Oggi parleremo del protocollo NTP, come descritto dall’RFC 5905 (la versione 4). È uno dei protocolli più vecchi e serve a sincronizzare gli orologi connessi alla rete.

Dal punto di vista hardware abbiamo all’interno dei nostri sistemi un dispositivo che funziona come clock. Questo clock contiene al suo interno un registro contatore, che viene incrementato a intervalli regolari grazie alla presenza di un segnale di clock periodico. Questo orologio funziona con un oscillatore al quarzo che dà un certo livello di precisione (specie perché è sempre alimentato, grazie a una piccola batteria al mercurio, in modo che l’orologio funzioni anche a macchina spenta).  
L’orologio che però viene consultato abitualmente è quello virtuale (quello hardware viene consultato solo durante la fase di bootstrap per inizializzare quello software), perché è più veloce accedervi. Questo orologio software è composto da celle di memoria RAM che contengono una struttura dati, al cui interno c’è un contatore. L’orologio software viene infatti aggiornato da dei timer programmabili, che lanciano delle richieste di interruzione alla CPU in modo da poter incrementare il contatore.

Tipicamente il timer potrà essere programmato per mandare una interrupt ogni millisecondo, in modo da non lanciarne troppe.

L’ora a livello software (per esempio in un sistema UNIX) viene rappresentata tramite una struttura dati che prevede l’uso di due contatori, che nella versione attuale sono da 32 bit. Il primo dei due contatori conta il numero di secondi e l’altro conta delle frazioni di secondo (in particolare i secondi vengono suddivisi in nanosecondi e secondi). Nel sistema UNIX il numero di secondi è visto come un intero con segno (con rappresentazione in complemento a 2) e per risalire alla data che si vuole si prende una data di riferimento chiamata Epoca (epoch): nei sistemi UNIX l’epoca è il 1/1/1970 a 00:00:00.

Il numero di bit usati fa sì che sia limitata la data maggiore rappresentabile (il valore dei secondi è destinato ad andare in overflow): nei sistemi UNIX la data maggiore rappresentabile è il 2037. Raggiunta quella data sarà necessario cambiare il modo di rappresentazione della data: la proposta migliore è di passare a una rappresentazione su 32 bit a una su 64 bit (inoltre così facendo si potrebbe ulteriormente aumentare la precisione/risoluzione del campo dei nanosecondi).

Nel frattempo, è stato sviluppato anche un altro standard, l’UTC, che corrisponde alle coordinate universali del tempo. La differenza rispetto alla versione di UNIX è che nello standard UTC i secondi sono rappresentati senza segno, ma l’epoca è spostata al 1/1/1900. Più o meno i due sistemi codificano nello stesso range di date.

L’idea generale del protocollo NTP è che quando un host si connette alla rete internet dovrebbe poter fare riferimento a una certa quantità di server NTP ai quali poter chiedere “che ore sono”. In tal modo si può sincronizzare l’orologio di una macchina locale con quello dei server di riferimento: se si era esaurita la batteria e si era persa l’indicazione corretta della data o se il clock è sfasato, accedendo alla rete si può usare il protocollo NTP per reimpostare l’ora esatta.

Le problematiche dietro la definizione di questo protocollo sono i ritardi nel ricevere una risposta da parte del server (quando ci arriva una risposta, l’ora indicata in essa è già passata). Dunque, anche se il server è molto preciso c’è comunque il rischio di ottenere dei risultati sbagliati. Inoltre, c’è il problema dei fusi orari.  
Per questo esiste la soluzione abbastanza semplice di far riferimento a un fuso orario in particolare: quello del meridiano di Greenwich, e di far aggiustare il fuso orario dall’host in seguito.

Infine, c’è il problema che i server NTP devono essere sincronizzati tra di loro: per fare ciò i server si scambiano informazioni sempre col protocollo NTP per sincronizzarsi. Dunque, il protocollo NTP deve poter funzionare anche in modalità Peer-To-Peer. Un ulteriore problema è che i server di riferimento dovrebbero avere un orologio interno più preciso di quelli normali. Gli orologi più precisi che si possono realizzare oggi sono gli orologi atomici, dunque, si usa come campione di tempo il tempo di decadimento degli ioni radioattivi. Poiché questi orologi atomici sono costosi (e pericolosi) tutte le nazioni hanno istituti dedicati per avere un orologio atomico di riferimento (qui in Italia ne abbiamo uno all’istituto Colonetti di Torino).

