cdot x + b\int \varepsilon dt \cdot x + c\frac{d\varepsilon}{dt} \cdot x \quad \text{dove} \quad \varepsilon = v_{desiderata} - v_{attuale}$$

Il termine proporzionale è correlato con la rigidità del sistema. Quello integrale serve per compensare gli attriti meccanici e per migliorare l'accuratezza a velocità prossime a zero. Infine il termine derivativo è utile per smorzare fenomeni di over-shooting e quindi aumentare la stabilità del sistema di controllo.

1.4 Uso del bus Ethernet: verifica dell'adeguatezza

L'Ethernet è in assoluto uno dei bus più conosciuti e utilizzati al mondo e il suo impiego sta crescendo nel segmento della domotica e anche del controllo industriale. Le schede di comunicazione Ethernet sono talmente diffuse che si ritiene sia assolutamente superfluo dilungarsi oltre sulle motivazioni di questa scelta.

Si consideri il bus Ethernet[3] come specificato dallo standard IEEE 802.3[4], in particolare con lunghezza minima del pacchetto di 64 byte allora:

- un sistema di comunicazione che abbia una banda di 10 Mbit/s per trasportare 64 byte, cioè 512 bit, di informazione impiega 51.2 uS;
- il mezzo trasmissivo è una risorsa concorrenziale sulla quale occorre evitare collisioni per cui l'*interframe gap*[5] minimo è di 96 bits, cioè 9.6 uS;
- un segnale che viaggi su un cavo per una lunghezza di 300 metri alla velocità di $300 \cdot 10^6$ metri al secondo impiega 1 uS per giungere a destinazione.

Ne consegue che un ricevitore in ascolto su un bus Ethernet attende almeno 62 uS per vedersi recapitato e leggere un pacchetto completo. Nell'ipotesi di controllare almeno due motori attraverso un solo PC il ciclo di controllo non può essere inferiore a 186 uS: trasmissione del master, risposta del primo e del secondo motore. Questo senza contare eventuali ritardi dovuti all'elaborazione del dato principalmente sulle schede di controllo motori[6].

Rinunciando a usare il protocollo TCP[7] e anche quello IP[8] si potrebbe utilizzare un pacchetto minimo di 22 byte composto da: 4 byte di dati per ogni singolo motore per almeno due motori più 14 byte per il MAC header dovuto al protocollo fisico di trasmissione. Nell'ipotesi di violare lo standard IEEE 802.3, riguardo alle misure anticollisione, si potrebbe chiudere il ciclo di controllo in un minimo teorico di 56 uS senza però avere alcun margine di errore sulla temporizzazione, pena una disastrosa collisione. Inoltre potrebbe non essere facile trovare delle schede di rete Ethernet che permettano, interfacciandosi con il firmware, di violare lo standard. Infine non vi sarebbe alcuna garanzia che tali schede sarebbero ugualmente reperibili in futuro.

Anche supponendo di poter accettare queste condizioni e di chiudere il ciclo di controllo in 128 uS il numero di motori che risulterebbero controllabili dal PC-master sarebbero teoricamente 4 al massimo. Oltre questo numero il software di controllo real-time dovrebbe essere scritto in maniera da gestire contemporaneamente più schede di rete.

[3] Ethernet - `http://en.wikipedia.org/wiki/Ethernet`

[4] IEEE 802.3 - `http://[en|it].wikipedia.org/wiki/IEEE_802.3`

[5] interframe gap - `http://en.wikipedia.org/wiki/Interframe_gap`

[6] Questi ritardi in realtà sono trascurabili, non tanto confidando su un processore particolarmente veloce quindi costoso e perciò difficilmente proponibile per la scheda di controllo, ma perché il firmware può essere scritto in maniera da trasmettere una risposta calcolata sui dati precedentemente elaborati. Infatti dividendo il tempo di ciclo in tre parti uguali si possono distribuire i compiti in modo da chiudere il ciclo senza introdurre ulteriori ritardi:

1. lettura encoder, aggiornamento attuatore, elaborazione dati e preparazione pacchetto;
2. ricezione del pacchetto dal PC master contenente i dati per aggiornare l'attuatore;
3. spedizione del pacchetto precedentemente preparato e ritorno al punto primo.

[7] TCP - `http://en.wikipedia.org/wiki/Transmission_Control_Protocol`

[8] IPv4 - `http://en.wikipedia.org/wiki/IPv4`

1.4.1 Necessità d'impiegare il fast Ethernet e relative implicazioni

Si potrebbe affermare che nemmeno vi fosse la necessità di considerare l'Ethernet a 10 Mbit/s quando ormai in commercio si trovano diversi modelli di schede di rete a 1000 Mbit/s a prezzo contenuto. Ciò che può apparire una piccola differenza di prezzo per un PC può diventare, nella prospettiva di adottare una tecnologia piuttosto che un altra per una produzione industriale, un costo tale da ridurre in modo inaccettabile la concorrenzialità della soluzione[9]. In particolare questo è vero per la scheda di controllo da integrare sui motori per la quale la necessità di avere a bordo un Ethernet a 100 Mbit/s non solo aumenta il costo del chipset specifico ma implica un salto di qualità verso schede ben più evolute, anche rispetto altri componenti quali CPU e RAM, perciò decisamente più costose.

Per le ragioni esposte nella sezione precedente la scelta di un bus Ethernet a 10 Mbit/s pare del tutto inappropriata per questo progetto. Detto questo occorre valutare se l'uso di controllori dotati di fast Ethernet a 100 Mbit/s può essere considerato soddisfacente oppure no. Aumentare la banda di un ordine di grandezza significa ridurre di altrettanto le tempistiche di comunicazione. Questa considerazione da sola è sufficiente per accettare il fast Ethernet, cioè lo standard IEEE 802.3u, come il minimo necessario.

Nonostante questa scelta nel proseguo di questo documento si trovano misure fatte anche su Ethernet 10 Mbit/s perché le quantità misurate, quale ad esempio lo jitter massimo, sono relative al software utilizzato e non alla banda disponibile o al periodo di ciclo. Anzi tanto più un microcontrollore è minimale tanto più il suo comportamento è prevedibile e quindi, come sarà più avanti spiegato, tanto meno interferisce con il processo di misura.

1.4.2 Sovradimensionamento: opportunità o problema?

Premesso che il tempo di sviluppo del software e quindi il suo costo dipende anche dalle possibilità dell'hardware per sovradimensionamento si intende l'uso di un hardware di caratteristiche superiori a quelle che sarebbero necessarie per implementare in modo funzionale e ragionevolmente conveniente la soluzione prevista.

Alessandro Rubini sostiene che l'uso di tecnologie sovradimensionate sia per definizione un errore. Io più ingenuamente mi affido al detto latino "*melius abundare quam deficere*", ovviamente a parità, o quasi, di costo. Al di là di mere disquisizioni filosofiche sostengo questa posizione perché sono convinto che una tecnologia sovradimensionata oggi è, probabilmente, quella che domani sarà percepita come comune e quindi anche più facilmente reperibile e mantenibile.

In effetti questa parrebbe essere un diatriba su una definizione piuttosto che su fatti rilevanti: dato un progetto il minimo hardware è qualcosa di relativamente univoco, a parità di software, mentre la distinzione fra ottimale e sovradimensionato è piuttosto opinabile. Se però nel contesto si inseriscono anche elementi esterni alla soluzione da implementare si può dire che il primo punto di vista sia:

- si sceglie un hardware che sia sufficientemente adatto alle esigenze

mentre il secondo è come il primo ma si estende così:

- scelto un hardware si valuta il suo costo e a parità si sceglie l'hardware più prestante.

[9] Il chip ENC28J60 offre una Ethernet a basso costo per microcontrollori su bus SPI, un nota significativa senza la quale, il lettore informato penserebbe che non è vero che il 10 Mbit/s costa meno del 100 Mbit/s perché periferiche solo a 10 Mbit/s non se ne vendono nemmeno più in campo PC che è poi quello che tutti conoscono.

Il secondo punto di vista premia il produttore hardware più concorrenziale, che strettamente nell'ipotesi di libero mercato, è quello che ragionevolmente ha investito di più nella ricerca e nello sviluppo. Affidarsi alla soluzione hardware di quest'ultimo significa affidarsi a quel produttore che avrà maggiore probabilità di vendere quindi di ridurre i costi per unità di prodotto e per il cui prodotto sarà maggiore la reperibilità e la facilità di reclutare personale con esperienza specifica.

In entrambi i punti di vista però è stato implicitamente considerato di evitare assolutamente situazioni di lock-in affidandosi solo a quelle soluzione hardware di produttori che rendano pubbliche tutte le necessarie specifiche tecniche. L'esperienza e la letteratura insegnano che il verificarsi di situazioni di lock-in tecnologico sono frutto di una iniziale scelta che verosimilmente appare come la più conveniente inizialmente per poi dimostrarsi già sul medio periodo molto più costosa delle alternative aperte.

Certamente vi è però un pericolo insito nell'uso di tecnologia sovradimensionata. Ad esempio ritornando al caso concreto di questo progetto si verifica che se il sistema funzionasse a 10 Mbit/s allora per certo funzionerebbe anche a 100 Mbit/s mentre il viceversa non è vero. A parte questo è sempre buona norma verificare che le previsioni teoriche, quale l'impossibilità dell'uso dell'Ethernet a 10 Mbit/s, siano compatibili con l'esperienza di laboratorio.

Inoltre cercare di far funzionare il sistema in condizioni limite permette di confrontarsi con situazioni che potrebbero non essere altrimenti prevedibili e quindi, non essendo state considerate in fase di progetto, nel migliore dei casi farebbero la loro comparsa nelle fasi avanzate di verifica oppure, nel peggiore dei casi, una volta messe in campo.

A questo punto però è assolutamente doveroso inserire la risposta avuta per e-mail da Alessandro circa questa sottosezione perché ciò che viene detto è sempre inderogabilmente legato al contesto e al cambiare di questo può cambiare radicalmente la sua opportunità: "*Sul discorso del sovradimensionato hai ragione sicuramente tu se la poni così. Tieni conto, però, che il mondo industriale si muove più lentamente del mondo PC. Se puoi mettere un ENC28 attaccato ad un microcontrollore SPI*[10] *oggi, non è detto che tu possa mettere un 100 Mbit/s sullo stesso microcontrollore, o su uno di costo equivalente, tra due anni. In realtà un altra cosa da tenere in conto sul sovradimensionato è la difficoltà di realizzazione. Una schedina con integrati DIP*[11] *si fa in casa con pochi pezzi ma se ci si sposta su alternative più potenti, anche a parità di costo, si inizia a dover fare la produzione fuori perché si parla subito di SMD*[12] *con passo di 0.8 mm e facilmente, oggi, si deve andare su BGA*[13] *che ha costi di produzione molto alti sui piccoli numeri. Quindi è verissimo il tuo discorso sui grandi numeri ma la situazione è ben diversa su produzioni più limitate.*"

[10]Serial Peripheral Interface - `http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus`
[11]Dual Inline Package - `http://en.wikipedia.org/wiki/Dual_In-line_Package`
[12]Surface Mounting Device - `http://en.wikipedia.org/wiki/Surface_mount_technology`
[13]Ball Grid Array - `http://en.wikipedia.org/wiki/Ball_grid_array`

2 Descrizione del sistema

Il sistema utilizzato in questo studio si compone di diverse parti:

- un PC-master che controlla gli attuatori e riceve informazioni dai sensori comunicando con le schede sui motori;
- per ogni motore una scheda, dotata di interfaccia Ethernet, capace di istruire gli attuatori e leggere i sensori relativi al motore da pilotare;
- un notebook per lo sniffing del traffico e che, in alcuni esperimenti, raccoglie le misurazioni;
- diversi switch/hub a cui demandare l'instradamento del traffico fra PC-master e attuatori;

2.1 Scelta chipset Realtek 8139

Si è scelto di usare schede di rete che montino il chipset Realtek 8139 perché

- hanno un costo esiguo,
- sono facilmente reperibili sul mercato
- e sono ampiamente supportate da Linux.

La banda disponibile su bus Ethernet varia da 10 a 1000 Mbit/s ma da questo punto di vista il limite è imposto dall'interfaccia Ethernet impiegata sulle schede motori e soprattutto dal dover contenere il loro costo di produzione.

2.2 Lista dell'hardware utilizzato

Riguardo ai personal computer basati su architettura Intel x86 la descrizione è volutamente limitata ad alcuni componenti in quanto gli altri sono indifferenti per quanto concerne questo progetto. Si sono utilizzati:

- PC x86 compatibile assemblato
  - Intel Celeron a 1200 MHz
  - scheda di rete PCI con chipset Realtek 8139 a 100 Mbit/s
  - un'altra scheda di rete aggiuntiva per la supervisione remota
- PC x86 compatibile laptop
  - Intel Pentium M core Banias a 1600 MHz
  - scheda di rete PCI integrata con chipset Realtek 8139 a 100 Mbit/s
  - un'altra scheda di rete PCMCIA aggiuntiva per la supervisione remota

Per il controllo motori si sono sperimentate due diverse soluzioni:

- Zefeer board [2] basata su processore ARM EP9315 equipaggiata con Ethernet a 100 Mbit/s
- Atmel AVR ATMEGA-88 [3] con Ethernet a 10 Mbit/s

E' stato valutato anche l'uso della board FOX distribuita da Acme Systems [4] e dotata di processore CRIS e Ethernet a 100 Mbit/s. Purtroppo per questa architettura non è disponibile nessun supporto real-time per Linux ma è stata impiegata comunque per avere un ulteriore possibilità di confronto, in particolare per quel che riguarda il ramo 2.4 del kernel.

