value and updating its clock. (It will receive clock values from all nonfaulty processes.)

The analysis introduces a new quantity, β_1, representing an upper bound on the closeness of the nonfaulty processes' clocks at $tmax^r$. That is, for any nonfaulty processes p and q, $|C^r_p(tmax^r) - C^r_q(tmax^r)| \leq \beta_1$. We show that if the following five inequalities are satisfied by the parameters, then the switch from the start-up algorithm to the maintenance algorithm (with parameter β) can be accomplished.

(1) $\beta_1 > 4\varepsilon + 4\rho(11\delta + 39\varepsilon)$

(2) $\beta \geq (\beta_1 + 2\varepsilon + \rho(6P - \beta_1 + 2\delta + 12\varepsilon)) / (1 - 8\rho)$

(3) $P > 2(1 + \rho)(\beta + \varepsilon) + (1 + \rho)\max\{\delta, \beta + \varepsilon\} + \rho\delta$

(4) $P \leq \beta/4\rho - \varepsilon/\rho - \rho(\beta + \delta + \varepsilon) - 2\beta - \delta - 2\varepsilon$

(5) $\beta \geq 4\varepsilon + 4\rho(3\beta + \delta + 3\varepsilon) + 8\rho^2(\beta + \delta + \varepsilon)$

The first inequality is imposed by the limitation on how closely the start-up algorithm can synchronize. The second inequality reflects the inaccuracy introduced during the switch. The last three are simply repeated from Section 4.5.1.

First we show that β_1 can be attained by the start-up algorithm.

Lemma 5-12: There exists an integer i such that $B^i \leq \beta_1$.

Proof: Since β_1 must be larger than $4\varepsilon + 4\rho(11\delta + 39\varepsilon)$, the result follows from Theorem 5-11, which states that the closeness of synchronization approaches $4\varepsilon + 4\rho(11\delta + 39\varepsilon)$ as the round number, i, increases. ∎

Note that the number of rounds, r, that the processes agree on is $\geq$ i, and that the worst-case B^r is no more than the worst-case B^i, which is at most β_1.

Lemma 5-13 shows that the first multiple of P reached by a nonfaulty process after finishing the start-up algorithm differs by at most one from that reached by another nonfaulty process.

Lemma 5-13: Let p and q be nonfaulty processes. Then

$|C^r_q(t^r_q) - C^r_p(t^r_p)| \leq P$.

Proof: $|C^r_q(t^r_q) - C^r_p(t^r_p)| \leq |C^r_q(t^r_p) + (1 + \rho)(t^r_q - t^r_p) - C^r_p(t^r_p)|$

$\leq |C^r_q(t^r_p) - C^r_p(t^r_p)| + (1 + \rho)(\delta + 3\varepsilon)$, by Lemma 5-2

$\leq |(C^r_q(t^r_p) - C^r_q(tmax^r)) - (C^r_p(t^r_p) - C^r_p(tmax^r))| + |C^r_q(tmax^r) - C^r_p(tmax^r)|$

$+ (1 + \rho)(\delta + 3\varepsilon)$

$\leq 2\rho(\text{tmax}^r - t^r_p) + \beta_1 + (1 + \rho)(\delta + 3\varepsilon)$, by Lemma 4-2 and definition of β_1

$\leq 2\rho(\delta + 3\varepsilon) + \beta_1 + (1 + \rho)(\delta + 3\varepsilon)$, by Lemma 5-2

$= \beta_1 + (1 + 3\rho)(\delta + 3\varepsilon)$.

Suppose in contradiction that $P < \beta_1 + (1 + 3\rho)(\delta + 3\varepsilon)$. By solving inequality (2) for β_1, we get

$\beta_1 \leq (\beta - 2\varepsilon - \rho(8\beta + 2\delta + 12\varepsilon + 6P)) / (1 - \rho)$,

which implies that

$P < (\beta - 2\varepsilon - \rho(8\beta + 2\delta + 12\varepsilon + 6P)) / (1 - \rho) + (1 + 3\rho)(\delta + 3\varepsilon)$.

