Trade-Off Results for Connection Management

*Marios Mavronicolas** & *Nikos Papadakis*

Department of Computer Science, University of Cyprus, Nicosia 1678, Cyprus

Abstract. A *connection management protocol* establishes a connection between two hosts across a wide-area network to allow reliable message delivery. Following previous work of Kleinberg *et al.* (*Proceedings of the 3rd Israel Symposium on the Theory of Computing and Systems*, pp. 258–267, January 1995), we study the precise impact of the behavior of processors' clocks with respect to real time on the performance of connection management protocols, under common assumptions on the pattern of failures of the network and the host nodes.

Two basic timing paradigms for clocks are considered: clocks that exhibit certain kind of a *drift* from the rate of real time, and clocks that display a pattern of *synchronization* to real time; within each paradigm, several timing conditions on the clocks are assumed, giving rise to corresponding timing models. We consider networks that can duplicate and reorder messages, and nodes that can crash. We are interested in simultaneously optimizing the following performance parameters: the *message delivery time*, which is the time required to deliver a message, and the *quiescence time*, which is the time that elapses between periods of *quiescence*, in which a processor returns to an initial state and deletes all earlier connection records. We establish trade-offs between message delivery time and quiescence time, in the form of tight lower and upper bounds, for each combination of timing models and failure types. Several special cases of our trade-off results significantly improve upon or extend previous ones shown by Kleinberg *et al.*

1 Introduction

Transport layer protocols, such as the TCP/IP Internet Suite, provide a reliable connection between two remote hosts, a *sender* and a *receiver*, across a wide-area network (see, e.g. [8, Chapter 6]). The sender wishes to establish a connection to the receiver, transmit information, and later release the connection. A *connection management protocol* handles the establishment and release of the connection. In turn, protocols built over the transport layer form the basis for ftp, telnet, remote procedure calls, and a number of other communication primitives that rely on reliable connections. In a large network, each sender typically maintains a number of parallel sessions; moreover, there can be a number of different

* Partially supported by funds for the promotion of research at University of Cyprus (research projects "Distributed Processing: Multiprocessors, Networks, and Verification," and "Distributed, Parallel and Concurrent Computations").

incarnations of a session with a single receiver, as the connection is opened, closed and opened again. In the presence of network faults such as message reordering or duplication, it is necessary to maintain records at each receiver keeping track of which packets have been received, acted on, and so forth. As the number of parallel sessions increases, however, memory limitations do not allow processing nodes to keep history records for very long. So, each processor must periodically *quiesce* by deleting past connection records.

Message delivery time determines the latency of packet transmission; thus, for applications with short incarnations, such as remote procedure calls, it is particularly important to keep message delivery time as low as possible. On the other hand, quiescence time determines the amount of information that needs to be stored at each node; for applications involving steady stream-like traffic with a priori known resource requirements, it is even necessary to keep quiescence time as small as possible, so that available buffer space at each individual node is not exhausted. A large number of protocols have been proposed to optimize either message delivery time or quiescence time (see, e.g. [1, 4, 7]). On the one extreme, timer-based protocols (see, e.g., [4]) achieve low message delivery time, while on the other extreme, the *three-packet handshake* protocol (see, e.g., [1]) guarantees low quiescence time.

We assume, throughout, that the network can fail by duplicating and reordering messages. We also consider node failures, where the receiver, but not the sender, may fail by crashing. We assume, however, that the receiver may not maintain in stable storage the time of its last crash, since, otherwise, by a trivial reduction to the case of message duplications (see [3, Section 6.2]), the receiver may deliver any message sent after its last crash. We consider two basic timing models. In the *drifting clocks* model, the clocks of the sender and the receiver run at a rate that may vary with time but always remains between $1/\rho$ and ρ times the rate of real time, for some fixed (and known) constant $\rho \geq 1$. In the *approximately synchronized clocks* model, or *approximate clocks* model for short, each of the clocks is always within ε of real time, for some fixed (and known) constant $\varepsilon \geq 0$. We follow [3] to express our time bounds in terms of two main parameters describing packet delays. The first of these parameters refers to a specific execution e of the system and is the *maximum packet delay in execution* e, denoted d_e; that is, d_e is the supremum of the times that elapse between the sending and the receipt of a packet in execution e. The second parameter of interest is the *maximum packet lifetime,* denoted μ, which is the longest amount of time, over all executions, that any copy of a packet may remain undelivered in the network; notice that μ is the maximum d_e over all executions e. While we may sometimes assume that μ is known, in contrast, neither S nor R may know d_e *a priori* in execution e.