2.3 Scelta kernel Linux 2.4 vs 2.6

La valutazione di quale ramo del kernel Linux [5] (2.4 vs 2.6) sarebbe stato meglio utilizzare dal punto di vista del real-time è stata fatta con una ricerca di informazioni sulla rete che è sintetizzata nella raccolta dei documenti qui sotto elencati:

- Realtime Audio vs. Linux 2.6 di Lee Revell, Philadelphia (USA) del 2006 [6]
- Realtime benchmarks 2.4 vs 2.6 di Peter Laurich, Ottawa (Canada) del dicembre 2004 [7]
- Xenomai 2.4 vs 2.6 in embedded space, a thread in Xenomai m-list [8]
- Comparing Linux 2.4 and Linux 2.6 Kernels di Wolfgang Denk dell'aprile 2005 [9]

Da questi documenti, e da altre informazioni di minore impatto sparse sulla rete, si può concludere che il kernel Linux ramo 2.4 è più piccolo, veloce e più maturo dal punto di vista della stabilità del codice sorgente[14] del ramo 2.6. In particolare dal documento che tratta del benchmark per il real-time audio citato risulta che il 2.4 ha latenze inferiori nonostante le notevoli migliorie introdotte nello scheduler del 2.6 almeno fino alla versione 2.6.14.

Questo responso non deve sorprendere in quanto sebbene l'architettura dello scheduler del ramo 2.6 sia molto migliore dal punto di vista della scalabilità la diminuzione delle latenze massime dipende in gran parte dalla dimensioni delle sezioni critiche non interrompibili nelle altre porzioni del kernel. Nel caso di sistemi dedicati il numero di processi è relativamente piccolo e costante inoltre la scelta di hardware particolare permette di escludere quelle parti di codice sorgente che contengono sezioni critiche particolarmente impegnative per la CPU.

In ogni caso è importante notare che nel contesto real-time è comunque necessario l'uso di uno scheduler, come quello fornito da RTAI o Xenomai, che si occupi della gestione dei task real-time mentre i processi meno prioritari sono lasciati allo scheduler del kernel il quale quindi ottiene la CPU solamente quando i processi real-time sono stati ragionevolmente serviti.

Nonostante sia emerso che il ramo 2.4 fosse migliore rispetto alle latenze introdotte è stato scelto il ramo 2.6 perché quest'ultimo è parso sufficientemente adeguato agli scopi e rispetto al 2.4 offre un maggiore supporto per l'hardware attuale e futuro. Infine perché un buon supporto real-time, quale quello offerto da RTAI, si ritiene debba essere ragionevolmente indipendente dal sistema operativo su cui si appoggia.

La possibilità di avere un maggiore supporto hardware è stato considerato come un fattore predominante nella scelta del ramo 2.6 per lo studio e sviluppo di questo progetto in quanto,

[14] Il ramo 2.4 è soggetto solo a correzioni d'errore e eventualmente al back-porting di codice scritto per il ramo 2.6 relativo a funzionalità che si ritengono troppo importanti per essere tralasciate ma che nel contempo non richiedono un cambio dell'architettura tale da creare una discontinuità nell'impiego rispetto alle precedenti versioni 2.4.x.

seppur vero che se ne prevede l'utilizzo in campo industriale dove le personalizzazioni sono anche molto forti, è altrettanto vero che fra gli obbiettivi prioritari di questo studio vi sia quello di utilizzare hardware di massa di cui ovviamente ci si aspetta di avere il supporto più vasto possibile. Infine poiché non è previsto a breve l'apertura di un ramo 2.8 si suppone che il 2.6 dominerà la scena per ancora svariati anni e quindi il supporto al nuovo hardware verrà aggiunto proprio in questo ramo.

2.4 Scelta del supporto real-time

In questo progetto il supporto real-time deve controllare due sottosistemi ognuno dei quali è basato su un componente hardware: la comunicazione di rete Ethernet attraverso il driver della relativa scheda e la gestione del tempo di CPU da dividersi adeguatamente fra i compiti da eseguirsi in tempo reale e il resto delle incombenze di sistema.

2.4.1 Real-time scheduler

Riguardo al supporto real-time per GNU/Linux i due progetti liberi e più maturi che godono di uno sviluppo in comunità sono RTAI [10] e Xenomai [11]. Al momento in cui questo studio prendeva vita le differenze fra questi due software erano forse al loro minimo storico avendo condiviso, per un precedente periodo, i principali sviluppatori.

Vale la pena citare la relativa FAQ sul sito di Xenomai [12]:

> There is actually a long list of differences, though both projects share a few ideas and support the RTDM layer. The major differences derive from the goals the projects aim for, and from their respective implementation. While RTAI is focused on lowest technically feasible latencies, Xenomai also considers clean extensibility (RTOS skins), portability, and maintainability as very important goals. Xenomai's path towards PREEMPT_RT support is another major difference compared to RTAI's objectives.

E' possibile trovare anche una comparazione più dettagliata [13] sulla lista di sviluppo di Xenomai.

Per gli obbiettivi di questo progetto la portabilità verso altri sistemi è stata valutata di gran lunga meno importante rispetto alla riduzione delle latenze e al contenimento del loro jitter. La scelta naturale cadde quindi su RTAI il quale per altro è un progetto sviluppato all'interno di un'università italiana: il Politecnico di Milano.

2.4.2 Real-time network

Riguardo al real-time network era impossibile non prendere in considerazione RTnet sotto un duplice punto di vista:

- utilizzo diretto eventualmente adattandolo con opportune modifiche

oppure

- utilizzo indiretto come fonte di codice sorgente da cui attingere

per l'implementazione di un sistema dedicato poichè un driver real-time non è un driver convenzionale solamente più veloce ma è anche architetturalmente diverso. Si è posta quindi una scelta fra due possibilità che non aveva la sua principale ragione solamente nel grado di adeguatezza che RTnet avrebbe dimostrato ma anche dalla filosofia dell'approccio da seguire:

- adattare un qualche software esistente al progetto

oppure

- scrivere il software a partire dall'hardware utilizzato

Nel primo caso si sarebbe partiti da un pacchetto che funziona e per passi successivi lo si sarebbe modificato fintanto che non si ottenesse il risultato atteso. Ovviamente il rischio sotteso era che il processo di adattamento si fosse scontrato con l'architettura del software di partenza in modo tale da rendere questa strada molto più difficile dell'alternativa.

Nel secondo caso non si sarebbe ereditata nessuna architettura che avrebbe potuto ostacolare e/o condizionare pesantemente il processo di sviluppo ma si sarebbe cominciato, se non proprio da zero, comunque da abbastanza poco. In questo secondo caso i primi risultati sarebbero stati visibili solo a uno stadio più avanzato dello sviluppo e il rischio sarebbe stato quello di perdere molto tempo nella fase iniziale.

A posteriori si può dire che si è sempre partiti dall'esistente infatti per quel che mi riguarda personalmente in campo PC ho sviluppato il codice necessario dentro la cornice NOMAC di RTnet. Per quanto riguarda Alessandro Rubini, essendosi occupato del software dei vari prototipi delle schede lato motore, in un caso ha sviluppato a partire dal codice di U-BOOT per la Zefeer mentre per la scheda con l'AVR dal codice dell'ENC28 e infine il tentativo di adattare all'architettura PC il driver senza interruzioni per la Realtek 8139 presente in U-BOOT è stato abbandonato per questioni di tempo.

2.5 Lista del software impiegato

Per il PC-master è stato impiegato:

- kernel vanilla 2.6.16.37
- rtai-3.4
- rtnet-0.9.7

Ho scelto di usare l'ultima versione della serie 2.6.16.x anche se RTAI supporta anche la serie 2.6.17.x perché al momento della scelta quest'ultima era ancora in sviluppo attivo.

Per il notebook è stato impiegato:

- kernel 2.6.12-19mdk + patch
- rtai-3.4 + patch
- rtnet-0.9.7

Poiché l'ambiente di lavoro del portatile era già configurato sia dal punto di vista dell'utente sia per quanto riguarda il supporto hardware si è ritenuto più veloce utilizzare un kernel quanto più compatibile con il sistema esistente e intervenire con delle modifiche al codice sorgente in maniera da poter installare RTAI.

Per la board ATMEL è stato sviluppato un firmware ad hoc mentre per la Zefeer board è stato impiegato U-BOOT "the universal boot loader" [14] e uno sviluppo software ad hoc.

L'uso di un sistema senza IRQ è stato un requisito indispensabile per offrire al PC-master un interlocutore che avesse uno jitter nei tempi di risposta pressoché nullo. Sarebbe infatti

stato impossibile fare alcune misure precise di jitter sul PC-master se il risponditore avesse aggiunto un contributo non trascurabile[15].

La FOX board è stata utilizzata con l'immagine fornita dal produttore che al momento dell'acquisto, precedente a questo studio, aveva ancora in uso il kernel Linux 2.4.31.

2.6 Configurazioni

2.6.1 Linux

Il kernel Linux è stato compilato con le seguenti opzioni:

- senza il supporto Realtek 8139 o con supporto modulare
- con il supporto TSC che richiede processori Pentium o superiori
- con il minimo dei driver necessari possibilmente integrati monoliticamente
- con il supporto per i moduli esterni necessario a RTAI e a RTnet

2.6.2 RTAI

RTAI è stato compilato con le seguenti opzioni:

- senza il supporto NET RPC
- senza il supporto per C++ nel kernel
- con il Real-Time Driver Model over RTAI
- con il supporto per applicazione RTAI 2.x
- per installarsi in `/usr/realtime`

2.6.3 RTnet

RTnet è stato compilato con le seguenti opzioni:

- con supporto a RTAI 3.3-cv o superiore e Xenomai 2.0.x
- con i driver Realtek 8139 e loop-back
- per installarsi in `/usr/realtime/rtnet`
- con una piccola modifica in `stack/include/rtnet.h` alla riga 39:

```
+ #if 0
= typedef uint64_t nanosecs_abs_t;
= typedef int64_t nanosecs_rel_t;
+ #endif
```

Per l'installazione e la configurazione di RTAI si è fatto riferimento al relativo *User Manual* [15] disponibile nella sezione documentazione del sito del progetto stesso.

[15]Una prima versione del prototipo basato sulla Zefeer aveva uno jitter non trascurabile ma è stato comunque utilizzato per fare il confronto fra misure prese in tutti i casi possibili, anche quelli non ottimali.

2.6.4 Laptop

Il laptop è stato configurato in maniera da escludere il supporto APIC (Advanced Programmable Interrupt Controller), quello PM (Power Management) e quindi anche quello ACPI (Advanced Configuration and Power Interface). Riguardo all'APIC perché questo portatile non pare disporne e comunque il manuale di RTAI consiglia la sua esclusione sulle macchine uniprocessore. Riguardo alla gestione del consumo energetico l'esclusione è caldamente consigliata perché sorgente di ritardi imprevedibili in particolare a causa della gestione degli stati di sleep della CPU e alla gestione dinamica della frequenza di lavoro che andrebbe a incidere sull'intepretazione delle letture del TSC (Time Step Counter) il cui supporto verrà trattato nel proseguo di questo documento.

Infine il codice sorgente di RTAI ha dovuto essere modificato come segue:

- in base/include/asm/rtai_atomic.h alla riga 28:

```
- #if LINUX_VERSION_CODE < KERNEL_VERSION(2,6,0)
+ #if LINUX_VERSION_CODE < KERNEL_VERSION(2,6,13)
```

- in addons/rtdm/xn.h alla riga 167:

```
= #elif LINUX_VERSION_CODE >= KERNEL_VERSION(2,6,10)
+ #define __va_to_kva(va) (va)
```

affinché potesse essere compilato senza errori e quindi funzionare.

Su questo laptop il supporto real-time si è dimostrato totalmente incompatibile con l'uso del sottosistema USB quindi si è proceduto a riconfigurare la partenza del sistema in modalità *single user* avendo cura di caricare solo i moduli kernel strettamente necessari da i quali ovviamente sono stati esclusi quelli relativi al sottosistema incriminato e alle relative periferiche. Questo limite non deve stupire in quanto il sottosistema USB è indicato nella documentazione di RTAI come uno di quelli critici per il buon funzionamento del supporto real-time anche se in questo caso il riavvio sistematico del laptop al primo movimento del mouse USB ha quanto mai semplificato l'eventuale decisione di mantenerlo attivo o escluderlo!

3 Misure e valutazioni preliminari

Questo capitolo riguarda la prima parte di misure che è stata fatta per determinare l'utilità di RTnet all'interno del progetto e per impratichirsi con questo software e con la tipologia di misure da effettuare.

3.1 Raccolta delle misure per campioni

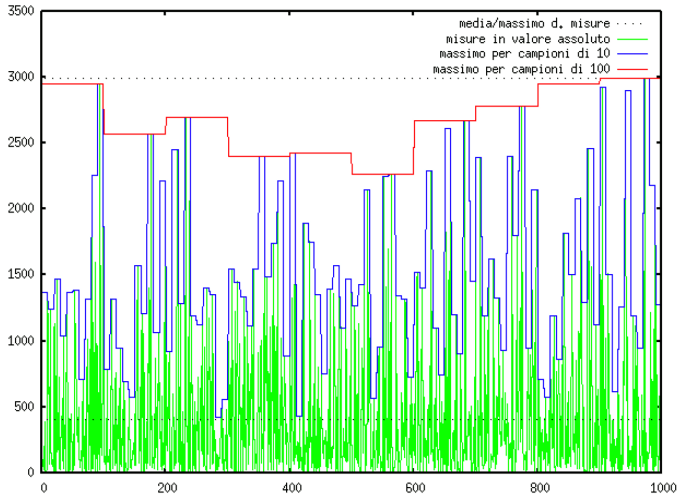
Vi sono due fondamentali maniere di raccogliere misure sperimentali:

- ogni singola misura rappresenta un dato quindi non vi è nessun accorpamento;
- le misure vengono accorpate in campioni e solo un valore calcolato sull'insieme è il dato.

I dati così raccolti sono assoggettati a un'elaborazione e a un'analisi che, nel secondo modo di raccogliere le misure, risulta significativa solamente se il metodo di accorpamento mantiene una relazione di significatività fra dati e misure cioè è compatibile con le proprietà e la natura del fenomeno osservato o della parte interessante di esso.

3.1.1 Dimensione del campione

La dimensione del campione in questo studio è tutto sommato poco rilevante però vale la pena di mostrare, almeno a livello intuitivo, come questo tipo di campionatura potrebbe interferire con l'analisi delle misure. Il seguente grafico:



è stato disegnato da una serie di dati generati da questa funzione:

$$m_i = 100 \cdot |\sin(x_i) * \cos(x_i)| * x_i^2 \text{ dove } x_i \in [0,\, 2\pi] \subset \Re \text{ e } i \in [0,\, 1000] \subset \aleph$$

calcolata sul codominio di x una variabile casuale distribuita uniformemente in un intervallo finito. L'impressione che trasferiscono i tre grafici è molto diversa fra loro: mentre il grafico delle misure è abbastanza caotico, quello fatto su grandi campioni appare statico, quello fatto su campioni di misura appropriata mostra un andamento dinamico ma leggibile.