This simplifies to $P < (\beta + \delta + \varepsilon - 8\rho\beta + \rho\delta - 3\rho\varepsilon) / (1 + 5\rho)$.

Combining this with inequality (3) yields

$2(1 + \rho)(\beta + \varepsilon) + (1 + \rho)\delta + \rho\delta < P < (\beta + \delta + \varepsilon - 8\rho\beta + \rho\delta - 3\rho\varepsilon) / (1 + 5\rho)$.

Solving for β gives $\beta < -(\varepsilon + 6\rho\delta + 15\rho\varepsilon) / (1 + 20\rho)$, which is a contradiction. ∎

The rest of the section is devoted to showing that the difference in real times when nonfaulty processes' clocks reach the first multiple of P at which they will all perform the maintenance algorithm is less than or equal to β. Consequently, this β can be preserved by the maintenance algorithm.

Define kP to be the first multiple of P reached by any nonfaulty process' r-th clock. The first multiple of P reached by any other nonfaulty process is either kP or (k + 1)P, by Lemma 5-13. At (k + 1)P some of the nonfaulty processes will actually update their clocks, and at (k + 2)P all of them will update their clocks.

Recall that $(k + 1)P = T^{k+1}$ and $U^{k+1} = T^{k+1} + (1 + \rho)(\beta + \delta + \varepsilon)$. Let $u^{k+1}_p = c^r_p(U^{k+1})$ and similarly for q.

Let s and t be two nonfaulty processes. Here is a description of the worst case:

- s has the smallest clock value at tmax^r, barely above (k–1)P, and its clock is slow.
- t's clock is fast and is β_1 ahead of s's at tmax^r.
- s updates its clock at U^{k+1}, by decrementing it as much as possible.

- t updates its clock at U^{k+1}, by incrementing it as much as possible.

First we must bound how far apart in real time nonfaulty processes' r-th clocks reach U^{k+1}.

Lemma 5-14: Let p and q be nonfaulty processes. Then

$|c^r_p(U^{k+1}) - c^r_q(U^{k+1})| \leq (1-\rho)\beta_1 + 2\rho(2P + \beta + \delta + \varepsilon)$.

Proof: Without loss of generality, suppose $c^r_p(U^{k+1}) \geq c^r_q(U^{k+1})$. Then

$|c^r_p(U^{k+1}) - c^r_q(U^{k+1})| = c^r_p(U^{k+1}) - c^r_q(U^{k+1})$

$= (c^r_p(U^{k+1}) - tmax^r) - (c^r_q(U^{k+1}) - tmax^r)$

$\leq (C^r_p(u^{k+1}_p) - C^r_p(tmax^r))(1+\rho) - (C^r_q(u^{k+1}_q) - C^r_q(tmax^r))(1-\rho)$, by the bounds on the drift rate

$\leq (2P + (1+\rho)(\beta + \delta + \varepsilon))(1+\rho) - (2P + (1+\rho)(\beta + \delta + \varepsilon) - \beta_1)(1-\rho)$

$= (1-\rho)\beta_1 + 2\rho(2P + \beta + \delta + \varepsilon)$. ∎

Next, we bound the additional spread introduced by the resetting of the clocks.

Lemma 5-15: Let s and t be the nonfaulty processes described above. Then

(a) $c^{r+1}_s(U^{k+1}) - c^r_s(U^{k+1}) \leq (1+\rho)(\varepsilon + \rho(4\beta + \delta + 5\varepsilon)$, and

(b) $c^r_t(U^{k+1}) - c^{r+1}_t(U^{k+1}) \leq (1+\rho)(\varepsilon + \rho(4\beta + \delta + 5\varepsilon)$.

Proof: (a) By Lemma 4-15, we know that s's new clock is at most $\alpha = \varepsilon + \rho(4\beta + \delta + 5\varepsilon)$ less than the "smallest" of the previous nonfaulty clocks at $c^r_s(U^{k+1}) = u^{k+1}_s$. Since s had the smallest clock before, $C^{r+1}_s(u^{k+1}_s) \geq C^r_s(u^{k+1}_s) - \alpha$. By the lower bound on the drift rate,

$c^{r+1}_s(U^{k+1}) - c^r_s(U^{k+1}) \leq (1+\rho)\alpha$.