We start with the case where there are network failures but not node failures. Our starting point is an ingenious connection management protocol designed by Kleinberg *et al* [3, Section 5] for the approximate clocks model in the presence of network failures. Roughly speaking, this protocol relies on a conservative estimation, made by the receiver, of the maximum delay in any specific execution;

these estimates are achieved through a "time-slicing" technique requiring both the sender and the receiver to use their (approximate) clocks in order to send to each other one time-stamped packet per each "time-slice". In turn, these estimates enable the receiver to determine when to deliver or quiesce. We make a key observation that the safety condition satisfied by this protocol, namely that it does not deliver a message twice, holds *independently* of the particular timing assumptions made for the approximately synchronized clocks model.[2] This leads us to consiider this protocol as a candidate of a *generic* connection management protocol which guarantees at-most-once message delivery in the presence of network failures for *any* model in which clocks are available to the sender and the receiver. Such a generic protocol would enjoy nice portability properties across models in which the available clocks satisfy different timing assumptions, but also work correctly for models in which the timing properties of the clocks are even *unknown,* or not amenable to a clean formalization.

There is, however, an additional, natural performance requirement on a generic connection management protocol. Since different applications may have different needs regarding which one of message delivery time and quiesce time to minimize while still retaining the other *bounded,* a connection management protocol is competitive in performance only if it appropriately allows such trade-offs between its message delivery time and quiescence time. Unfortunately, as we explain, the connection management protocol of Kleinberg *et al.* [3, Section 5] fails to allow such quantitative trade-offs. For the approximate clocks model, this protocol achieves upper bounds of $(1 + 2/\delta)d_e + (4 + 4/\delta)\varepsilon + c$ and $(\delta + 2)d_e + (2\delta + 6)\varepsilon + c$ on message delivery time and quiescence time, respectively, for any constant $c > 0$, where $\delta \geq 1$ is a "trade-off" parameter (cf. [3, Theorem 5]). Increasing δ lowers the upper bound on message delivery time but raises the upper bound on quiescence time; on the other hand, decreasing δ raises the upper bound on message delivery time but lowers the upper bound on quiescence time. Notice that the upper bound on message delivery time increases as δ decreases down to 1 but still remains bounded above by a *finite* quantity, namely $3d_e + 8\varepsilon + c$; unfortunately, the same does not hold on the way the upper bound on quiescence time increases as δ increases: the limit of the upper bound on quiescence time, as δ becomes large, is infinity. Thus, the connection management protocol of Kleinberg *et al.* [3, Section 5] renders itself "impractical," due to unbounded increase in the amount of connection records that needs to be kept at each node, for applications requiring the latency of packet transmission to become arbitrarily low. Call a connection management protocol *bounded* if the upper bounds it achieves on message delivery time and quiescence time are both bounded functions of the involved trade-off parameters. The work of Kleinberg *et al.* [3] leaves open the question of whether there exists or not a generic *and* bounded connection management protocol. We resolve this question by a judi-

[2] An inspection of the proof of [3, Theorem 5] reveals that the timing assumptions in the approximate clocks model are explicitly used in the analysis of the performance of this protocol, namely in deriving upper bounds on the message delivery time and quiescence time it achieves, but not in its correctness proof.

cious adjustment of the timing conditions which the receiver uses to determine when to deliver or quiesce in the generic protocol of Kleinberg *et al* [3]; the result is another generic connection management protocol which is also bounded for the approximately synchronized clocks model (see discussion next).