Nel proseguo di questo documento verranno specificate le modalità operative realmente utilizzate nella raccolta delle misure.

3.2 Latenza e jitter

Sia $T_{ax}(\bullet)\epsilon\,[T_{ax}^{min},\, T_{ax}^{max}] \subset \Re$ una funzione tempo assoluto, cioè monotona crescente, considerata in un codominio limitato. Si consideri una sequenza di eventi e per ognuno di essi la relativa misura di tempo T_{ax} si definisce campione l'insieme di queste misure. Sia presa una misura e la successiva indicate con n e $n+1$ appartenenti a uno stesso campione composto da G misure e si consideri la loro differenza in valore assoluto:

$$|T_{ax}(n+1) - T_{ax}(n)| = \Delta T_{ax}(n)$$

allora per ogni n si può scrivere:

$$\Delta T_{ax}(n) = \Delta T_{ax}^{min} + \delta T_{ax}(n) \text{ dove } \delta T_{ax}\epsilon \left[0,\, \Delta T_{ax}^{max} - \Delta T_{ax}^{min}\right] \subset \Re \text{ e } n \in [1,\, ...,\, G] \subset \aleph.$$

Si definiscono le quantità ΔT_{ax}^{min}, ΔT_{ax}^{max} e δT_{ax}rispettivamente come la **latenza minima**, la **latenza massima** e lo **jitter** dell'evento arbitrario ma periodico indicato con il pedice ax. Inoltre si può definire $\Delta T_{ax}^{avg} = \sum_{n=1}^{G-1} \frac{\Delta T_{ax}(n)}{G-1}$ la **latenza media** che risulta utile qualora la distribuzione dei ritardi sia esattamente simmetrica, o ragionevolmente approssimabile a una distribuzione simmetrica rispetto alla media, per cui si può scrivere:

$$\Delta T_{ax}(n) = \Delta T_{ax}^{avg} \pm \varepsilon T_{ax}(n) \text{ dove } \varepsilon T_{ax}\epsilon \left[0,\, \tfrac{\Delta T_{ax}^{max} - \Delta T_{ax}^{min}}{2}\right] \subset \Re \text{ e } n \in [1,\, ...,\, G] \subset \aleph.$$

In seguito si vedrà che non sempre questo vincolo di simmetria è soddisfatto (cfr. sez. 5.4) ma soprattutto verrà detto perché la latenza media, e la relativa notazione per esprimere ΔT_{ax}, non sono di interesse per questo studio (cfr. sez. 5.3.3).

Sempre in seguito si vedrà che é possibile parlare di latenza e di jitter non solo per eventi periodici ma anche per una coppia di eventi qualsiasi purché siano rigorosamente legati fra loro da una relazione di causa-effetto cioè: all'accadere di uno segue necessariamente, entro un intervallo di tempo limitato, la manifestazione dell'altro.

3.2.1 Quantità interessanti da osservare

In breve si può anticipare che conoscere le latenze massime permette di stabilire qual'è il minimo periodo entro cui é possibile chiudere il ciclo di controllo mentre la conoscenza dello jitter massimo permette di stabilire con quanta precisione é possibile pilotare i motori, ovviamente a parità di hardware e software impiegato.

In questo studio si pone la necessità di osservare, come spiegato nella sezione 3.2.1, periodo minino e jitter massimo di un evento periodico o assimilabile. Calcolare il massimo e il minimo direttamente oppure sui campioni è equivalente:

$$\max(m_1,\, ...,\, m_N) = \max\left(\max(m_1,\, ...,\, m_G),\, \max(m_{G+1},\, ...,\, m_{2G}),\, ...,\, \max(m_{kG+1},\, ...,\, m_N)\right)$$

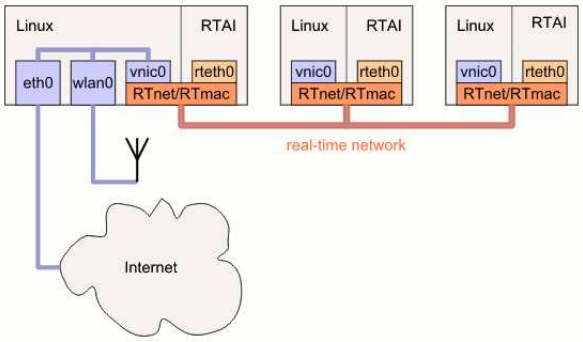
analogamente per il calcolo del minimo. Considerare solo valori non negativi permette di utilizzare variabili di tipo `unsigned long` per l'elaborazione delle statistiche e di mantenerne quanto più semplice possibile il codice senza incidere sulla significatività delle misure.

3.3 RTnet: architettura e valutazione

Il protocollo implementato da RTnet prevede una divisione del tempo in time-slot che se non altrimenti specificato vale 5 mS. All'interno di ogni time-slot vengono effettuate altre sottodivisioni che si possono chiamare canali e nei quali vengono smistate le comunicazioni di rete in tempo reale. Questo meccanismo generalmente noto come Time Division Multiplexing[16] è molto diffuso nell'ambito delle telecomunicazioni mentre in RTnet prende il nome di Time Division Multiple Access[17]; abbreviato con TDMA si occupa anche della sincronizzazione degli slave e prevede la possibilità di indicare master sostitutivi in caso di fallimento di quello in ruolo.

Tale struttura temporale è decisa da un master rispetto al quale tutti gli altri si comportano come slave. Questo significa, che indipendentemente da come viene cablata la rete fisica, una rete RTnet ha pregi e difetti di una topologia a stella con il master al centro. Ad ogni slave viene assegnato un canale e quindi il numero di slave deve essere determinato al momento di stabilire le specifiche del sistema.

Typical Network Setup



The figure above shows a typical configuration of an RTnet real-time network. To achieve deterministic behaviour, a dedicated network segment is required. Non-real-time traffic will be tunneled through this segment. One of the three real-time stations is configured as a gateway to arbitrary non-real-time networks.

RTnet implements UDP/IP, ICMP and ARP in a deterministic way. It provides a POSIX socket API to real-time user space processes and kernel modules. To avoid unpredictable collisions and congestions on Ethernet, an additional protocol layer called RTmac controls the media access. A dedicated Ethernet segment is required to guarantee bounded transmission delays, but RTnet also includes a mechanism to tunnel non real-time traffic like TCP/IP over RTmac, thus allowing a "single-cable" solution for connecting control systems. Some possible application domains are fieldbus replacements, distributed real-time computing, or video/audio streaming. - Fonte: `www.rtnet.org`

Alternativamente si può abilitare il supporto Real-Time Configuration Service[18], abbreviato in RTcfg, che permette di gestire dinamicamente il numero di slave purché esso non ecceda il massimo stabilito. Uno di questi canali è impiegato per la comunicazione TCP/IP non real-time la quale è offerta in user-space attraverso un device analogo a quello che fornirebbe un kernel driver per una qualsiasi scheda di rete reale.

Per maggiori informazioni sull'architettura e il funzionamento di RTnet si rimanda alla lettura del documento intitolato "RTnet - A Flexible Hard Real-Time Networking Framework" [16].

3.3.1 Prime misure per la valutazione di RTnet

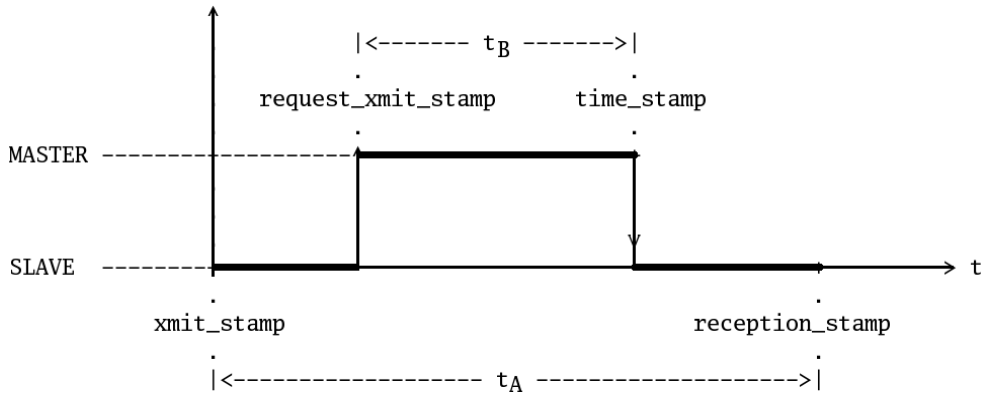
I test che seguono hanno il compito di verificare la durata minima del time-slot e di fornire una prima stima dello jitter massimo.

[16]TDM - `http://en.wikipedia.org/wiki/Time-division_multiplexing`
[17]TDMA specification - `http://www.rts.uni-hannover.de/rtnet/lxr/source/Documentation/TDMA.spec`
[18]RTcfg specification - `http://www.rts.uni-hannover.de/rtnet/lxr/source/Documentation/RTcfg.spec`

Quando il PC-slave si connette il kernel stampa nei log di sistema alcune righe che informano del minimo e del massimo ritardo rilevato, ad esempio:

```
RTAI[sched]:  timer setup = 2010 ns, resched latency = 2688 ns.
...
TDMA: calibrated master-to-slave packet delay:  17 us (min/max:  17/18 us)
```

Facendo una sequenza di connessioni si verifica la ripetibilità di queste misure e si registra il minimo dei minimi e il massimo dei massimi. Una stima cautelativa, cioè un maggiorante, dello jitter è la differenza fra queste due quantità. Analizzando il codice sorgente del file `rtnet/rtmac/tdma/tdma_proto.c` si deduce che la misura sopra presentata raccoglie dati in nella funzione `tdma_packet_rx()`, calcolata fra quattro istanti temporali:



t_A è il tempo dall'inizio della trasmissione alla fine della ricezione della risposta

t_B è il tempo necessario al server per acquisire il pacchetto e spedire la risposta

Supponendo che il mezzo di trasmissione comporti un ritardo uguale in entrambe le direzioni[19] allora $T_{volo} = \frac{t_A - t_B}{2}$ è il tempo di volo fra master e slave.

La tabella sottostante riporta la sequenza di verifica di una connessione cavo cross a 100 Mbit/s, con offset massimo consentito di 20 uS:

time-slot (uS)	si connette?
128	si
64	no
96	si
80	no
88	no
92	si
90	si

Il time-slot minimo risulta essere di 90 uS e su 10 connessioni fatte con tale impostazione si stima che lo jitter sia 19 max - 17 min = 2 uS

La tabella sottostante riporta la sequenza di verifica di una connessione mediante due cavi e un hub Netgear EN108 10 Mbit/S, con offset massimo consentito di 100 uS:

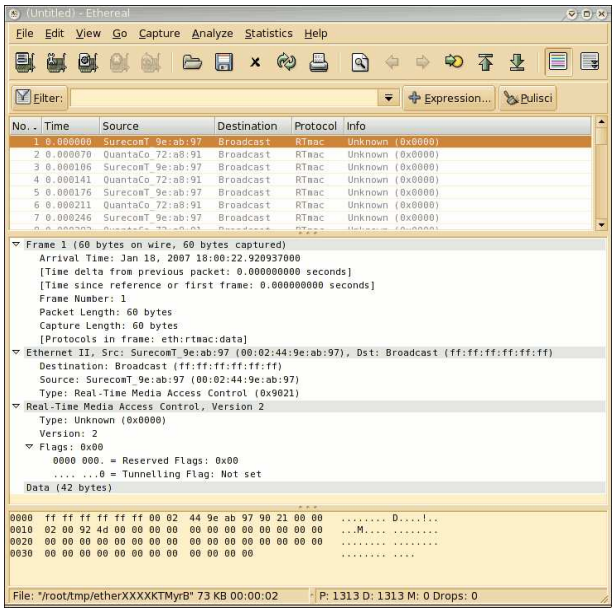
time-slot (uS)	si connette?
256	no
512	si
384	si
320	si
296	si
276	no
286	no

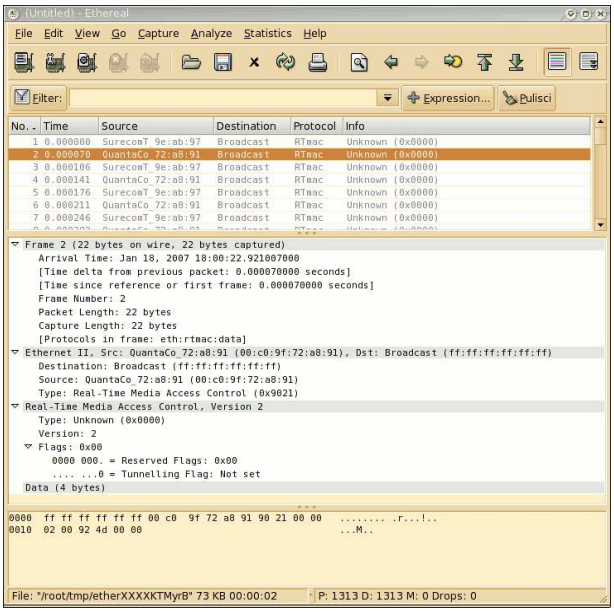
Il time-slot minimo risulta essere di 296 uS e su 10 connessioni fatte con tale impostazione si stima che lo jitter sia 79 max - 73 min = 6 uS

[19]Nel caso di un cavo di rete incrociato questa ipotesi è banalmente soddisfatta ma se la cablatura della rete avviene attraverso un dispositivo concentratore potrebbe non esserlo. Un hub o uno switch aggiunge sicuramente dei ritardi di trasmissione si suppone però che la differenza di ritardo fra le due direzioni sia trascurabile.