(b) Lemma 4-15 also states that t's new clock is at most α more than the "largest" of the previous nonfaulty clocks at u^{k+1}_t, which was t's clock. The argument is similar to (a). ∎

Finally, we can bound the maximum difference in real time between two nonfaulty processes' clocks reaching T^{k+2}. Let i_p be the index of p's logical clock that is in effect when T^{k+2} is reached.

Theorem 5-16: Let p and q be nonfaulty processes and $i = i_p$ and $j = i_q$. Then

$|c^i_p(T^{k+2}) - c^j_q(T^{k+2})| \leq \beta$.

Proof: Without loss of generality, suppose $c^i_p(T^{k+1}) \geq c^j_q(T^{k+2})$. Then

$|c^i_p(T^{k+1}) - c^j_q(T^{k+2})| = c^i_p(T^{k+1}) - c^j_q(T^{k+2})$

$\leq c^{r+1}_s(T^{k+2}) - c^{r+1}_t(T^{k+2})$

for nonfaulty processes s and t that behave as described above.

We know from Lemma 4-2 that

$(c^{r+1}_s(T^{k+2}) - c^{r+1}_t(T^{k+2})) - (c^{r+1}_s(U^{k+1}) - c^{r+1}_t(U^{k+1}))$

$\leq 2\rho(P - (1 + \rho)(\beta + \delta + \varepsilon))$.

Thus $c^{r+1}_s(T^{k+2}) - c^{r+1}_t(T^{k+2})$

$\leq 2\rho(P - (1 + \rho)(\beta + \delta + \varepsilon)) + c^{r+1}_s(U^{k+1}) - c^{r+1}_t(U^{k+1})$

$= 2\rho(P - (1 + \rho)(\beta + \delta + \varepsilon)) + c^{r+1}_s(U^{k+1}) - c^{r}_s(U^{k+1}) + c^{r}_t(U^{k+1}) - c^{r+1}_t(U^{k+1})$

$+ c^{r}_s(U^{k+1}) - c^{r}_t(U^{k+1})$

$\leq 2\rho(P - (1 + \rho)(\beta + \delta + \varepsilon)) + 2(1 + \rho)(\varepsilon + \rho(4\beta + \delta + 5\varepsilon))$

$+ c^{r}_s(U^{k+1}) - c^{r}_t(U^{k+1})$, by Lemma 5-15

$\leq 2\rho(P - (1 + \rho)(\beta + \delta + \varepsilon)) + 2(1 + \rho)(\varepsilon + \rho(4\beta + \delta + 5\varepsilon))$

$+ (1 - \rho)\beta_1 + 2\rho(2P + \beta + \delta + \varepsilon)$, by Lemma 5-14

$\leq \beta$, by inequality (2). ∎

This β is approximately 6ε, which is slightly larger than the smallest one maintainable, 4ε. To shrink it back down, P can be made slightly smaller than required by the maintenance algorithm, as long as the lower bound of inequality (3) isn't violated. Since the synchronization procedure is performed more often, the clocks don't drift apart as much, and consequently, they can be more closely synchronized. Once the desired β is reached, P can be increased again. (The computational costs associated with performing the synchronization procedure and the possible degradation of validity may make it advisable to resychronize more infrequently.)

5.6 Using Only the Start-up Algorithm

A natural idea is to use Algorithm 5-1 solely, and never switch to the mainenance algorithm. Both algorithms can synchronize clocks to within approximately 4ε, so such a policy would sacrifice very little in accuracy. Using just the one algorithm is conceptually simpler and avoids introducing the additional error during the switch-over. However, if the system does no work during the period of time when processes have clocks with different indices, it is important to

minimize this interval. Algorithm 5-1 has such an interval of length $\delta + 3\varepsilon$; for Algorithm 4-1, it is approximately $\beta + 2\rho(\beta + \delta + \varepsilon)$. Depending on the choice of values for the parameters, Algorithm 4-1 may be superior in this regard.