Altri posti in cui si trovano questi orologi atomici sono i satelliti (poiché è necessario per calcolare il minimo ritardo dei segnali gps).

Dal punto di vista del protocollo NTP, non c’è distinzione tra quale sia l’orologio atomico che sta usando per effettuare la sincronizzazione, basta che sia abbastanza preciso.

Col protocollo NTP è previsto che ci sia una gerarchia di server ripartita in 16 livelli, che corrispondono a diverse precisioni degli orologi utilizzati. Questi livelli vengono chiamati “strati”: il livello più preciso è lo strato 0, che tipicamente è un orologio atomico. Gli strati successivi comunicano con gli strati precedenti per avere il campione di tempo di riferimento (via via dunque c’è una riduzione della precisione).  
Lo strato 16, viene utilizzato per individuare le macchine che non si sono ancora connesse al protocollo NTP e quindi non si sono ancora connesse alla rete (->durante la connessione a internet, prima di essersi sincronizzati, gli host funzionano come strato 16 del protocollo NTP). L’orologio non è un computer, quindi il primo dispositivo (più preciso) ad avere il protocollo NTP è il computer allo strato 1 (che è direttamente collegato all’orologio atomico e il suo orologio software rimane sincronizzato con esso). -> lo strato 1 è il più preciso disponibile in rete. Un server di strato 2 è un server che è connesso a un server di strato 1. Analogamente uno di strato 3 aggiusta il proprio orologio interno con le informazioni che riceve da un server di strato 2 e così via.

Questa gerarchia serve a disturbare il meno possibile i server di strato 1 (si possono contattare anche lo strato 2, 3, 4 o 5… se non serve un elevatissima precisione): ciò limita notevolmente il traffico che deve essere gestito dai server di strato più alto.

Il protocollo NTP è un protocollo di livello applicativo, è quindi basato sul livello di trasporto UDP, perché evitando il three way handshake si può limitare al minimo la latenza. C’è la possibilità di perdita di messaggi, per compensare ciò, il protocollo prevede che sia possibile contattare più server contemporaneamente. Quando un server risponde (oppure no) si tiene conto delle risposte che sono / non sono arrivate. Ciclicamente si mandano richieste a tutti i server e poi si tiene conto di tutte le risposte e dopo un po’ di tempo si ritrasmettono le richieste a tutti i server (in maniera simile al protocollo di ritrasmissione). Dopo un po’ di conversazioni, si stabiliscono quali sono i server che rispondono di più e si fa una scrematura in cui si smette di contattare i server che rispondono di meno. Fatto ciò, bisogna verificare il tempo di round trip (RTT). I messaggi sono infatti realizzati con dei time stamp (che tengono l’ora come segnata dall’orologio); dalla parte del client sulla richiesta viene inserito un primo time stamp t1 (che potrebbe segnare un’ora sbagliata) e nel momento in cui la richiesta arriva al server, questo aggiunge un secondo time stamp t2. Quando il server manda la risposta, aggiunge un terzo istante di tempo t3 e se il client la riceve aggiunge un quarto time stamp t4.

Avendo ricevuto la risposta, il client ha adesso quattro time stamp. Usando queste informazioni il client può calcolare il round trip time del messaggio sulla rete, eliminando il tempo di attesa sul server:

(t4 – t1) – (t3 – t2)

A questo punto entra in gioco l’ipotesi che il tempo di andata dei messaggi sia uguale a quello di ritorno, se così fosse il delay sarebbe calcolabile dividendo per due il round trip time. In tal caso l’ora corrente sarebbe ottenibile facendo:

D = ((t4 – t1) – (t3 – t2)) / 2

Tempo attuale = t3 + D

Se la nostra rete fosse deterministica potremmo concludere il protocollo qui, ma sappiamo per esperienza che facendo più esperimenti si otterranno tanti delay diversi. Ecco, quindi, che torna in gioco la statistica; si fa una stima basata sulle risposte dei server che sono stati considerati “affidabili” dalla scrematura di prima. È quindi necessario calcolare valor medio e variabilità, tale variabilità parteciperà alla classificazione dei server: i server che hanno variabilità minore sono preferiti (il valore assoluto del delay non è così importante, perché quello si può stimare, ma una variabilità eccessiva è ciò che dà veramente fastidio).