3.3.2 Analisi dei pacchetti RTnet via RTmac

Gli screenshot sotto mostrano il contenuto dei pacchetti RTnet transitati attraverso lo stack Real-Time Media Access Control[20]:





Quello di destra mostra il pacchetto catturato in locale da Ethereal[21] prima che fosse inviato verso il master remoto quindi ha una dimensione di 22 byte mentre quelli ricevuti, di cui ne è presentato un esemplare sulla sinistra, hanno una dimensione di 60 byte perché lo standard di comunicazione 802.3 prevede lo zero-padding per tutti quei segmenti di lunghezza inferiore al minimo consentito. La minima lunghezza è 64 ma gli ultimi 4 byte sono di CRC (Circular Redundancy Code) i quali però non vengono mostrati da Ethereal essendo inutili a ricezione correttamente avvenuta.

Un'altra cosa importante da notare è che lo stack RTmac risulta essere completamente trasparente rispetto alla comunicazione TCP/IP su Ethernet questo però non significa che sia possibile collegare una rete RTnet ad una convenzionale senza perdere la predicibilità tipica della rete RTnet.

3.3.3 Misure di jitter in condizioni di stress

Per studiare il comportamento di RTnet sotto carico sono state prese in considerazione diverse fonti di stress per il sistema, ognuna delle quali può applicarsi tanto al PC-master quanto al PC-slave, qui sotto elencate:

- per il carico di rete
  - `ping -l 8192 -fqb 10.0.0.255`
- per il carico di I/O
  - `hdparm -d 0 /dev/hda; cat /dev/hda >/dev/null`

[20] RTmac specification - `http://www.rts.uni-hannover.de/rtnet/lxr/source/Documentation/RTmac.spec`
[21] Ethereal - `http://www.ethereal.com`

Il sistema di test è composto dal PC-master impersonato dal Celeron e da un PC-slave impersonato dal laptop. La tabella seguente mostra le combinazioni valutate e la relativa misura dello jitter massimo:

Master	**Slave**		
	scarico	*ping-flood*	*cat-disk*
scarico	9 uS	-	11 uS
ping-flood	10 us	-	-
cat-disk	19 uS	-	-

Le tre misure di valore 9, 10, 11 uS sono fra di loro compatibili con gli errori di misura che sono stimati in 1 uS[22] mentre dal confronto fra le misure di valore 9 e 19 uS si deduce che l'impatto del stress I/O provocato sul master è determinante per la precisione dell'intero sistema. La documentazione di RTnet sostiene che il sistema è grado di gestire la comunicazione di rete non real-time e il test di *ping-flood* ha verificato che in effetti il determinismo del sistema non viene modificato di una quantità apprezzabile. L'architettura di RTnet prevede che sia il master a fornire la sincronizzazione all'intero sistema quindi non sorprende che l'impatto del *cat-disk* sia molto diverso a seconda se a essere stressato è il master piuttosto che lo slave.

Le celle della colonna slave *ping-flood* sono vuote perché è stato supposto[23] essere indifferente su quale nodo viene generata la congestione del canale non real-time. Le altre celle sono vuote in quanto composizione di diversi carichi quindi situazioni più complesse inutili da studiare in questa fase preliminare.

3.3.4 Conclusioni

La scelta degli strumenti software è risultata appropriata ma a priori non era scontato quindi la prima parte di questo studio ha avuto lo scopo di impratichirsi con RTAI e RTnet e verificare la loro adeguatezza rispetto allo scopo di questo studio. Inizialmente sono state fatte delle misurazioni utilizzando gli strumenti di misura dei tempi forniti con il modulo RTmac di RTnet e successivamente questi strumenti sono stati adattati per le misure che interessavano. Infine si è sviluppata una gestione adeguata basandosi sul modello NOMAC che RTnet mette a disposizione per lo sviluppo di moduli personalizzati.

[22] è stato indicato proprio questo valore perché è la massima deviazione della media all'interno di uno stesso blocco di misure

[23] Il ping-flood ha come destinatario l'indirizzo di broadcast quindi tutti i nodi riceveranno la medesima quantità di pacchetti non real-time e ognuno di loro risponderà. A regime si suppone che anche il carico di CPU dovuto alla gestione del ping in kernel space sarà uguale su tutti i nodi. In questo caso ve ne sono solo due e dotati di CPU grosso modo equivalenti. In casi più generali i nodi dotati di minore potenza di calcolo potrebbero mostrare una maggiore sofferenza.

4 Misure di tempo in spazio kernel

"All'informatico spesso servono misure di ritardi molto precise, per esempio per valutare il tempo di esecuzione delle procedure e poterne ottimizzare la resa. Il conteggio dei cicli dell'orologio di sistema è una misura inaccettabilmente grossolana a questi scopi, indipendentemente dalla velocità del processore: un processore più veloce può gestire più interruzioni, ma solo perché svolge più lavoro nello stesso lasso di tempo; non possiamo mai avere più di una interruzione ogni molte migliaia di istruzioni macchina, ma un'interruzione molto frequente modifica sostanzialmente le prestazioni del sistema sotto esame, rendendo vana la misura." - Alessandro Rubini da "Misure di tempo" [17]

4.1 L'uso del Time Stamp Counter

Le CPU della famiglia Pentium, ma anche altri processori moderni, sono dotati di un registro a 64 bit che incrementa ad ogni ciclo di clock. Questo registro è chiamato TSC: Time Stamp Counter [18]. Ovviamente su CPU a 32 bit ogni volta che risultassero sufficienti si leggono di questo registro solamente i 32 bit meno significativi aumentando l'efficienza della lettura e della successiva elaborazione.

A meno di overflow questo contatore, cioè la differenza fra due letture, permette una misura molto precisa degli intervalli di tempo purché si conosca con precisione la frequenza di lavoro della CPU altrimenti l'intero ottenuto non può essere convertito in unità temporali.

Se si vuole confrontare delle misure di tempo ottenute con la lettura del TSC su due sistemi differenti allora bisogna tenere di conto anche della precisione dei due diversi quarzi. Analogamente se si confrontano due misure fatte sulla stessa macchina ma in periodi di tempo abbastanza distanti allora queste due misure debbono essere considerate come se fossero state prese su due macchine differenti e l'errore del quarzo dovrebbe teoricamente essere considerato.

Linux mette a disposizione la costante `CPU_KHZ` per conoscere la frequenza di lavoro della CPU. Poiché questa è una costante che viene inizializzata alla partenza del sistema è possibile che durante il funzionamento il quarzo presenti un drift nella frequenza ad esempio dovuto al riscaldamento della macchina. Poiché l'errore del quarzo comprensivo del drift termico è stimato nell'ordine di parti su milione questo contributo verrà sistematicamente ignorato nel proseguo di questo studio perché ininfluente.

L'overflow di un contatore a 32 bit è di facile gestione infatti nel misurare l'intervallo si fa la differenza usando variabili di tipo `unsigned long`, purché si abbia la certezza di misurare intervalli temporali più brevi del tempo di overflow cioè, per una CPU che lavori a 4 GHz:

$$\frac{2^{32}}{CPU\_KHZ \cdot 1000} \sim 1 \text{ secondo}$$

4.1.1 L'importanza del TSC

Per leggere questo contatore esiste una specifica funzione ma se nella configurazione del kernel Linux non si è avuta l'accortezza di specificare un processore che abbia il TSC tale funzione, e tutte quelle che da essa dipendono, interrogano il Real Time Clock (RTC) mediante il chipset 8254 producendo due effetti:

- un carico di sistema molto maggiore
- una granularità del tempo assai più grossolana.

Questo significa che la sincronizzazione di processi real-time avviene con un errore dell'ordine della metà del più piccolo intervallo di tempo apprezzabile attraverso lo 8254 [20].

Inizialmente a causa di una svista il supporto TSC risultava disabilitato sul PC-master e quindi tutte le misure della fase di verifica di RTnet risulteranno affette da latenze piccole ma da jitter molto grandi. Ovviamente qualsiasi tentativo di ridurre l'ampiezza di questi jitter si è rivelata un'impresa senza speranza, nonostante l'uso della modalità *one-shot* e del *busy-loop*[24] per sincronizzare con precisione il ciclo, fintanto che il supporto TSC non è stato attivato nella configurazione del kernel e quest'ultimo ricompilato.

In seguito le serie di misure effettuate senza supporto TSC saranno espressamente differenziate e sono comunque state riportate perché dal punto di vista del tempo medio di ciclo esse rimangono significative mentre dal punto di vista degli jitter massimi è interessante fare il confronto per capire quanto realmente questo registro sia importante: a parità di ciclo lo jitter di attivazione di un task periodico real-time passa dall'intervallo $1 \div 0.2$ mS senza TSC fino a ridursi all'intervallo $1 \div 0.2$ uS nel caso di TSC disponibile, modalità *one-shot* e *busy-loop*.

4.2 Il calcolo dei tempi

Precedentemente si è mostrato che la trasformazione da valori TSC a unità temporali si ottiene con una divisione ma quest'operazione aritmetica in kernel space è limitata a 32 bit sui processori a 32 bit, salvo modifiche al kernel, inoltre risulta essere l'operazione aritmetica di gran lunga più impegnativa per la CPU.

Il problema del calcolo dei tempi non è nelle prestazioni in quanto le CPU moderne sono talmente potenti da non risentire di questa occupazione all'interno di un ciclo di 128 uS. Invece la criticità risiede nella precisione dei calcoli infatti la funzione `rt_get_cpu_time_ns()` fornita da RTAI produce un risultato che ha una granularità di 244,14 nS che sicuramente nel contesto di RTAI è più che accettabile e tutto sommato anche nel contesto di questo progetto.

Vista anche la valenza didattica di questa tesi si è comunque deciso di affrontare il problema del calcolo dei tempi senza utilizzare le funzioni specifiche di RTAI ma sviluppando una serie di macro in grado di massimizzare la precisione del calcolo in riferimento al periodo impostato. Con ciò non significa che le funzioni di RTAI siano state completamente ignorate anzi, a parte le funzioni di statistica che fanno la misura dei tempi leggendo direttamente il TSC, sono state impiegate ovunque compreso il *busy-loop* cioè nella sincronizzazione di precisione dell'attivazione del ciclo principale.

4.2.1 Aritmetica intera a 32 bit

La necessità di convertire le differenze fra diverse letture di TSC in unità temporali nasce dalla modalità di raccolta delle misure e dal fatto che se le statistiche fossero fatte direttamente sulle differenze di TSC si avrebbe l'overflow delle variabili a 32 bit già per il calcolo della media quando il calcolo della varianza richiede la somma di quadrati.

Le soluzioni possibili quindi sono due: memorizzare tutte le letture e passarle in user space in maniera da elaborarle con il supporto per la virgola mobile oppure convertire le differenze TSC in unità temporali e usare un thread periodico in spazio kernel per stampare i risultati delle statistiche per ogni campione di misure.

Avendo scelto la seconda modalità si vuole fare la seguente operazione con aritmetica intera a 32 bit e cercando di massimizzare la precisione del risultato:

[24] Questi due termini sono funzionalmente correlati e sono commentati nella sezione 5.4.5 dove *one-shoot* corrisponde al ciclo aperiodico e il *busy-loop* è un espediente per compensare le fluttuazioni di attivazione.

$$\triangle T_{nS} = \frac{TSC_2 - TSC_1}{CPU_{KHz}} \cdot 10^6 = \frac{d}{k} \cdot 10^6$$

si verifica per quanto riguarda gli ordini di grandezza, in un ambito analogo a quello tipico di RTAI:

$k \leq 2^{22}$ per valori di CPU clock non maggiori di 4 GHz
$d \leq 2^{25}$ per valori di periodo non maggiori di 8 mS
$10^6 \leq 2^{20}$primo maggiorante in base 2

Quindi se l'operazione di divisione fosse eseguita senza precauzioni si otterrebbe un risultato che nel migliore dei casi è significativo per un bit solamente e nella maggior parte dei casi è nullo. Se all'operazione di divisione si facesse precedere la moltiplicazione $d \cdot 10^6 < 2^{45}$ questa potrebbe mandare in overflow la variabile. Si consideri invece la seguente:

$10^6 = 15625 \cdot 2^6$ scomposizione intera con massimo fattore in base 2

allora il calcolo sopra si può fare così:

$$\triangle T_{nS} = \frac{d \cdot 2^6}{k} \cdot 15625$$

in questo modo i bit significativi sono al massimo 25+6-22 = 9 ma più di ogni altra cosa la risoluzione è di 15625 nS. Si noti che k e 15625 sono entrambe costanti una volta che il sistema ha fatto il boot, quindi:

$$\triangle T_{nS} = \frac{d \cdot 2^6}{k/15625} = \frac{d \cdot 2^6}{k'}$$

in questo modo però k' è una variabile di cui al massimo sono significativi solo 8 bit quindi comporta avere una risoluzione di $\frac{1}{256}$ del valore del periodo. Una buon sistema per portare ad almeno 16 bit significativi il denominatore potrebbe essere, ad esempio:

$$\triangle T_{nS} = \frac{d \cdot 2^6}{k/2^6} \cdot \frac{15625}{2^6} = \frac{d'}{k'} \cdot 244,140625$$

che è proprio il valore corrisponde alla granularità misurata della funzione `rt_get_cpu_time()` di RTAI.

Attraverso questi progressivi tentativi si capisce che se si vuole ulteriormente aumentare la risoluzione temporale non ci si può basare su assunzioni circa il periodo e la frequenza massime ma calcolare i fattori di adattamento della divisione in modo dinamico:

$$\left\{ \begin{array}{l} \left.\begin{array}{l} m = 31 - log_2\left(T_{uS}^{ciclo} \cdot CPU_{MHz}\right) \\ n = log_2\left(CPU_{KHz}\right) - 16 \\ k'' = \frac{CPU_{KHz}}{2^n} \end{array}\right\} cost. \\ \triangle T_{nS} = \left(\left(\frac{(d \cdot 2^m)}{k''}\right) \cdot 15625\right) \cdot \frac{2^6}{2^{n+m}} \end{array} \right.$$

Per una CPU a 1.2 GHz con un ciclo di 128 uS si ottiene $n = 5$ e $m = 13$ da cui si calcola che la risoluzione temporale è inferiore a 4 nS, esattamente:

$$dt_{nS} = \frac{15625}{2^{n+m-6}} \cong 3.815$$

Una tale risoluzione temporale è fondamentale per misurare il tempo di esecuzione di una breve sequenza di istruzioni macchina.