Chapter Six

Conclusion

6.1 Summary

In conclusion, we have presented a precise formal model to describe a system of distributed processes, each of which has its own clock. Within this model we proved a lower bound on how closely clocks can be synchronized even under strong simplifying assumptions.

The major part of the thesis was the description and analysis of an algorithm to synchronize the clocks of a completely connected network in the presence of clock drift, uncertainty in the message delivery time, and Byzantine process faults. Since it does not use digital signatures, the algorithm requires that more than two thirds of the processes be nonfaulty. Our algorithm is an improvement over those in [7] based on Byzantine Agreement protocols in that the number of messages per round is n^2 instead of exponential, and that the size of the adjustment made at each round is a small amount independent of the number of faults.

The algorithm in [5] works for a more general communication network, and, since it uses digital signatures, only requires that more than half the processes be nonfaulty. However, the size of the adjustment depends on the number of faulty processes.

The issue of which algorithm synchronizes the the most closely is difficult to resolve because of differing assumptions about the underlying model. For instance, Algorithm 4-1 of this thesis can achieve a closeness of synchronization of approximately 4ε in our notation. However, we assume that local processing time is negligible; otherwise Lamport [8] claims that actually there is an implicit factor of n in the ε, in which case the closeness of synchronization achieved by our algorithm depends on the number of processes as do those in [7].

We also modified Algorithm 4-1 to produce an algorithm to establish synchronization initially among clocks with arbitrary values. This algorithm also handles clock drift, uncertainty in the message delivery time, and Byzantine process faults. This problem, as far as we know, had not been addressed previously for real-time clocks.

6.2 Open Questions

It would be interesting to know more lower bounds on the closeness of synchronization achievable. For example, a question posed by J. Halpern is to determine a lower bound when the communication network has an arbitrary configuration and the uncertainty in the message delivery time is different for each link.

There are also no known lower bounds for the case of clock drift and faulty processes.

The validity of algorithm 5-1 has not been computed. If this algorithm were used solely, knowing how the processes' clocks increase in relation to real time would be of interest. Lower bounds in general for the validity conditions are not known.

It seems reasonable that there is a tradeoff between the closeness of synchronization and the validity, since the synchronization procedure must be performed more often in order to synchronize more closely, but each resychronization event potentially worsens the validity. This tradeoff has not been quantified.

M. Fischer [4] has suggested an "asynchronous" version of Algorithm 5-1 to establish synchronization. In his version, a nonfaulty process wakes up at an arbitrary time with arbitrary values for its correction variable and array of differences. Every P as measured on its *physical* (not logical) clock, the process performs the fault-tolerant averaging function and updates its clock. It seems that the clock values should converge, but at what rate?

What kind of algorithms that use the fault-tolerant averaging function can be used in more general communication graphs?

Another avenue of investigation is using the fault-tolerant averaging function together with the capability for authentication to see if algorithms with higher fault-tolerance than those of this thesis and better accuracy than those in [5] can be designed.

Appendix A

Multisets

This Appendix consists of definitions and lemmas concerning multisets needed for the proofs of Lemmas 4-9 and 5-10. These definitions and lemmas are analogous to some in [1].

A *multiset* U is a finite collection of real numbers in which the same number may appear more than once. The largest value in U is denoted *max(U)*, and the smallest value in U is denoted *min(U)*. The *diameter* of U, *diam(U)*, is max(U) – min(U). Let *s(U)* be the multiset obtained by deleting one occurrence of min(U), and *l(U)* be the multiset obtained by deleting one occurrence of max(U). If $|U| \geq 2f + 1$, we define *reduce(U)* to be $l^f s^f(U)$, the result of removing the f largest and f smallest elements of U.

Given two multisets U and V with $|U| \leq |V|$, consider an injection c mapping U to V. For any nonnegative real number x, define $S_x(c)$ to be $\{u \in U: |u - c(u)| > x\}$. We define the *x-distance* between U and V to be $d_x(U,V) = \min_c\{|S_x(c)|\}$. We say c *witnesses* $d_x(U,V)$ if $|S_x(c)| = d_x(U,V)$. The x-distance between U and V is the number of elements of U that cannot be matched up with an element of V which is the same to within x. If $|u - c(u)| \leq x$, then we say u and c(u) are *x-paired* by c. The *midpoint* of U, *mid(U)*, is $\frac{1}{2}[\max(U) + \min(U)]$.

