of their first waiting intervals. At the end of its second waiting interval, p broadcasts a READY message indicating that it is ready to begin the next round. However, if p receives f + 1 READY messages during its second waiting interval, it terminates its second interval early, and goes ahead and broadcasts READY. As soon as p receives n – f READY messages, it updates the clock according to the adjustment calculated earlier, and begins its next round by broadcasting its new clock value. (This algorithm uses some ideas from [3].)

A process need only keep clock differences for one round at a time. The waiting intervals are designed so that during round i a nonfaulty process p will not receive a READY message from another nonfaulty process until p has finished collecting round i clock values. Round i + 1 clock values are not broadcast until after READY is broadcast, so p will certainly not receive round i + 1 clock values until after it has finished collecting round i clock values. However, round i + 1 clock values might arrive during the second waiting interval and while the process is collecting READY messages. As a result, the adjustment is calculated at the end of the first waiting interval and the difference for any round i + 1 clock value received during round i is decremented by the amount of the adjustment.

5.2.2 Code for an Arbitrary Process

Global constants: δ, ε, ρ, n, f: as usual.

Local variables (all initially arbitrary):

- T: clock time at which current round began.
- U: clock time at which the first waiting period is to end.
- V: clock time at which the second waiting period is to end.
- DIFF: array of clock differences between other processes and this one for current round.
- SENT-READY: set of processes from whom READY messages have been received in current round.
- CORR: correction variable.
- A: adjustment to clock.

The code is in Figure 5-1.

```
beginstep(w)
do forever  /* each iteration is a round */
     T := NOW
     broadcast(T)
     U := T + (1 + ρ)(2δ + 4ε)
     set-timer(U)

/* first waiting interval:  collect clock values */

     while ~(w = TIMER & NOW = U) do
          if w = (m,q) then  DIFF[q] := m + δ - NOW endif
          endstep
          beginstep(w)
          endwhile

/* end of first waiting interval */

     A := mid(reduce(DIFF))
     V := U + (1 + ρ)(4ε + 4ρ(δ + 2ε) + 2ρ²(δ + 2ε))
     set-timer(V)
     SENT-READY := ∅

/* second waiting interval:  collect READY messages and clock values
   for next round */

     while ~(w = TIMER & NOW = V) do
          if w = (READY,q) then
               SENT-READY := SENT-READY ∪ {q}
               if |SENT-READY| = f + 1 then exit endif
          elseif w = (m,q) then DIFF[q] := m + δ - NOW endif
          endstep
          beginstep(w)
          endwhile

/* end of second waiting interval due to timer or f + 1 READY messages */

     broadcast(READY)
     endstep
     beginstep(w)

/* collect n - f READY messages and next round clock values */

     while true do
          if w = (READY,q) then
               SENT-READY := SENT-READY ∪ {q}
               if |SENT-READY| = n - f then exit endif
          elseif w = (m,q) then DIFF[q] := m + δ - NOW endif
          endstep
          beginstep(w)
          endwhile

/* update clock and begin next round */

     DIFF := DIFF - A
     CORR := CORR + A
     endstep
     beginstep(w)
     enddo
```

Figure 5-1: Algorithm 5-1, Establishing Synchronization

5.3 Analysis

We will use the following notation in addition to that introduced already.

- $VAL^i_p(q)$ is the value of q's round i message to p.
- $DIFF^i_p(q) = VAL^i_p(q) + \delta - ARR^i_p(q)$, p's estimate of the difference between p's and q's clocks.
- $DIFF^i_p$ is the multiset of $DIFF^i_p(q)$ values.
- t^i_p is the real time when p begins round i.
- u^i_p is the real time when p begins the second waiting interval during round i.
- v^i_p is the real time when p sends READY during round i (and thus ends the second waiting interval).
- $arr^i_p(q)$ is the real time when p first receives a round i clock value from q.
- $rdy^i_p(q)$ is the real time when p first receives READY from q during round i.
- $tmax^i = \max\{t^i_p\}$ for p nonfaulty, the latest real time when a nonfaulty process begins round i.
- $B^i = \max\{|C^i_p(tmax^i) - C^i_q(tmax^i)|\}$ for p and q nonfaulty, the maximum difference between nonfaulty clock values at $tmax^i$.