4.2.2 Note sull'implementazione

Nel codice sorgente del modulo NOMAC nella versione con TSC abilitato la formula precedentemente mostrata è calcolata in modo più contorto di quanto qui esposto e anche in modo molto più disinibito[25] per cui vengono utilizzati $n = 6$ e $m = 14$ per un tempo di ciclo di 128 uS e una frequenza di processore di 1595 MHz. Questi valori sono al limite dell'overflow delle variabili in gioco e quindi l'intero sistema di calcolo non offre nessun margine di sicurezza in caso si abbiano oscillazioni di periodo, cioè jitter, relativamente non trascurabili. Ovviamente poiché abbattere drasticamente l'ampiezza degli jitter è proprio il compito principale del codice inserito nel modulo NOMAC questa disinvoltura nel fare i calcoli non ha creato problemi di correttezza nelle statistiche ma è bene tenerne conto se si vuole riutilizzarne il codice.

I valori di n, m e k'' sono calcolati al momento dell'inizializzazione del modulo NOMAC e rimangono costanti per il resto del tempo. Le moltiplicazioni o divisioni per potenze di due si implementano mediante *bit-shift* opportuni mentre la divisione intera arrotondata e il logaritmo in base due si implementano cosi:

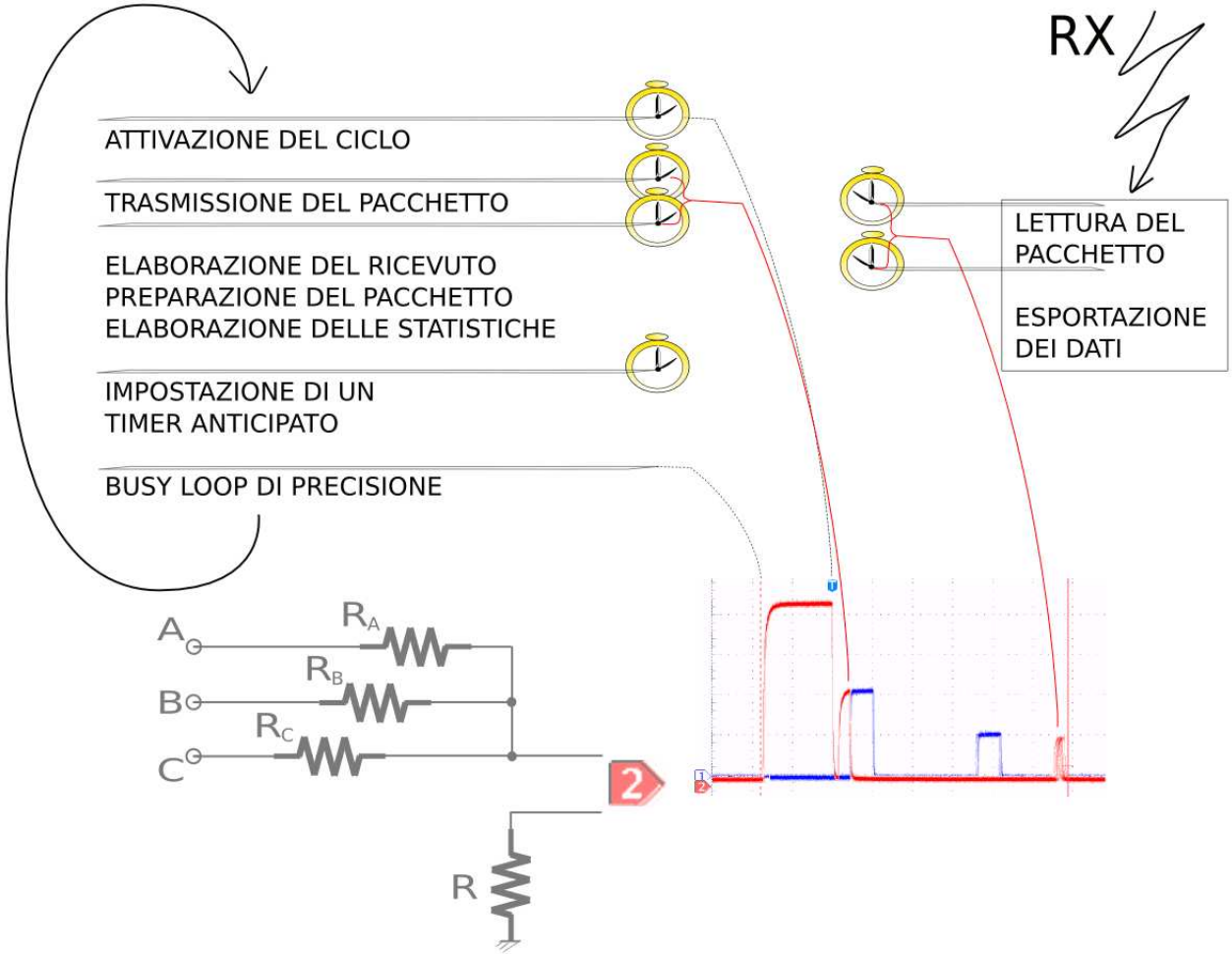
```
#define adiv(d, k) ((d + (k>>1)) / k)

#define log2(x) ({unsigned long i=0; for(i;x>0;i++) x=(x>>1); i;})
```

Alternativamente alle macro si possono usare delle funzioni inline che in più garantiscono il *type-checking*. Un vezzo per nulla trascurabile anche dal punto di vista dei calcoli considerato che la divisione di una variabile di tipo `unsigned long` da parte di una di tipo `long` produce risultati errati quando si fanno calcoli al limite dell'overflow.

[25] Questa distanza fra teoria qui esposta e implementazione nel codice è dovuta al fatto che la pratica ha preceduto la teoria in quanto solo a posteriori mi sono posto il problema di giustificare le costanti trovate e criticare le soluzioni adottate. Spero nessuno si scandalizzi nel leggere che una qualche esperienza pratica abbia preceduto la comprensione del fenomeno!

5 Misure in laboratorio

Il sistema di raccolta delle misure è basato su una funzione che viene attivata a ogni chiamata del task real-time che calcola il tempo fra due specifiche letture di tempo. La raccolta avviene per campioni di molte decine o diverse centinaia di misure e di questi campioni si conoscono solo valori d'insieme quali: minimo, media, massimo, numero di misure nel campione e per alcuni tipi di misura anche la varianza. Queste misure sono collezionate in una struttura dati fino al momento in cui si attiva un task periodico non real-time che si occupa di stamparle nel log di sistema e contrassegnare la struttura dati come da annullare.



Il grafico sopra mostra i punti di misura e la loro proiezione temporale sul ciclo e la discussione teorica che seguirà nella prossima sezione dovrebbe risultare più chiara se confrontata con questa figura.

Sul diagramma preso dall'oscilloscopio il fronte di discesa contrassegnato dalla T blu è l'istante di attivazione. In seguito nella sezione 5.4 si vedrà che tale riferimento è molto stabile e quindi assolutamente ragionevole averlo indicarlo all'oscilloscopio come trigger su cui sincronizzarsi per graficare gli altri eventi.

In fondo all'ascissa si trova la forma d'onda relativa alla funzione di ricezione che appare assai meno stabile ma è proprio questa caratteristica ciò che si vuole quantificare quando si parla di jitter.

Infine nella figura si vede anche lo schema del tripartitore resistivo utilizzato per combinare diversi segnali in un solo canale dell'oscilloscopio.

5.1 L'intima natura dello jitter

Vale la pena di fare qualche considerazione per capire la natura di questo tipo di misura. L'intervallo di tempo fra la trasmissione di un pacchetto e la ricezione della risposta dipende da diversi fattori alcuni di questi incidono sul tempo di andata del pacchetto, altri sul ritorno, alcuni sono costanti e altri variabili.

La seguente equazione esplicita alcune di queste quantità variabili e le mette in relazione con le tempistiche relative a un n-esimo pacchetto:

$$\begin{aligned} T_{rx}(n) = & \quad T_{rtx}(n) & +\Delta T_{volo} + \Delta T_{rx} \\ = & \quad T_{tx}(n) & +\Delta T_{volo} + \Delta T_{trx} \quad +\Delta T_{volo} + \Delta T_{rx} \end{aligned} \qquad \text{(eq. 1)}$$

Per togliere dalla scena i termini costanti si procede con un confronto con il pacchetto successivo:

$$\begin{aligned} T_{rx}(n+1) - T_{rx}(n) &= T_{tx}(n+1) && +\Delta T_{volo} && +\Delta T_{rtx} + \Delta T_{volo} && +\Delta T_{rx} \\ &- T_{tx}(n) && -\Delta T_{volo} && -\Delta T_{rtx} - \Delta T_{volo} && -\Delta T_{rx} \\ &\leq T_{tx}(n+1) - T_{tx}(n) && && +\delta T_{rtx}^{max} && +\delta T_{rx}^{max} \end{aligned}$$

Riordinando tutti i termini in n a sinistra e focalizzando l'attenzione sull'estremo superiore degli intervalli di variabilità delle quantità in gioco si ottiene la seguente disequazione:

$$|(T_{rx}(n+1) - T_{tx}(n+1)) - (T_{rx}(n) - T_{tx}(n))| \leq \delta T_{rtx}^{max} + \delta T_{rx}^{max}$$

Indipendentemente da come è stata trovata la disequazione sopra essa è generalmente vera per qualsiasi coppia $(n,\, n+m)$. In particolare è vera per quei valori tali per cui:

$$\Delta T_{rx-tx}^{max} - \Delta T_{rx-tx}^{min} \leq \delta T_{rtx}^{max} + \delta T_{rx}^{max}$$

Se questa sopra fosse un'equazione a sinistra si inserirebbero i valori sperimentali mentre a destra si troverebbe la quantità cercata δT_{rx}^{max} a patto di conoscere o poter trascurare lo jitter del risponditore. Più avanti nella sezione 5.3 saranno affrontati i vari casi determinati dal rapporto fra δT_{rtx}^{max} e δT_{rx}^{max}.

La quantità δT_{rx}^{max} ha una peculiarità a cui fare attenzione: le latenze costanti non influiscono ne sulla significatività delle misure e neppure sul loro valore purché la loro somma non sia sistematicamente superiore al periodo altrimenti occorre imporre un contatore progressivo al pacchetto in modo da poter correlare l'istante di trasmissione ad esso associato con l'istante di ricezione, oppure ovviamente, inserire il tempo di trasmissione nel pacchetto stesso in maniera che esso sia preservato durante il tragitto.

5.1.1 Jitter di un evento periodico

Una maniera molto più semplice di fare una misura analoga è quella di calcolare l'intervallo di tempo fra un evento di ricezione e il successivo. Bisogna però capire quanto questo metodo sia significativo e in che maniera rapportarlo al precedente.

Si riprende l'equazione n.1 della sezione 5.1 con lo scopo di espandere il tempo di trasmissione in maniera da esplicitare anche questo contributo variabile:

$$\begin{aligned} T_{rx}(n) = & \quad T_{tx}(n) \quad + \Delta T_{volo} + \Delta T_{trx} \quad + \Delta T_{volo} + \Delta T_{rx} \\ = & \quad T_{ax}(n) + \Delta T_{tx} \quad + \Delta T_{volo} + \Delta T_{trx} \quad + \Delta T_{volo} + \Delta T_{rx} \end{aligned}$$

dove $T_{ax}(n)$ è l'istante n-esimo di attivazione del task di trasmissione che risulta utile per togliere dalla scena i termini costanti analogamente a quanto fatto sopra:

$$\Delta T^{max}_{rx'-rx} - \Delta T^{min}_{ax'-ax} \leq \delta T^{max}_{tx} + \delta T^{max}_{trx} + \delta T^{max}_{rx}$$

considerando che $\Delta T^{min}_{rx'-rx} \leq \Delta T^{min}_{ax'-ax} \leq \Delta T_{ciclo}$ e definendo $\delta T^{max}_{trx} = \delta T^{max}_{tx} + \delta T^{max}_{rx}$ allora si ottiene:

$$\Delta T^{max}_{rx'-rx} - \Delta T_{ciclo} \leq \delta T^{max}_{rtx} + \delta T^{max}_{trx}$$

escludendo la perdita di pacchetti[26] si osserva che i pacchetti trasmessi prima o dopo devono essere ricevuti quindi su un tempo abbastanza lungo la media dell'intervallo calcolato fra due ricezioni successive coincide con il periodo di trasmissione:

$$media_{n=1\ldots G}\left(T_{rx}(n+1) - T_{rx}(n)\right) \longrightarrow \Delta T_{ciclo} \text{ per } G \rightarrow = +\infty$$

In particolare se le misure sono raccolte a campioni di centinaia o migliaia questa condizione risulta statisticamente sempre vera quindi la disequazione è simmetrica rispetto al valore ΔT_{ciclo} cioè:

$$\left(\Delta T^{max}_{rx'-rx} - \Delta T_{ciclo}\right) + \left(\Delta T_{ciclo} - \Delta T^{min}_{rx'-rx}\right) = \Delta T^{max}_{rx'-rx} - \Delta T^{min}_{rx'-rx} \leq 2\left(\delta T^{max}_{rtx} + \delta T^{max}_{trx}\right)$$

La quantità δT^{max}_{trx}, per definizione, contiene in sé anche il contributo dello jitter di trasmissione e quest'ultimo contiene il contributo dello jitter di attivazione quindi si tratta di una quantità complessa e meno facilmente associabile all'evento della mera ricezione rispetto alla precedente. Nonostante questo qualora lo jitter di trasmissione risultasse relativamente trascurabile allora δT^{max}_{trx} risulterebbe essere una buona misura.

Un'altra differenza importante è che questa quantità risulta essere uno jitter correlato ad un evento periodico e come tale ha un'ampiezza doppia rispetto a quelli misurati sugli intervalli non periodici. In buona sostanza per confrontare onestamente δT^{max}_{trx} con δT^{max}_{rx} bisogna moltiplicare per due quest'ultimo oppure dividere il primo.

[26] Nel momento che si inseriscano pozzi e sorgenti di pacchetti all'interno della rete il determinismo in ricezione non può più essere garantito mentre invece è ancora possibile garantirlo se sulla rete vi sono ritardi e/o condensatori di pacchetti.