For any multiset U and real number r, define U + r to be the multiset obtained by adding r to every element of U; that is, $U + r = \{u + r: u \in U\}$. It is obvious that mid and reduce are invariant under this operation.

The next lemma bounds the diameter of a reduced multiset.

Lemma A-1: Let U and W be multisets such that $|U| = n$, $|W| = n - f$, and $d_x(W,U) = 0$, where $n \geq 2f + 1$. Then

$\max(\text{reduce}(U)) \leq \max(W) + x$ and $\min(\text{reduce}(U)) \geq \min(W) - x$.

Proof: We show the result for max; a similar argument holds for min. Let c witness $d_x(W,U)$. Suppose none of the f elements deleted from the high end of U are x-paired with elements of W by c. Since $d_x(W,U) = 0$, the remaining n – f elements of U are x-paired with elements of W by c, and thus every element of reduce(U) is x-paired with an element of W. Suppose max(reduce(U)) is x-paired with w in W by c. Then $\max(\text{reduce}(U)) \leq w + x \leq \max(W) + x$.

Now suppose one of the elements deleted from the high end of U is x-paired with an

element of W by c. Let u be the largest such, and suppose it was paired with w in W. Then max(reduce(U)) $\leq$ u $\leq$ w + x $\leq$ max(W) + x. ∎

We show that the x-distance between two multisets is not increased by removing the largest (or smallest) element from each.

Lemma A-2: Let U and V be multisets, each with at least one element. Then $d_x(l(U),l(V)) \leq d_x(U,V)$ and $d_x(s(U),s(V)) \leq d_x(U,V)$.

Proof: We give the proof in detail for l; a symmetric argument holds for s. Let M = l(U) and N = l(V). Let c witness $d_x(U,V)$. We construct an injection c' from M to N and show that $|S_x(c')| \leq |S_x(c)|$. Since $d_x(M,N) \leq |S_x(c')|$ and $|S_x(c)| = d_x(U,V)$, it follows that $d_x(M,N) \leq d_x(U,V)$.

Suppose u = max(U) and v = max(V). (These are the deleted elements.)

Case 1: c(u) = v. Define c'(m) = c(m) for all m in M. Obviously c' is an injection. $|S_x(c')| \leq |S_x(c)|$ since either $S_x(c') = S_x(c)$ or $S_x(c') = S_x(c) - \{u\}$.

Case 2: c(u) ≠ v and there is no u' in U such that c(u') = v. This is the same as Case 1.

Case 3: c(u) ≠ v, and there is u' in U such that c(u') = v. Suppose c(u) = v'. Define c'(u') = v' and c'(m) = c(m) for all m in M besides v'. Obviously c' is an injection. Now we show that $|S_x(c')| \leq |S_x(c)|$.

If u or u' or both are in $S_x(c)$ then whether or not u' is in $S_x(c')$ the inequality holds. The only trouble arises if u and u' are both not in $S_x(c)$ but u' is in $S_x(c')$. Suppose that is the case. Then $|u' - c'(u')| = |u' - v'| > x$. There are two possibilities:

(i) $u' > v' + x$. Since u is not in $S_x(c)$, $|u - c(u)| = |u - v'| \leq x$. So $v' \geq u - x$. Hence $u' > v' + x \geq u - x + x$, which implies that $u' > u$. But this contradicts u being the largest element of U.

(ii) $v' > u' + x$. Since u' is not in $S_x(c)$, $|u' - c(u')| = |u' - v| \leq x$. So $u' \geq v - x$. Hence $v' > u' + x \geq v - x + x$, which implies that $v' > v$. But this contradicts v being the largest element of V.

∎

The next lemma shows that the results of reducing two multisets, each of whose x-distance from a third multiset is 0, can't contain values that are too far apart.

Lemma A-3: Let U, V, and W be multisets such that |U| = |V| = n and |W| = n – f, where n > 3f. If $d_x(W,U) = 0$ and $d_x(W,V) = 0$, then

min(reduce(U)) – max(reduce(V)) $\leq$ 2x.

