

$$\delta = \frac{(T_4 - T_1) + (T_3 - T_2)}{2}$$

Eight pairs of (θ, δ) values are buffered, finally taking the minimal value found for δ as the best estimation for the delay between the two servers, and subsequently the associated value θ as the most reliable estimation of the offset.

Applying NTP symmetrically should, in principle, also let B adjust its clock to that of A . However, if B 's clock is known to be more accurate, then such an adjustment would be foolish. To solve this problem, NTP divides servers into strata. A server with a **reference clock** such as a WWV receiver or an atomic clock, is known to be a **stratum-1 server** (the clock itself is said to operate at stratum 0). When A contacts B , it will only adjust its time if its own stratum level is higher than that of B . Moreover, after the synchronization, A 's stratum level will become one higher than that of B . In other words, if B is a stratum- k server, then A will become a stratum- $(k+1)$ server if its original stratum level was already larger than k . Due to the symmetry of NTP, if A 's stratum level was *lower* than that of B , B will adjust itself to A .

There are many important features about NTP, of which many relate to identifying and masking errors, but also security attacks. NTP is described in Mills (1992) and is known to achieve (worldwide) accuracy in the range of 1–50 msec. The newest version (NTPv4) was initially documented only by means of its implementation, but a detailed description can now be found in Mills (2006).

The Berkeley Algorithm

In many algorithms such as NTP, the time server is passive. Other machines periodically ask it for the time. All it does is respond to their queries. In Berkeley UNIX, exactly the opposite approach is taken (Gusella and Zatti, 1989). Here the time server (actually, a time daemon) is active, polling every machine from time to time to ask what time it is there. Based on the answers, it computes an average time and tells all the other machines to advance their clocks to the new time or slow their clocks down until some specified reduction has been achieved. This method is suitable for a system in which no machine has a WWV receiver. The time daemon's time must be set manually by the operator periodically. The method is illustrated in Fig. 6-7.

In Fig. 6-7(a), at 3:00, the time daemon tells the other machines its time and asks for theirs. In Fig. 6-7(b), they respond with how far ahead or behind the time daemon they are. Armed with these numbers, the time daemon computes the average and tells each machine how to adjust its clock [see Fig. 6-7(c)].

Note that for many purposes, it is sufficient that all machines agree on the same time. It is not essential that this time also agrees with the real time as announced on the radio every hour. If in our example of Fig. 6-7 the time daemon's clock would never be manually calibrated, no harm is done provided