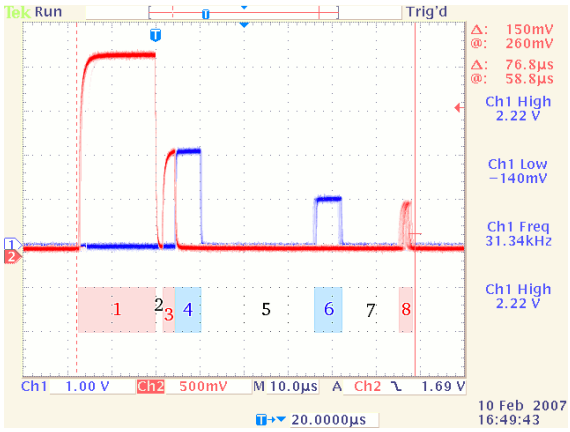
5.2 Misure all'oscilloscopio

In questa pagina sono raccolti alcuni screenshot dello schermo dell'oscilloscopio: sul primo canale il segnale in blu è relativo all'attivazione del controllore Ethernet sulla Zefeer mentre sul secondo canale il segnale è quello preso da tre piedini della parallela diversamente modulati grazie a un tripartitore resistivo. I tre piedini della parallela vengono pilotati all'interno del modulo NOMAC e permetto di stabilire la sequenza delle operazioni.

Nella prima figura è stata evidenziata proprio questa sequenza di eventi:

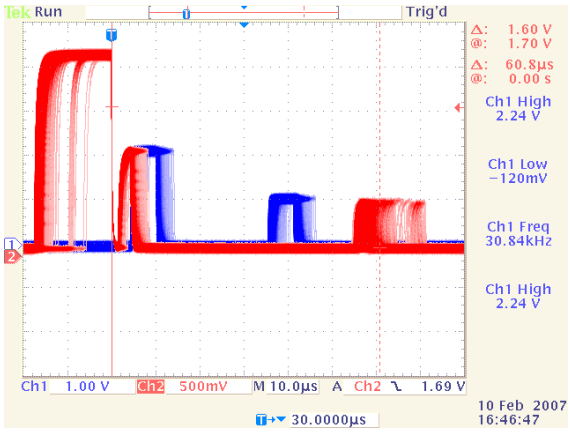
1	busy-loop	17.6 uS	5	elaborazione del risponditore	25.6 uS
2	transitorio di trasmissione	1.6 uS	6	volo di ritorno a 100 Mbit/s	6.4 uS
3	trasmissione	2.8 uS	7	transitorio di ricezione	12.8 uS
4	volo di andata a 100 Mbit/s	6.4 uS	8	ricezione	3.2 uS

1. ciclo chiuso su cavo incrociato

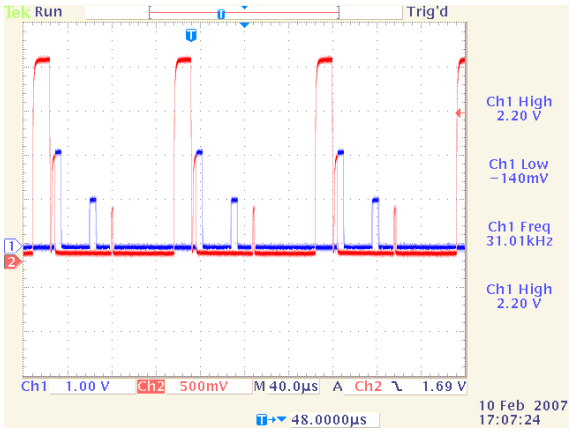


4. ciclo chiuso con HUB intermezzo

2. ciclo chiuso con sistema sotto stress



3. ciclo chiuso: vista multi-periodica



Si noti che nella figura n.2 oltre alle latenze si vedono anche le loro variazioni, cioè gli jitter, che si manifestano come allargamento dei fronti di salita e discesa. Mentre nella figura n.4 si noti che l'inserimento di un HUB a 10 Mbit/s fra le parti ha allargato la chiusura del ciclo da 76 uS a 192 uS avendo aumentando di quasi 10 volte i tempi di volo del pacchetto, ognuno dei quali è passato da 6.4 uS a 57.6 uS[27].

[27]Si noti che questo tempo 57.6 uS compare 3 volte nel grafico n.4: in andata e al ritorno fra 1^o- 3^o fronte e fra 2^o- 4^o fronte. Inoltre $57.6 = 64 - 6.4$ uS corrisponde esattamente al tempo di volo necessario a 10 Mbit/s meno il tempo a 100 Mbit/s, forse solo per una coincidenza.

5.3 Misura dello jitter di ricezione

Nella misura dello jitter l'unico aspetto interessante da osservare è il massimo registrato per campione infatti la distribuzione di questi massimi, sopratutto in condizioni di stress del sistema, ci informa circa la precisione di un evento rispetto all'istante predeterminato: sia esso la partenza di un task oppure la ricezione di una risposta. Consideriamo due sistemi:

- il PC-master invia periodicamente pacchetti e si occupa di misurare il tempo trascorso fra una ricezione e la successiva;
- il risponditore si limita a ricevere un pacchetto e a rimandarlo al mittente nel minor tempo possibile.

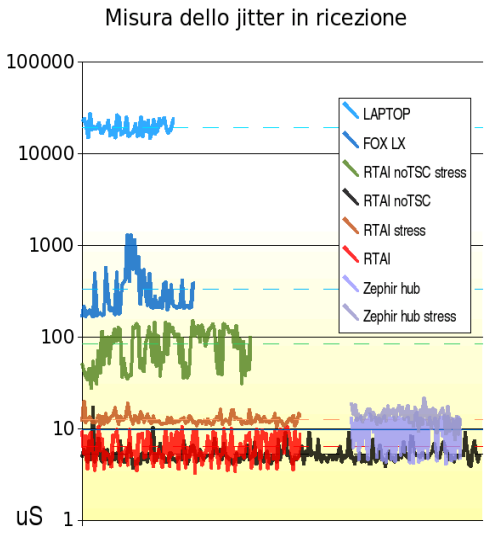
Supponendo che il PC-master trasmetta con perfetta regolarità allora i tempi misurati in ricezione permettono di valutare la stabilità dei tempi del risponditore. Invece supponendo che sia il risponditore a operare senza alcuno jitter allora si può valutare la stabilità del PC-master. La tabella seguente riassume i casi possibili:

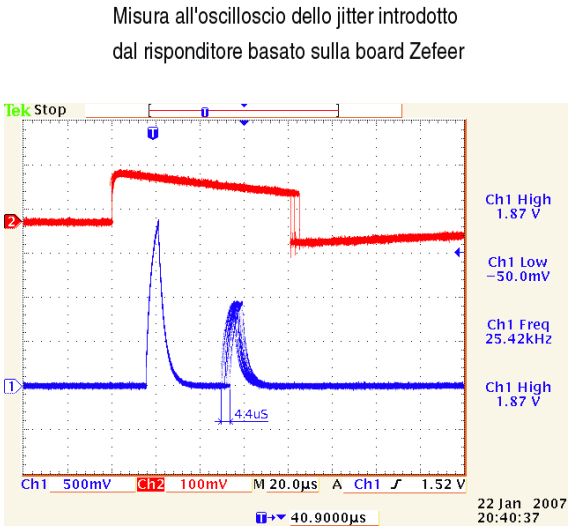
	Condizione	Misura
1	Tx jitter PC-master $\ll$ jitter risponditore	massimo jitter del risponditore
2	Tx jitter PC-master $\sim$ jitter risponditore	una composizione degli jitter
3	Tx jitter PC-master $\gg$ jitter risponditore	massimo Rx jitter del PC-master

Nel secondo caso supponendo che entrambi i sistemi abbiano esattamente il medesimo comportamento e che lo dimostrino in modo del tutto indipendente l'uno dall'altro allora la misura si riferisce alla somma dello jitter massimo di entrambi i sistemi.

5.3.1 Analisi delle misure

Le misure sono presentate in un grafico che mostra l'andamento, per diversi sistemi, del massimo jitter di ricezione inteso come $jitter_{max} = \Delta T^{max}_{rx'-rx} - \Delta T^{min}_{rx'-rx}$:





Rientra nel caso 3:
l'Atmel AVR ATMEGA-88 ha latenze di decine di microsecondi ma uno $jitter_{max}$ stimato[28] inferiore al microsecondo quindi riflette fedelmente il PC-master Celeron 1.2 GHz con 2.6.16 real-time, senza TSC, sotto carico di cui lo $jitter_{max}$ varia fra 30 e 150 uS.

Buona approssimazione del caso 3:
analogamente al caso precedente l'Atmel riflette il PC-master Celeron 1.2 GHz con kernel 2.6.16 e RTAI di cui lo $jitter_{max}$a seconda della situazione varia: senza TSC fra 4 e 17 uS, con TSC fra 3 e 10 uS e infine con TSC sotto stress fra 10 e 20 uS.

Esempio del caso 2:
la scheda Zefeer con la prima versione di risponditore U-BOOT senza IRQ inserisce uno $jitter_{max}$ di circa 4 uS, come evidenzia la misura fatta con l'oscilloscopio, interagisce con il PC-master configurato RTAI con TSC; complessivamente lo $jitter_{max}$ risulta essere: atteso[29] fra 4 e 14 uS, misurato 4 e 15 uS; invece in condizione di stress risulta essere: atteso fra 10 e 24 uS, misurato 7 e 22 uS.

Buona approssimazione del caso 1:
Acme Systems FOX LX con CPU CRIS 100MHz, kernel 2.4.31, sistema dedicato alla risposta ma non real-time, di cui lo $jitter_{max}$ varia fra 0.2 e 1.3 mS questa misura è due ordini di grandezza maggiore di quella sopra quindi il contributo del PC-master in essa risulta trascurabile.

Caso 1 con accodamento di pacchetti:
laptop con kernel 2.6.12 configurato per l'uso come desktop e KDE in funzione di cui lo $jitter_{max}$ varia fra 15 e 27 mS.

In questo caso il laptop non rispondeva al singolo pacchetto ma li accodava per poi ottimizzare la loro elaborazione. Questa ipotesi è stata suggerita dal fatto che il minimo risultava essere una misura inverosimilmente troppo piccola rispetto al periodo e la spiegazione più plausibile è che i pacchetti arrivati al risponditore venivano messi in una coda e poi elaborati insieme per ottimizzare le prestazioni in media. Perciò le risposte a diversi pacchetti ritornavano insieme al trasmettitore sul quale si è dovuto disattivare, appunto, il controllo sul numero sequenziale[30] che identifica il pacchetto e relativa risposta.

In questo particolare caso il minimo rappresenta il tempo necessario per elaborare un pacchetto sulla coda di ricezione da parte del PC-master ma poiché nelle misure fatte esso corrisponde alla granularità temporale l'unica cosa che si può dire è che tale elaborazione non richiede mai un tempo più grande di 244 nS oppure, in altre parole, che lo jitter di elaborazione è nullo o comunque inferiore alla risoluzione temporale.

[28] C'è un'oscillazione nella ricezione, perché il codice che controlla il piedino della Ethernet è costituito da un ciclo stretto di due istruzione la cui variabilità è pari appunto a due istruzioni macchina su una CPU a 10 MHz. Anche considerando alcuni cicli macchina per istruzione lo jitter massimo si stima inferiore al microsecondo.

[29] Il risultato della composizione degli jitter massimi è la somma aritmetica degli stessi mentre il risultato della composizione dei minimi degli jitter massimi equivale al valore del più grande fra questi minimi

[30] Si incrementa un contatore di errori qualora la risposta a un pacchetto arrivi dopo l'invio del pacchetto successivo. Per come è stato realizzato il sistema di comunicazione (cfr. nota 5 della sez. 1.4 e sez. 5) la preparazione del pacchetto successivo avviene con i dati ottenuti in risposta al precedente senza i quali il controllo è da considerarsi a "ciclo aperto" e quindi una segnalazione d'errore è d'obbligo.

5.3.2 Tabella comparativa per tipologia

Nella tabella che segue si confrontano le misure dei due differenti jitter sopra discussi misurati nel caso particolare in cui, essendo possibile trascurare il contributo del risponditore, è possibile isolare il termine legato al PC-master:

TSC	**TSC stress**	**noTSC**	**noTSC stress**	**caso 3: δT_{rtx}^{max} trascurabile**
$4 \div 6$ uS	$5 \div 9$ uS	$2 \div 33$ uS	$24 \div 75$ uS	$\Delta T_{rx-tx}^{max} - \Delta T_{rx-tx}^{min} \leq \delta T_{rx}^{max}$
$3 \div 10$ uS	$10 \div 20$ uS	$4 \div 17$ uS	$30 \div 150$ uS	$\Delta T_{rx'-rx}^{max} - \Delta T_{rx'-rx}^{min} \leq 2 \cdot \delta T_{trx}^{max}$

La relazione generale sottesa è che i valori della seconda riga sono doppi di quelli della prima eccezione fatta per la terza colonna per la quale appaiono invertiti. Questo non deve stupire infatti misurando il sistema in quiete in realtà si afferma solamente che non vi è alcun carico aggiunto al sistema durante la misura. Questo però non garantisce che non vi sia del carico proprio del sistema stesso. Per verificare questa affermazione si può analizzare le due serie di dati con un indicatore robusto quale la mediana che risulta essere rispettivamente di 2.9 e 5.1 uS cioè fra loro in proporzione corretta salvo sporadici picchi di carico che invece probabilmente inquinato le misure di massimo.

5.3.3 Tabella comparativa per ordine di grandezza

In genere in queste tabelle riportano i valori di latenza media ma si consideri che nella fascia più alta jitter e latenze hanno, necessariamente, lo stesso ordine di grandezza. Invece nelle fasce più basse latenza e jitter hanno significati e valori completamente diversi. In alcuni contesti la latenza non è un grande problema: più è grande e più tempo sarà necessario per dare risposta a uno stimolo mentre invece tanto più è grande lo jitter tanto più l'intervallo di tempo necessario alla risposta è indeterminato.