Note that $tmax^i$ has a slightly different meaning from that in Chapter 4.

From now on, terms of order ρ^2 and higher will be ignored. Since 10^{-6} seconds is an often quoted reasonable value for ρ [5, 7, 9], terms of order ρ^2 are negligible. The second-order terms in the assignment to V in line 13 of the code are needed for strict correctness, but will not appear in the analysis.

Lemma 5-2 proves together inductively that the time between two nonfaulty processes beginning a round is bounded, and that when a nonfaulty process q receives READY from another nonfaulty process, q has already finished the first waiting period. First we show a preliminary lemma needed by Lemma 5-2.

Lemma 5-1: Let $i \geq 0$, p and q be any nonfaulty processes, and r be the first nonfaulty process to send READY at round i. Then

$$rdy^i_q(p) \geq u^i_q - (t^i_q - t^i_r) + \delta + 3\varepsilon.$$

Proof: Since r is the first nonfaulty process to send READY, it doesn't send until its full second waiting interval has elapsed. Then

$rdy^i_q(p) \geq v^i_p + \delta - \varepsilon$

$\geq v^i_r + \delta - \varepsilon$

$\geq t^i_r + (2\delta + 4\varepsilon) + (4\varepsilon + 4\rho(\delta + 2\varepsilon)) + \delta - \varepsilon$, by definition of v^i_r and the upper bound on the drift rate

$= t^i_r + 3\delta + 7\varepsilon + 4\rho\delta + 8\rho\varepsilon$,

and

$u^i_q \leq t^i_r + (t^i_q - t^i_r) + (u^i_q - t^i_q)$

$\leq t^i_r + (t^i_q - t^i_r) + (1 + \rho)^2(2\delta + 4\varepsilon)$, by definition of u^i_q and the lower bound on the drift rate

$= t^i_r + (t^i_q - t^i_r) + 2\delta + 4\varepsilon + 4\rho\delta + 8\rho\varepsilon$.

Thus, $t^i_r \geq u^i_q - (t^i_q - t^i_r) - 2\delta - 4\varepsilon - 4\rho\delta - 8\rho\varepsilon$, implying

$rdy^i_q(p) \geq u^i_q - (t^i_q - t^i_r) - 2\delta - 4\varepsilon - 4\rho\delta - 8\rho\varepsilon + 3\delta + 7\varepsilon + 4\rho\delta + 8\rho\varepsilon$

$= u^i_q - (t^i_q - t^i_r) + \delta + 3\varepsilon$. ∎

Lemma 5-2: For any nonfaulty processes p and q and any $i \geq 0$,

(a) $|t^i_p - t^i_q| \leq \delta + 3\varepsilon$, and

(b) $rdy^i_q(p) \geq u^i_q$.

Proof: We proceed by induction on i.

Basis: i = 0.

(a) $|t^0_p - t^0_q| \leq \delta + \varepsilon$, because as soon as p wakes up, it sends its round 0 message to all other processes. The receipt of this message, which occurs at most $\delta + \varepsilon$ later, causes q to begin round 0, if it hasn't already done so.

(b) Let r be the first nonfaulty process to send READY at round 0. By Lemma 5-1,

$rdy^0_q(p) \geq u^0_q - (t^0_q - t^0_r) + \delta + 3\varepsilon$

$\geq u^0_q - (\delta + \varepsilon) + \delta + 3\varepsilon$, by part (a)

$> u^0_q$.

Induction: Assume for i – 1 and show for i.

(a) Let s be the first nonfaulty process to begin round i. Then s receives n – f READY messages during its round i – 1 (after u^{i-1}_s). At least n – 2f of them are from nonfaulty processes by part (b) of the induction hypothesis. These n – 2f nonfaulty processes

also send READY messages to all the other processes. By $t^i_s + 2\varepsilon$, every nonfaulty process receives at least $n - 2f \geq f + 1$ READY messages and broadcasts READY. Thus q receives $n - f$ READY messages by $t^i_s + 2\varepsilon + \delta + \varepsilon$. Thus,

$t^i_q \leq t^i_s + \delta + 3\varepsilon$

$\leq t^i_p + \delta + 3\varepsilon$, by choice of s,

which implies $t^i_q - t^i_p \leq \delta + 3\varepsilon$.