Proof: First we show that $d_{2x}(U,V) \leq f$. Let c_U witness $d_x(W,U)$ and c_V witness $d_x(W,V)$. Define an injection c from U to V as follows: if there is w in W such that $c_U(w)$ = u, then let $c(u) = c_V(w)$; otherwise, let c(u) be any unused element of V. For each of

the n – f elements w in W, there is u in U such that $u = c_U(w)$. Thus $|u - c(u)| \leq |u - w| + |w - c(u)| = |c_U(w) - w| + |w - c_V(w)| \leq x + x = 2x$. Thus $S_{2x}(c) \leq f$, so $d_{2x}(U,V) \leq f$.

Then by applying Lemma A-2 f times, we know that $d_{2x}(\text{reduce}(U),\text{reduce}(V)) \leq f$. Since $|\text{reduce}(U)| = |\text{reduce}(V)| = n - 2f > f$, there are u in reduce(U) and v in reduce(V) such that $|u - v| \leq 2x$. Thus $\min(\text{reduce}(U)) - \max(\text{reduce}(V)) \leq u - v \leq 2x$. ∎

Lemma A-4 is the main multiset result. It bounds the difference between the midpoints of two reduced multisets in terms of a particular third multiset.

Lemma A-4: Let U, V, and W be multisets such that $|U| = |V| = n$ and $|W| = n - f$, where $n > 3f$. If $d_x(W,U) = 0$ and $d_x(W,V) = 0$, then

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

Proof: $|\text{mid}(\text{reduce}(U)) - \text{mid}(\text{reduce}(V))|$

$= \frac{1}{2}|\max(\text{reduce}(U)) + \min(\text{reduce}(U)) - \max(\text{reduce}(V)) - \min(\text{reduce}(V))|$

$= \frac{1}{2}|\max(\text{reduce}(U)) - \min(\text{reduce}(V)) + \min(\text{reduce}(U)) - \max(\text{reduce}(V))|$

If the quantity inside the absolute value signs is nonnegative, this expression is equal to

$\frac{1}{2}[\max(\text{reduce}(U)) - \min(\text{reduce}(V)) + \min(\text{reduce}(U)) - \max(\text{reduce}(V))]$

$\leq \frac{1}{2}(\max(W) + x - (\min(W) - x) + \min(\text{reduce}(U)) - \max(\text{reduce}(V)))$, by applying Lemma A-1 twice

$= \frac{1}{2}(\text{diam}(W) + 2x + \min(\text{reduce}(U)) - \max(\text{reduce}(V)))$

$\leq \frac{1}{2}(\text{diam}(W) + 2x + 2x)$, by Lemma A-3

$= \frac{1}{2}\text{diam}(W) + 2x$.

If the quantity inside the absolute value is nonpositive, then symmetric reasoning gives the result. ∎

References

[1] D. Dolev, N. Lynch, S. Pinter, E. Stark and W. Weihl.
Reaching Approximate Agreement in the Presence of Faults.
In *Proceedings of the 3rd Annual IEEE Symposium on Distributed Software and Database Systems*. 1983.

[2] D. Dolev, J. Halpern and R. Strong.
On the Possibility and Impossibility of Achieving Clock Synchronization.
In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*. 1984.

[3] C. Dwork, N. Lynch and L. Stockmeyer.
Consensus in the Presence of Partial Synchrony.
In *Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing*. 1984.

[4] M. Fischer.
Personal communication.

[5] J. Halpern, B. Simons and R. Strong.
Fault-Tolerant Clock Synchronization.
In *Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing*. 1984.

[6] L. Lamport.
Time, clocks, and the ordering of events in a distributed system.
Communications of the ACM 21(7), July, 1978.

[7] L. Lamport and P.M. Melliar-Smith.
Synchronizing clocks in the presence of faults.
Research Report, SRI International, March, 1982.

[8] L. Lamport.
Personal communication.

[9] K. Marzullo.
Loosely-Coupled Distributed Services: a Distributed Time Service.
PhD thesis, Stanford University, 1983.