Sistema	**nS**	**categoria**
Laptop 2.6 con KDE in uso	10^7	multimedia
FOX 2.4, solo risponditore	10^6	“
PC 2.6 RTAI senza TSC sotto stress	10^5	software real-time
PC 2.6 RTAI senza TSC	10^4	“
PC 2.6 RTAI sotto stress	10^4	“
PC 2.6 RTAI	10^4	“
Zefeer Uboot-IRQ-less solo risponditore	10^3	firmware real-time
Atmel AVR solo risponditore	10^2	“

Se si volesse fare un controllo motori con lo scopo di pilotare gli attuatori di un robot affinché stia in equilibrio su due gambe sarebbe indispensabile che la risposta a uno sbilanciamento fosse abbastanza rapida da impedire l'irreversibilità della situazione, cioè si parla di sistemi real-time con dead-line critica. Altrimenti se il controllo motori ha lo scopo di far percorrere a uno strumento una data traiettoria è irrilevante quanto tempo impieghi il sistema a far partire i motori ma è fondamentale, invece, la precisione del movimento quindi la precisione del campionamento temporale: cioè si parla di sistemi real-time deterministici.

5.4 Misura dello jitter di attivazione

Nella misura dello jitter di attivazione di un task l'unica variabile interessante da osservare è ancora il massimo jitter registrato per campione infatti la distribuzione di questi massimi, sopratutto in condizioni di stress del sistema, ci informa circa la bontà del determinismo sotto il profilo della precisione di attivazione di un task rispetto all'istante predeterminato.

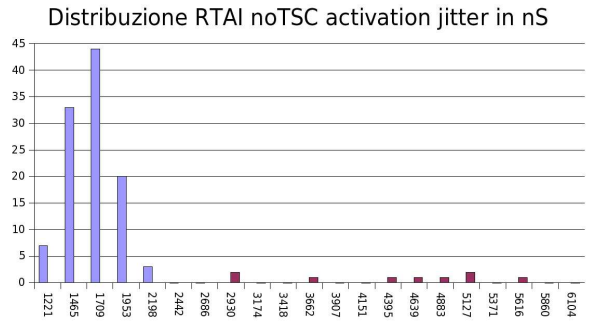


grafico ottenuto da 120 campioni per un totale di 23.847 misure
PC-master senza TSC configurato, ciclo periodico non compensato

Il sistema risponde con jitter massimi distribuiti principalmente fra 1000 e 2000 nS. Apparentemente il profilo sembra quello tipico di una distribuzione di Poisson con picco a 1700 uS e larghezza a metà altezza di 600 nS. Ma non si può far a meno di notare una coda di valori sospetta che si prolunga verso i 6000 nS. Infatti per una tale distribuzione oltre il valore di $1700 + 3 \cdot 600 = 3500$ nS ci si aspetterebbe di trovare al massimo una misura e non invece sette.

Questa coda è un'anomalia statisticamente rilevante[31] sintomo che il sistema in questione ha ancora molto da rivelare circa il suo comportamento.

Quando si mette il sistema in una situazione di stress il suo profilo di jitter cambia notevolmente al punto da richiedere una scala dei tempi molto più ampia per essere rappresentato.

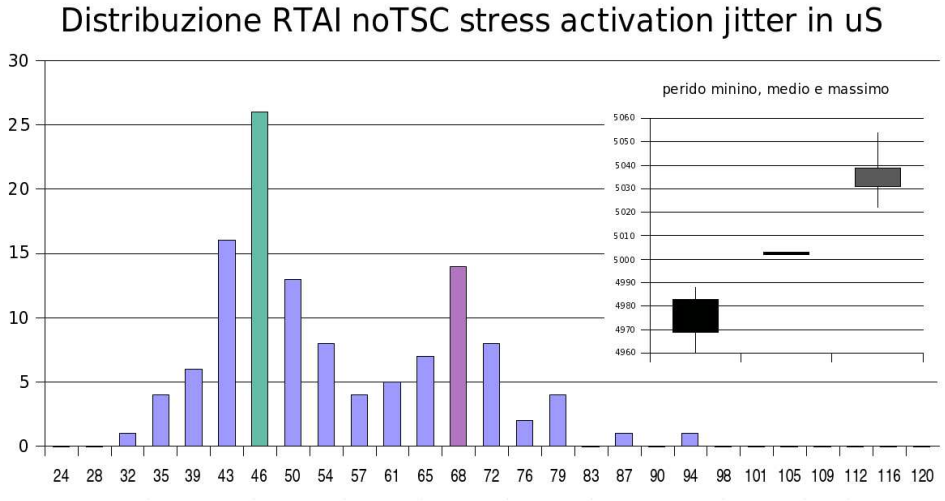


grafico ottenuto da 120 campioni per un totale di 23.847 misure
PC-master senza TSC configurato, ciclo periodico non compensato
stress ottenuto con il comando cat /dev/hda | od -x su ssh attiva

Il sistema sotto stress mostra una figura di jitter verosimilmente composta da due distribuzioni gaussiane centrate rispettivamente su 45 e 69 uS che contribuiscono con peso differente[32] al

[31] La stima è stata fatta usando la tabella delle probabilità rispetto alla deviazione standard di una gaussiana invece di una distribuzione di Poisson ma la sproporzione fra valore atteso e trovato è tale da non poter essere imputato a questo. Vedere anche `http://tenstep.it/ebook/04.09Metodologia-Sei-Sigma.htm`

[32] L'ampiezza di probabilità si misura come numero di campioni raccolti nel rettangolo più alto diviso il numero di campioni totali (ad es.: 26/120 = 22%, 14/120 = 12%). Invece il contributo percentuale, cioè la

profilo finale. Anche la stabilità del periodo ne risente mostrando visibili variazioni dell'ordine del $\pm 10\%$.

5.4.1 Possibili interpretazione delle misure raccolte

Ci sono due possibili spiegazioni per la distribuzione mostrata nel grafico precedente:

1. che sia la composizione di due cause indipendenti;
2. che vi sia una sola causa combinatoria di 23 uS che genera la serie:

   $(S_1 = 23)$, $S_2 = 46$, $S_3 = 69$, $S_4 = 92$, $(S_5 = 115)$, ...

L'ultima spiegazione è la più affascinante sotto vari aspetti:

- è semplice in quanto si fonda sull'esistenza di una sola causa;
- questo causa ha proprietà combinatorie intimamente legate al sistema;
- il termine fondamentale è nascosto dalla modalità di misura adottata.

Si noti che per il modo con cui sono state raccolte le misure il termine S_1, che ci si aspetta sia il più probabile di tutti in quanto fondamentale, diventa individuabile in un campione solo se tutte le misure in esso sono riferibili a S_1 infatti basta un solo valore riferibile a un termine successivo per spostare verso l'alto il massimo del campione nascondendo così il contributo di tutti quelli precedenti.

Una contro verifica sperimentale per negare l'esistenza di questo termine S_1 da 23 uS, e quindi dell'ipotesi nel suo insieme, è quella di ridurre il numero G di misure all'interno di un campione, cioè per $G \to 1$ il contributo del primo termine deve diventare sempre più evidente oppure è smentita la sua esistenza.

Invece l'esistenza del termine S_3 non si può escludere prendendo in esame solo questi dati, anche se non appare in modo statisticamente rilevante in questo grafico, dovrebbe diventarlo aumentando il numero di misure complessive.

5.4.2 Modello statistico risultante e relativa simulazione

Supponendo che vi sia un'unica causa e che essa si possa combinare più volte generando la serie sopra citata allora il fatto che il primo termine non si sia manifestato su $N = 120$ campioni mediamente di $G = 200$, cioè

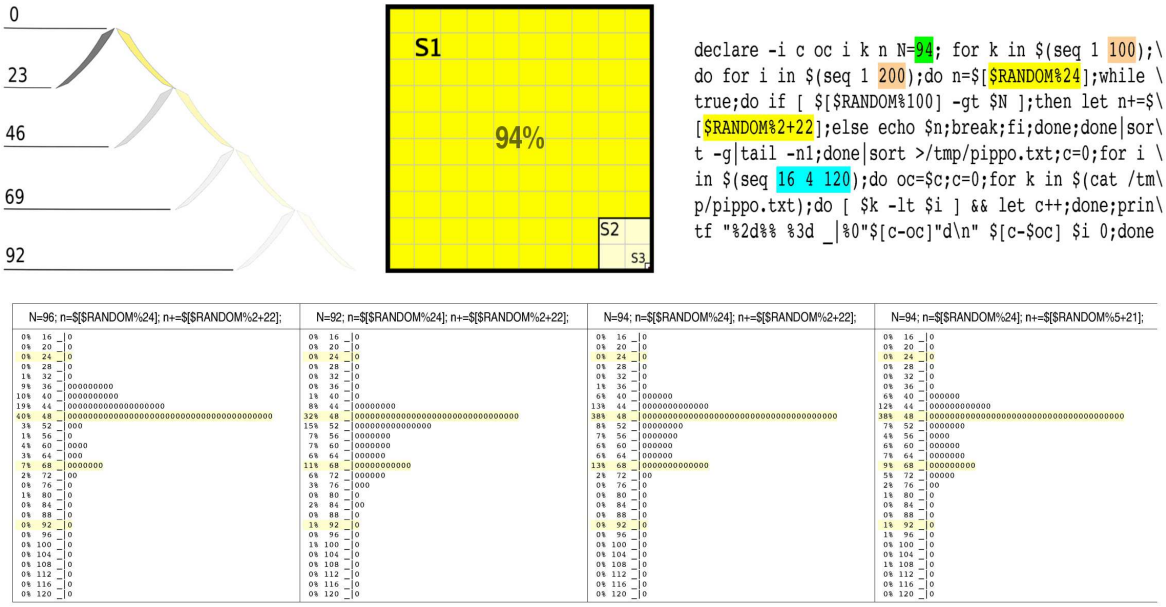
$$P_{S1} = (P_{nc})^{G=200} \leq \tfrac{1}{N=120} \Rightarrow P_{nc} \leq 97,634677\%$$

ci fornisce un maggiorante per la probabilità di non combinazione. Il fatto che il quinto termine della serie non si sia manifestato su $M = 23847$ misure, cioè

$$P_{S5} = (P_c)^4 \leq \tfrac{1}{M=23847} \Rightarrow P_{nc} = 1 - P_c \geq 92\%$$

probabilità di manifestarsi di ogni singolo termine gaussiano, si calcola come numero dei campioni che si reputa dovuti al termine diviso il numero di campioni complessivi, in questo caso, circa 66% e 33% Quest'ultima è una misura meno precisa della precedente per via della parziale sovrapposizione delle due distribuzioni ma molto più significativa qualora si voglia confrontare l'importanza relativa dei due contributi.

ci fornisce un minorante per la probabilità di non combinazione. Se all'interno di questo intervallo di probabilità si prende, per esempio, il valore $P_{nc} = 94\%$ allora la distribuzione di probabilità per i singoli termini, cioè per $G = 1$, sarebbe rappresentata dalle aree identificate dalla seguente binomiale:



il modello realizzato in bash script genera insiemi statistici che sono pilotati dal valore di probabilità (verde)
inoltre è in grado di simulare la raccolta dati per campione e la modalità di presentazione (arancio e azzurro)
infine permette di inserire dell'incertezza sul valore di inizio e su quello di incremento (giallo)

Le quattro simulazioni mostrano una progressione di condizioni che converge verso un modello tarato in maniera da produrre insiemi di misure simili ai risultati trovati sperimentalmente. Il fatto che il modello sia compatibile con le misure non prova l'ipotesi iniziale però fornisce una verifica empirica che l'ipotesi è in grado di spiegare un comportamento come quello misurato.

5.4.3 Ipotesi sulla causa delle misure osservate

L'unica critica al modello sopra proposto è stata fondamentalmente di tipo scettico: "*difficile credere che un sistema deterministico quale un software per elaboratore possa essere correttamente rappresentato da un modello probabilistico*". Non avendo tempo per una verifica sperimentale del modello proposto una spiegazione che è parsa ragionevole è che l'esistenza di un certo grado di caos sia dovuto alla gestione delle interruzioni che sono un evento asincrono (elemento caotico) rispetto allo stato di cache (elemento di non linearità).

Il processore Intel Celeron Mendocino[33] ha una cache di primo livello di 32 Kb per una frequenza di lavoro di 1.2 GHz e per una media di un istruzione a ciclo di clock si ottiene: $\frac{32 \cdot 1024}{1.2 \cdot 10^9} = 27.3$ uS che è un tempo ragionevolmente vicino a quello per il quale il modello mostrava una somiglianza dei risultati rilevante con le misure. Non stupirebbe se fatto un calcolo più accurato si scoprisse che in media si possono salvare in cache istruzioni per un tempo corrispondente a quello legato al primo termine della serie individuata nella sezione 5.4.1.