By reversing the roles of p and q in the above argument, we obtain $t^i_p - t^i_q \leq \delta + 3\varepsilon$.

(b) Let r be the first nonfaulty process to send READY at round i. By Lemma 5-1,

$rdy^i_q(p) \geq u^i_q - (t^i_q - t^i_r) + \delta + 3\varepsilon$

$\geq u^i_q - (\delta + 3\varepsilon) + \delta + 3\varepsilon$, by part (a)

$= u^i_q$. ∎

Next we show that a process waits a sufficient length of time to receive clock values from all nonfaulty processes before beginning the second waiting interval in a round.

Lemma 5-3: Let p and q be nonfaulty, and $i \geq 0$. Then $arr^i_p(q) \leq u^i_p$.

Proof: By the lower bound on the drift rate, $u^i_p \geq t^i_p + 2\delta + 4\varepsilon$. Lemma 5-2 implies that q sends its round i clock value by $t^i_p + \delta + 3\varepsilon$. Thus $arr^i_p(q) \leq t^i_p + 2\delta + 4\varepsilon \leq u^i_p$. ∎

The next two lemmas bound how long a round can last for one process. First we bound how long a process must wait after sending READY to receive $n - f$ READY messages.

Lemma 5-4: For p nonfaulty and $i \geq 0$, $t^{i+1}_p - v^i_p \leq 2\delta + 4\varepsilon + 4\rho(\delta + 4\varepsilon)$.

Proof: The worst case occurs if p is as far ahead of the other nonfaulty processes as possible, its clock is fast, the other clocks are slow, and the slow processes' READY messages take as long as possible to arrive. However, as soon as they arrive, p begins the next round. Let q be one of the slow nonfaulty processes.

$$t^{i+1}_p - v^i_p = (t^{i+1}_p - v^i_q) + (v^i_q - u^i_q) + (u^i_q - t^i_q) + (t^i_q - t^i_p) - (v^i_p - u^i_p) - (u^i_p - t^i_p)$$

$$\leq (\delta + \varepsilon) + (1 + \rho)^2(4\varepsilon + 4\rho(\delta + 2\varepsilon)) + (1 + \rho)^2(2\delta + 4\varepsilon) + (\delta + 3\varepsilon) - (4\varepsilon + 4\rho(\delta + 2\varepsilon)) - (2\delta + 4\varepsilon)$$

$= 2\delta + 4\varepsilon + 4\rho(\delta + 4\varepsilon)$, ignoring ρ^2 terms. ∎

Lemma 5-5: For any nonfaulty process p and any $i \geq 0$,

$t^{i+1}_p - t^i_p \leq 4\delta + 12\varepsilon + 4\rho(3\delta + 10\varepsilon)$.

Proof: $t^{i+1}_p - t^i_p = (t^{i+1}_p - v^i_p) + (v^i_p - u^i_p) + (u^i_p - t^i_p)$

$\leq 2\delta + 4\varepsilon + 4\rho(\delta + 4\varepsilon) + (v^i_p - u^i_p) + (u^i_p - t^i_p)$, by Lemma 5-4

$\leq 2\delta + 4\varepsilon + 4\rho(\delta + 4\varepsilon) + (1 + \rho)^2(4\varepsilon + 4\rho(\delta + 2\varepsilon)) + (1 + \rho)^2(2\delta + 4\varepsilon)$

$= 4\delta + 12\varepsilon + 4\rho(3\delta + 10\varepsilon)$. ∎

Now we give an upper bound on how far apart $tmax^i$ and $tmax^{i+1}$ can be.

Lemma 5-6: For any $i \geq 0$,

$tmax^{i+1} - tmax^i \leq 4\delta + 12\varepsilon + 4\rho(3\delta + 10\varepsilon)$.