Nella realtà si parte con una frequenza abbastanza alta di interrogazione dei server e, quando si inizia a rendersi conto della situazione ci si inizia a focalizzare solo sui server preferiti. Man mano che le misure vanno avanti si può rallentare/diradare le richieste (se normalmente si può partire con una richiesta al minuto, dopo un po’ si può iniziare a raddoppiare questo intervallo di tempo, anche perché migliora la precisione del proprio orologio).

I server della lista iniziale da dover chiamare possono essere di più strati. Dopo che avviene la scrematura e la sincronizzazione ha successo, l’host stesso diventa parte dello strato 1+lo strato del miglior server che si sta usando (se tra i server rimasti ce n’è uno di strato 2 l’host diventa un server di strato 3, se non ce n’è di strato 2 ma ce n’è uno di strato 3 allora l’host diventa di strato 4) -> teoricamente, l’host stesso potrebbe diventare un server contattabile per ottenere l’ora (ciò funziona particolarmente bene per le reti locali: si usa una macchina in particolare come server NTP e le altre macchine della LAN avranno un comodo server con una latenza bassa a portata di mano, basta inserirlo nella lista dei server da contattare).

Una caratteristica che vogliamo mantenere è di avere orologi che crescano monotonamente nel tempo (non è conveniente avere orologi che possono andare indietro nel tempo), per questo motivo se il mio orologio software segna una data nel futuro non posso farlo tornare indietro in un colpo solo: quello che posso fare, però, è rallentare i suoi incrementi fino a che non è tornato in pari con l’ora corrente (non si agisce sul valore, ma sulla frequenza di aggiornamento). Dunque se il nostro orologio è avanti di 5 minuti, si rallenta il suo aggiornamento in modo che nell’arco di qualche ora recuperi la propria situazione anomala.

Giusto per avere un’idea, in condizioni ottimali, si può arrivare a precisioni dell’ordine di 1 millisecondo utilizzando il protocollo NTP. Per condizioni ottimali si intende con la rete che non ha problemi di congestioni e quando si è riusciti a contattare server strato di livello abbastanza basso la cui variabilità nella latenza è bassa.  
In situazioni non tanto ottimali, potremmo arrivare a precisioni di 1 decimo di secondo (più o meno 100 millisecondi).  
Dunque, una macchina connessa alla rete avrà un orologio locale la cui precisione varierà tra un decimo e un millesimo di secondo.

Perché è così importante avere un’idea così precisa di che ore sono? Ci sono una serie di problematiche di sicurezza che possono essere risolte tramite certificati digitali, ad esempio se non abbiamo un’idea precisa di che ore sono un certificato non valido può risultare valido e dunque l’algoritmo criptografico diventa inaffidabile. Da questo punto di vista, la precisione che otteniamo con il protocollo NTP è più che sufficiente.

Ci sono casi in cui si vuole usare esplicitamente indicazioni di ora che non sono attuali: in tal caso bisogna staccarsi dal protocollo NTP, per esempio se vogliamo usare licenze software scadute. Un altro caso più serio è se ho il PC che pensa di essere al 2032 mi conviene prima aggiustare almeno la data e poi lasciare il protocollo NTP ad aggiornare l’ora ai secondi corretti.

La versione 4 del protocollo prevede di usare sia i time stamp a 32 bit che quelli a 64 bit (il protocollo è già pronto all’overflow della data). La versione a 64 bit dei time stamp in realtà è un upgrade: il contatore da 64 bit è in realtà formato dall’unione di due contatori da 32 bit, i primi 32 bit rappresentano la data in maniera classica, mentre i successivi 32 servono a indicare l’epoca (l’epoca 0 è quella attuale, l’epoca 1 sarà quella che parte dal 2030 circa l’epoca 2 partirà dal 2100 e qualcosa ecc.).