[33] Intel Mendocino (in fondo alla pagina) - `http://it.wikipedia.org/wiki/Mendocino_(Intel)`

5.4.4 Impatto dell'uso del TSC sullo jitter di attivazione

Ritornando alle misure di stabilità del periodo quanto visto nelle sezioni precedenti deve considerarsi un risultato altamente migliorabile come mostra il grafico seguente

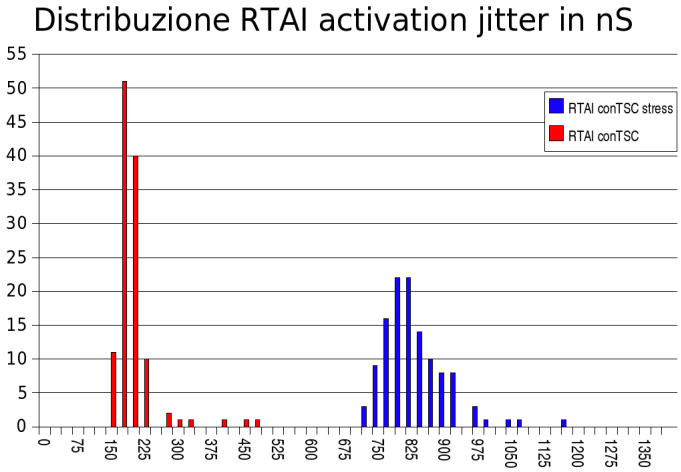
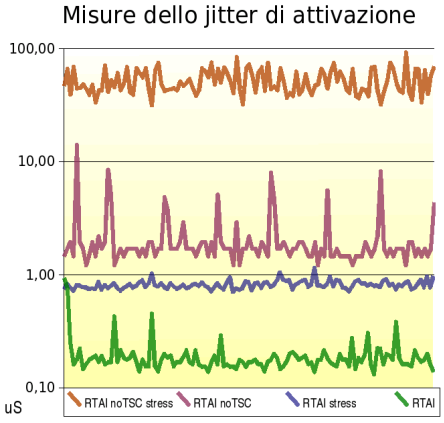
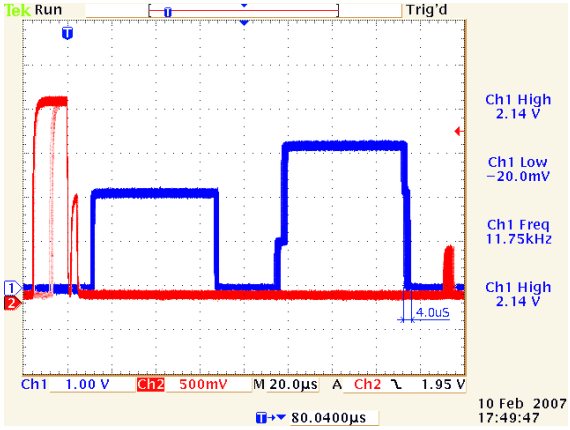


grafico ottenuto a partire da 119 campioni per un totale di 118.973 misure su PC-master con TSC configurato, ciclo aperiodico e busy-loop di precisione stato di stress ottenuto con il comando cat /dev/hda | od -x su ssh attiva

Vale la pena notare che, con l'uso del TSC, l'incertezza sul periodo è passata dal 10% allo 0.1% quindi è stata ridotta di ben due ordini di grandezza. Il comportamento del sistema sottoposto a stress non è qualitativamente diverso da quello rilassato salvo un fenomeno di dispersione che trasla e allarga la distribuzione di Poisson, a sinistra, facendola assomigliare maggiormente a una gaussiana, a destra:





Se il modello proposto nella sezione 5.4.2 risultasse appropriato allora queste distribuzioni sarebbero intrinsecamente delle binomiali dove la possibilità di studiarle con modelli continui come la distribuzione di Poisson o di Gauss risiederebbe nell'assimetricità della probabilità fra eventi opposti $P_c = 6\%$ contro $1 - P_c = 94\%$, nell'aggiunta di una quantità più o meno rilevante di rumore bianco a livello di singolo evento dovuto allo stress a cui viene sottoposto il sistema sotto analisi e infine dalla modalità di analisi stessa che può rivelare o nascondere più o meno le cause fondamentali. Questi tre criteri sono stati inseriti tutti e evidenziati nel modello sopra citato.

5.4.5 Tabelle riassuntive delle misure di attivazione

Le tabelle con intestazione arancione si riferiscono a misure effettuate sul sistema configurato senza il supporto per il TSC e con ciclo periodico. Evidenziati in giallo gli indici interessanti mentre in arancione o in verde è stato evidenziato, per ogni singola tabella, il massimo jitter di attivazione che è il valore di confronto fra i vari casi.

1 Active periodico noTSC

	min uS	avg uS	max uS	jlrg nS	javr nS	sgm2/ /1024	rps Hz	err Hz	sigma nS
Minimo	1322	1333	1334	152	-1	2	749	0	45
Media	1332	1333	1335	2809	0	68	749	0	264
Moda	1333	1333	1334	198	0	2	749	0	45
Mediana	1333	1333	1334	893	0	3,5	749	0	60
Massimo	1333	1333	1345	23452	0	1228	750	0	1121
Somma	-	-	-	-	-	-	88419	0	-

2 Active stress periodico noTSC

	min uS	avg uS	max uS	jlrg nS	javr nS	sgm2/ /1024	rps Hz	err Hz	sigma nS
Minimo	1304	1334	1334	412	-2	11	748	0	106
Media	1332	1334	1336	3508	0	356	749	0	604
Moda	1333	1334	1335	2289	0	293	749	0	548
Mediana	1333	1334	1335	2510	0	303	749	0	557
Massimo	1333	1334	1363	59250	1	7234	750	0	2722
Somma	-	-	-	-	-	-	88412	0	-

3 Active aperiodico

	min uS	avg uS	max uS	jlrg nS	javr nS	sgm2/ /1024	rps Hz	err Hz	sigma nS
Minimo	1000	1000	1000	130	0	1	996	0	32
Media	1000	1000	1000	193	0	2	999	0	39
Moda	1000	1000	1000	168	0	1	999	0	32
Mediana	1000	1000	1000	176	0	1	999	0	32
Massimo	1000	1000	1001	969	0	7	1000	0	85
Somma	-	-	-	-	-	-	115860	0	-

4 Active stress aperiodico

	min uS	avg uS	max uS	jlrg nS	javr nS	sgm2/ /1024	rps Hz	err Hz	sigma nS
Minimo	1000	1000	1001	717	0	25	989	0	160
Media	1000	1000	1001	875	0	37	995	0	195
Moda	1000	1000	1001	923	0	37	994	0	195
Mediana	1000	1000	1001	863	0	37	994	0	195
Massimo	1000	1000	1001	1182	0	49	1000	0	224
Somma	-	-	-	-	-	-	115376	0	-

5 Active stress nodma conirq aperiodico

	min uS	avg uS	max uS	jlrg nS	javr nS	sgm2/ /1024	rps Hz	err Hz	sigma nS
Minimo	1000	1000	1000	298	0	4	737	0	64
Media	1000	1000	1001	512	0	6	767	0	80
Moda	1000	1000	1001	511	0	6	769	0	78
Mediana	1000	1000	1001	491,5	0	6	769	0	78
Massimo	1000	1000	1001	1099	0	14	774	0	120
Somma	-	-	-	-	-	-	89009	0	-

6 Active stress nodma aperiodico

	min uS	avg uS	max uS	jlrg nS	javr nS	sgm2/ /1024	rps Hz	err Hz	sigma nS
Minimo	1000	1000	1000	252	0	3	737	0	55
Media	1000	1000	1000	389	0	4	767	0	63
Moda	1000	1000	1000	328	0	4	770	0	64
Mediana	1000	1000	1000	351	0	4	770	0	64
Massimo	1000	1000	1001	885	0	15	777	0	124
Somma	-	-	-	-	-	-	88998	0	-

7 Active stress conirq aperiodico

	min uS	avg uS	max uS	jlrg nS	javr nS	sgm2/ /1024	rps Hz	err Hz	sigma nS
Minimo	1000	1000	1001	671	0	22	989	0	150
Media	1000	1000	1001	813	0	31	995	0	178
Moda	1000	1000	1001	763	0	30	993	0	175
Mediana	1000	1000	1001	801	0	30	994	0	175
Massimo	1000	1000	1001	1152	1	41	1001	0	205
Somma	-	-	-	-	-	-	113413	0	-

Di seguito una leggenda esplicativa delle parole chiave usate per identificare i vari casi nelle tabelle sopra riportate:

periodico Vi sono fondamentalmente due modalità differenti per generare un evento periodicamente. Una di queste in RTAI è indicata come *periodic_mode* ed è utilizzata per l'attivazione periodica di un thread. Il thread è strutturato in maniera da bloccarsi sull'attesa di un evento mentre il motore di RTAI si occupa della generazione periodica di questo evento. Tale sospensione non è bloccante per il resto del sistema, cioè ne l'attesa e neppure la generazione sono realizzate mediante alcun tipo di polling. Perciò la CPU viene ceduta ad altri processi ma questo comporta il rischio che il risveglio avvenga in ritardo rispetto all'istante previsto. La variazione del periodo è il doppio del massimo ritardo con cui il thread percepisce l'evento di sblocco (cfr. sez. 5.1 in particolare l'ultima disequazione).

aperiodico L'uso di questo termine si riferisce alla seconda modalità con cui si può creare un evento periodico che in RTAI è indicata con *one shot mode*. Il thread in questo caso, eseguito il suo compito, si pone in uno stato di sospensione per un certa quantità di tempo. Ovviamente anche questa sospensione non è bloccante per il resto del sistema ma il tempo di sospensione può essere ricalcolato in maniera da compensare, almeno in parte, le precedenti fluttuazioni. Inoltre è possibile anticipare lo sblocco del thread in modo da prevenire eventuali ritardi e introdurre

un ciclo di attesa attivo, altrove definito come *busy-loop*, in maniera da compensare eventuali anticipi. Questa modalità permette di contenere efficacemente le variazioni del periodo.

stress Il PC-master è sottoposto a un carico di sistema che impatta sui tre sottosistemi principali: processore, disco e rete. In questo caso per rete si intende quella di supervisione. Questa infatti non ha nulla a che fare con la rete real-time di cui lo stato di stress, al fine di verificarne la congestione, è stato valutato solo nella fase preliminare delle misure (cfr. sez. 3.3.3). Generare una grande quantità di traffico sulla rete di supervisione equivale a generare molte richieste di gestione di interrupt da parte della relativa scheda di rete.

nodma Se questa parola chiave si riferisce al supporto DMA (Direct Memory Access) del disco rigido. In particolare significa che esso è stato disattivato. In questo caso il carico su disco ha una diversa personalità: si presenta infatti come una generazione intensa di interrupt piuttosto che come un inteso flusso di dati soggetti a elaborazione. Il manuale di RTAI consiglia di valutare la disabilitazione del DMA per ridurre una possibile fonte di grandi latenze.

conirq Se questa parola chiave compare allora il *busy-loop* si intende non protetto da una preventiva disattivazione di tutte gli interrupt del sistema.

noTSC Le tabelle contrassegnate da questa parola chiave si riferiscono alle misure effettuare sul sistema privo del supporto TSC (crf. sez. 4.1 e successive).

Il confronto fra le prime due tabelle e le altre elimina ogni dubbio sull'importanza del TSC per quel che riguarda la precisione temporale del periodo. Questa questione è stata ampiamente discussa nella sezione 4.1 e quindi non è il caso di dilungarsi ulteriormente.

Il confronto fra la tabella 3 e la 4 mostra che anche in condizioni di stress il sistema di compensazione del *busy-loop* contiene efficacemente lo jitter massimo si noti però che la varianza cresce notevolmente.

Il confronto fra la tabella 4 e la 7 farebbe pensare che proteggere il *busy-loop* disabilitando gli interrupt non produca risultati apprezzabili sul fronte del contenimento dello jitter massimo (-2.5%) e nemmeno la varianza pare essere stata ridotta sensibilmente. Invece il confronto fra la tabella 5 e la 6 mostra che tale affermazione è meno vera (-20%) quando si disattiva il DMA in quanto le richieste di interrupt si fanno molto più pressanti a causa della diversa gestione I/O del disco rigido. Infine il confronto fra la tabella 4 e la 6 mostra che il DMA è realmente una fonte di disturbo tale da aumentare lo jitter massimo in modo apprezzabile (+33%) e sulla varianza l'effetto è anche più evidente.

Come più volte detto l'oggetto di misura è lo jitter massimo ma in queste tabelle la varianza sulle misure fornisce un indicatore di affidabilità: tanto più è piccola tanto maggiore è la probabilità che la misura effettuata sia significativa di un buon contenimento dello jitter massimo.

Diritto d'autore

Licenza

La licenza in sintesi

Chiunque è libero di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire, recitare e modificare quest'opera alle seguenti condizioni:

- Deve attribuire la paternità dell'opera nei modi indicati dall'autore o da chi gli ha dato l'opera in licenza.
- Ogni volta che usa o distribuisce quest'opera, deve farlo secondo i termini di questa licenza, che va comunicata con chiarezza.
- In ogni caso, è possibile concordare col titolare dei diritti d'autore utilizzi di quest'opera non consentiti da questa licenza.

Riferimenti bibliografici

[1] http://www.phase.eu/techdocs/guidelines/tech-bul-ing1.pdf - Phase Motion Control, The Brushless Motor Revolution: Optimal Motor Selection Guidelines

[2] http://ar.linux.it/pub/srt-2006/zefeer.html - Pagina su come utilizzare Linux sulla Zefeer board

[3] http://www.atmel.com/dyn/products/product_card.asp?PN=ATmega88 - Atmel AVR ATMEGA-88 board

[4] http://www.acmesystems.it - Acme Systems, sito del produttore della FOX board

[5] http://kernel.org/ - Linux Kernel sito di riferimento per il download dei sorgenti

[6] http://lac.zkm.de/2006/papers/lac2006_lee_revell.pdf - Realtime Audio vs. Linux 2.6 di Lee Revell

[7] http://linuxdevices.com/articles/AT3479098230.html - Realtime benchmarks 2.4 vs 2.6 di Peter Laurich

[8] https://mail.gna.org/public/xenomai-core/2005-10/msg00085.html - Xenomai 2.4 vs 2.6 in embedded space

[9] http://www.denx.de/wiki/Know/Linux24vs26 - Comparing Linux 2.4 and Linux 2.6 Kernels

[10] https://www.rtai.org - RTAI the RealTime Application Interface for Linux

[11] http://www.xenomai.org - Xenomai a Real-Time framework for Linux

[12] http://www.xenomai.org/index.php/FAQs - Xenomai Frequently Asked Questions

[13] https://mail.gna.org/public/xenomai-help/2006-08/msg00115.html - Xenomai and RTAI comparison

[14] http://u-boot.sourceforge.net/ - U-BOOT: the universal boot loader

[15] https://www.rtai.org/index.php?module=documents&JAS_DocumentManager_op=viewDocument&JAS_Document_id=44 - RTAI User Manual

[16] http://www.rts.uni-hannover.de/rtnet/download/RTnet-ETFA05.pdf - RTnet: A Flexible Hard Real-Time Networking Framework

[17] http://www.linux.it/~rubini/docs/time/time.html - Misure di tempo in spazio kernel di Alessandro Rubini

[18] http://lwn.net/Articles/209101 - Time Stamp Counter - LWN article

[19] http://www.geocities.com/krishnapg/RDTSC.html - Intel's Time Stamp Counter assembler code example

[20] http://www.linuxdevices.com/articles/AT6298863382.html - History, Motivations, and Overview of RTAI

[21] http://www.isd.mel.nist.gov/projects/rtlinux/motor-jitter.pdf - Real-time Operating System Timing Jitter and its Impact on Motor Control

[22] http://it.wikipedia.org/wiki/Variabile_casuale_poissoniana - Variabile casuale poissoniana e la sua distribuzione

[23] http://creativecommons.org/licenses/by-nd/2.5/legalcode - Testo integrale della licenza Creative Commons adottata