Proof: Let p be the nonfaulty process such that $t^{i+1}_p = tmax^{i+1}$. Then

$tmax^{i+1} - tmax^i = t^{i+1}_p - tmax^i \leq t^{i+1}_p - t^i_p$

$\leq 4\delta + 12\varepsilon + 4\rho(3\delta + 10\varepsilon)$, by Lemma 5-5. ∎

Lemma 5-7 bounds the amount of real time between the time a nonfaulty process receives a round i message from another nonfaulty process and the time the last nonfaulty process begins round $i + 1$.

Lemma 5-7: For any $i \geq 0$ and nonfaulty processes p and q,

$tmax^{i+1} - arr^i_p(q) \leq 5\delta + 19\varepsilon + 4\rho(3\delta + 10\varepsilon)$.

Proof: $tmax^{i+1} - arr^i_p(q) = (tmax^{i+1} - t^{i+1}_p) + (t^{i+1}_p - t^i_p) + (t^i_p - t^i_q) - (arr^i_p(q) - t^i_q)$

$\leq (\delta + 3\varepsilon) + (4\delta + 12\varepsilon + 4\rho(3\delta + 10\varepsilon)) + (\delta + 3\varepsilon) - (\delta - \varepsilon)$, by Lemmas 5-2 and 5-5 and the lower bound on the message delay

$= 5\delta + 19\varepsilon + 4\rho(3\delta + 10\varepsilon)$. ∎

The next lemma bounds the error in a nonfaulty process' estimate of another nonfaulty process' local time at a particular real time.

Lemma 5-8: Let p and r be nonfaulty. Then

$|DIFF^i_p(r) + C^i_p(tmax^{i+1}) - C^i_r(tmax^{i+1})| \leq \varepsilon + \rho(11\delta + 39\varepsilon)$.

Proof: $|DIFF^i_p(r) + C^i_p(tmax^{i+1}) - C^i_r(tmax^{i+1})|$

$= |VAL^i_p(r) + \delta - ARR^i_p(r) + C^i_p(tmax^{i+1}) - C^i_r(tmax^{i+1})|$.

If the quantity in the absolute value signs is negative, then this expression is equal to

$C^i_r(tmax^{i+1}) - C^i_p(tmax^{i+1}) + C^i_p(arr^i_p(r)) - \delta - VAL^i_p(r)$

$\leq C^i_r(tmax^{i+1}) - C^i_p(tmax^{i+1}) + C^i_p(arr^i_p(r)) - \delta - C^i_r(arr^i_p(r) - \delta - \varepsilon)$, since the delay is at most $\delta + \varepsilon$

$\leq C^i_r(tmax^{i+1}) - C^i_p(tmax^{i+1}) + C^i_p(arr^i_p(r)) - \delta - C^i_r(arr^i_p(r)) + (1 + \rho)(\delta + \varepsilon)$, since the clock drift is at most $1 + \rho$

$= (C^i_r(tmax^{i+1}) - C^i_p(tmax^{i+1})) - (C^i_r(arr^i_p(r)) - C^i_p(arr^i_p(r))) - \delta + \delta + \varepsilon + \rho\delta + \rho\varepsilon$

$\leq 2\rho(tmax^{i+1} - arr^i_p(r)) + \varepsilon + \rho\delta + \rho\varepsilon$, by Lemma 4-2

$\leq 2\rho(5\delta + 19\varepsilon) + \varepsilon + \rho\delta + \rho\varepsilon$, by Lemma 5-7

$= \varepsilon + \rho(11\delta + 39\varepsilon)$.

If the quantity in the absolute value signs is positive, a similar argument shows that $|DIFF^i_p(r) + C^i_p(tmax^{i+1}) - C^i_r(tmax^{i+1})| \leq \varepsilon + \rho(11\delta + 37\varepsilon)$. ∎

The next lemma bounds how far apart two processes' i-th clocks are at the time when the last process begins round i + 1. The bound is in terms of how far apart the clocks are when the last process begins round i.

Lemma 5-9: For any nonfaulty p and q, and any i,

$|C^i_p(tmax^{i+1}) - C^i_q(tmax^{i+1})| \leq B^i + 8\rho(\delta + 3\varepsilon)$.

Proof: $|C^i_p(tmax^{i+1}) - C^i_q(tmax^{i+1})|$

$\leq |C^i_p(tmax^i) - C^i_q(tmax^i)| + |(C^i_q(tmax^{i+1}) - C^i_q(tmax^{i+1})) - (C^i_p(tmax^i) - C^i_q(tmax^i))|$

$\leq B^i + 2\rho(tmax^{i+1} - tmax^i)$, by definition of B^i and Lemma 4-2

$\leq B^i + 2\rho(4\delta + 12\varepsilon)$, by Lemma 5-6 and ignoring ρ^2 terms

$= B^i + 8\rho(\delta + 3\varepsilon)$. ∎

Now we can state the main result, bounding B^{i+1} in terms of B^i.

Theorem 5-10: $B^{i+1} \leq \frac{1}{2}B^i + 2\varepsilon + 2\rho(11\delta + 39\varepsilon)$.

Proof: $B^{i+1} = \max\{|C^{i+1}_p(tmax^{i+1}) - C^{i+1}_q(tmax^{i+1})|\}$ for nonfaulty p and q.

Let $x = \varepsilon + \rho(11\delta + 39\varepsilon)$.

We now define three multisets U, V, and W that satisfy the hypotheses of Lemma A-4. Let

$U = DIFF^i_p + C^i_p(tmax^{i+1})$,

$V = DIFF^i_q + C^i_q(tmax^{i+1})$, and

$W = \{C^i_r(tmax^{i+1})$: r is nonfaulty$\}$.

U and V have size n; W has size n – f.

Define an injection from W to U as follows. Map each element C^i_r in W to $DIFF^i_p(r) + C^i_p(tmax^{i+1})$ in U. Since Lemma 5-8 implies that

$|DIFF^i_p(r) + C^i_p(tmax^{i+1}) - C^i_r(tmax^{i+1})| \leq x$

for all the n – f nonfaulty processes, $d_x(W,U) = 0$. Similarly, $d_x(W,V) = 0$.

By Lemma 5-9, $diam(W) \leq B^i + 8\rho(\delta + 3\varepsilon)$. Thus, Lemma A-4 implies

$|mid(reduce(U)) - mid(reduce(V))| \leq \frac{1}{2}diam(W) + 2x$

$= \frac{1}{2}B^i + 2\varepsilon + 2\rho(11\delta + 39\varepsilon)$.

Since $mid(reduce(U)) = mid(reduce(DIFF^i_p + C^i_p(tmax^{i+1})))$

$= mid(reduce(DIFF^i_p)) + C^i_p(tmax^{i+1})$

$= ADJ^i_p + C^i_p(tmax^{i+1})$

$= C^{i+1}_p(tmax^{i+1})$

and similarly $mid(reduce(V)) = C^{i+1}_q(tmax^{i+1})$, the result follows. ∎

We obtain an approximate bound on how closely this algorithm will synchronize the clocks by considering the limit of B^i as the round number increases without bound.

Theorem 5-11: This algorithm can synchronize clocks to within $4\varepsilon + 4\rho(11\delta + 39\varepsilon)$.

Proof: $\lim_{i\to\infty} B^i$

$= \lim_{i\to\infty}[B^0/2^i + (1 + 1/2 + \ldots + 1/2^{i-1})(2\varepsilon + 2\rho(11\delta + 39\varepsilon))]$

$= 4\varepsilon + 4\rho(11\delta + 39\varepsilon)$, since the limit of the geometric series is 2. ∎

As was the case for Algorithm 4-1, if the number of processes, n, increases while f, the number of faulty processes remained fixed, a greater closeness of synchronization can be achieved by modifying Algorithm 5-1 so that it computes the mean instead of the midpoint of the range of values. which approaches $2\varepsilon + 2\rho P$ as n approaches infinity.

After modifying Algorithm 5-1, we get

$B^i \leq B^{i-1}f/(n-2f) + 2\varepsilon + 2\rho(11\delta + 39\varepsilon)$.

This is the same as

$B^i \leq B^0 f/(n-2f) + (1 - (f/(n-2f))^i)/(1 - f/(n-2f))(2\varepsilon + 2\rho(11\delta + 39\varepsilon))$,

which approaches $2\varepsilon + 2\rho(11\delta + 39\varepsilon)$ as n approaches infinity.

5.4 Determining the Number of Rounds

The nonfaulty processes must determine how many rounds of this algorithm must be run to establish the desired degree of synchronization before switching to the maintenance algorithm. The basic idea is for each nonfaulty process p to estimate B^0, and then calculate a sufficient number of rounds, $NROUNDS_p$, using the known rate of convergence. B^0 is estimated by having p calculate an overestimate and an underestimate for $C^0_q(tmax^0)$ for each q, and letting the estimated B^0 be the difference between the maximum overestimate and the minimum underestimate.

Let p's overestimate for $C^0_q(tmax^0)$ be $OVER_p(q)$ and p's underestimate for $C^0_q(tmax^0)$ be $UNDER_p(q)$.

For the overestimate, we assume that q's clock is fast, and that the maximum amount of time elapses between t^0_q (when q sent the message) and $tmax^0$. That maximum is $\delta + \varepsilon$ since every nonfaulty process begins round 0 as soon as it receives a message. Thus,

$$OVER_p(q) = VAL^0_p(q) + (1 + \rho)(\delta + \varepsilon).$$

Similarly, we can derive the underestimate. We assume that q is the last nonfaulty process to begin round 0. Thus,

$$UNDER_p(q) = VAL^0_p(q).$$

Process p computes its estimate of B^0,

$$B^0_p = \max_q\{OVER_p(q)\} - \min_q\{UNDER_p(q)\}.$$

Now p estimates how many rounds are needed until the spread is close enough. There is a predetermined $\gamma \geq 4\varepsilon + 4\rho(11\delta + 39\varepsilon)$, which is the desired closeness of synchronization for the start-up algorithm. After j rounds,

$$B^j \leq B^0_p/2^j + (1 + 1/2 + \ldots + 1/2^{j-1})(2\varepsilon + 2\rho(11\delta + 39\varepsilon)).$$

Process p sets the right hand side equal to γ and solves for j to obtain its estimate of the required number of rounds, $NROUNDS_p$.

Now each process executes a Byzantine Agreement protocol on the vector of NROUNDS values, one value for each process. The processes are guaranteed to have the same vector at the end of the Byzantine Agreement protocol. Each process chooses the (f + 1)-st smallest element of the resulting vector as the required number of rounds. The smallest number of rounds computed by a nonfaulty process will suffice to achieve the desired closeness of synchronization. Variations in the number of rounds computed by different nonfaulty processes are due to spurious values introduced by faulty processes and to different message delays. However, the range computed by any nonfaulty process is guaranteed to include the actual values of all nonfaulty processes at $tmax^0$, so the range determined by the process that computes the smallest number of rounds also includes all the actual values. In order to guarantee that each process chooses a number of rounds that is at least as large as the smallest one computed by a nonfaulty process, it chooses the (f + 1)-st smallest element of the vector of values.

Any Byzantine Agreement protocol requires at least f + 1 rounds. The processes can execute this algorithm in parallel with the clock synchronization algorithm, beginning at round 0. The clock synchronization algorithm imposes a round structure on the processes' communications. The Byzantine Agreement algorithm can be executed using this round structure. Each BA message can also include information needed for the clock synchronization algorithm (namely, the current clock value). However, the processes will always need to do at least f + 2 rounds, one to obtain the estimated number of rounds and f + 1 for the Byzantine Agreement algorithm.

5.5 Switching to the Maintenance Algorithm

After the processes have done the required number of rounds (denoted by r throughout this section) of the start-up algorithm, they cease executing it. The processes should begin the maintenance algorithm as soon as possible after ending the start-up algorithm in order to minimize the inaccuracy introduced by the clock drift.

In the maintenance algorithm each process broadcasts its clock value when its clock reaches T^i, for i = 0, 1, ..., where $T^{i+1} = T^i + P$. Let T^0 be a multiple of P. It is shown below in Lemma 5-13 that the first multiple of P reached by nonfaulty p's clock after finishing the required r rounds differs by at most one from the first multiple reached by nonfaulty q's clock after the r rounds. When a process reaches the first multiple of P after it has ended the start-up algorithm, it broadcasts its clock value as in the maintenance algorithm, but doesn't update its clock. At the next multiple of P, the process begins the full maintenance algorithm by broadcasting its clock