Assume first that clocks are drifting. We establish a trade-off between message delivery time and quiescence time that must hold for *some* execution of any connection management protocol. More specifically, we show that for any fixed constant δ, $0 \leq \delta \leq 2$, either a lower bound of $(3 - \delta\rho)d_e$ on message delivery time holds, or a lower bound of $\rho^2(\mu - (3 - \delta)d_e)$ on quiesce time holds for some execution e of any arbitrary connection management protocol. Kleinberg *et al.* [3, Theorem 4] show that for any connection management protocol there exists an execution e with $d_e < \mu/3$ for which either a lower bound of $3d_e$ on message delivery time holds or a lower bound of $\rho^2(\mu - 3d_e)$ on quiesce time holds. Our result extends and improves upon [3, Theorem 4] in a significant way: it is a substantial refinement of [3, Theorem 4] by incorporating the trade-off parameter δ; note that [3, Theorem 4] is the special case of our result with $\delta = 0$.

We next turn to the case of approximately synchronized clocks. We present both lower and upper bounds. We start with lower bounds. Kleinberg *et al.* [3, Section 5] consider the special case of perfect clocks (i.e., approximately synchronized clocks with $\varepsilon = 0$); in particular, Kleinberg *et al.* show that a certain trade-off between message delivery time and quiesce time must hold for *some* execution of any connection management protocol. In more detail, Kleinberg *et al.* [3, Theorem 6] show, assuming $\varepsilon = 0$, that for any connection management protocol, for any constant δ' where $0 < \delta' < 2$, there exists some execution e for which either a lower bound of $(1 + \delta')d_e$ on message delivery time holds, or a lower bound of $\min\{\mu, 2d_e/\delta'\}$ on quiesce time holds; notice that the latter lower bound never exceeds μ, Kleinberg *et al.* remark [3, Section 5]: "For general $\varepsilon > 0$, we do not know how to obtain a correspondingly tight lower bound, and leave this as an open question." We resolve this open problem of Kleinberg *et al.* by presenting a corresponding trade-off result for the case of general ε. More specifically, we show that for any fixed constant $\delta > 1$, either a lower bound of $(3 - 2/\delta)d_e + \varepsilon$ on message delivery time holds, or a lower bound of $(\delta/(\delta - 1))d_e + \varepsilon$ on quiesce time holds for some execution e of any arbitrary connection management protocol.

For purpose of direct comparison to the previous result of Kleinberg *et al.* [3, Theorem 6], which holds for the special case where $\varepsilon = 0$, set $\delta' = 2(1 - 1/\delta)$ where $\delta > 1$; under this substitution, the lower bounds on message delivery time and quiescence time in that result can be expressed as $(3 - 2/\delta)d_e$ and $\min\{\mu, (\delta/(\delta - 1))d_e\}$, respectively. We remark that these expressions are almost identical to those obtained by setting $\varepsilon = 0$ in the corresponding lower bounds we have shown. Clearly, our results imply that the timing uncertainty ε in the approximately synchronized clocks model incurs an additive overhead that is proportional to ε on each of the message delivery time and the quiescence time. Our trade-off result improves upon the corresponding result of Kleinberg *et al.* [3,

Theorem 6] in two significant ways. First, it extends [3, Theorem 6] to the case of general $\varepsilon \geq 0$. Second, when specialized for the case where $\varepsilon = 0$, the lower bound of $(\delta/(\delta-1))d_e$ on quiesce time improves upon the corresponding lower bound of $\min\{\mu, (\delta/(\delta-1))d_e\}$, shown in [3, Theorem 6], since $\min\{\mu, (\delta/(\delta-1))d_e\} \leq \mu$, while it can be verified that $(\delta/(\delta-1))d_e$ exceeds μ if δ is chosen so that $\delta < \mu/(\mu-\delta)$.

We continue with upper bounds. We use the timing assumptions made in the approximately synchronized clocks model to carry out a careful timing analysis of our generic connection management protocol. This analysis reveals upper bounds on message delivery time and quiesce time which not only incorporate the trade-off parameter δ, but also improve upon the corresponding upper bounds achieved by the corresponding protocol in [3, Theorem 5]. More specifically, we show upper bounds of $(3-1/\delta)d_e+(4-1/\delta)2\varepsilon+c$ and $(3+1/\delta)d_e+(4+1/\delta)2\varepsilon+c$ on message delivery time and quiesce time, respectively, for any constant $c > 0$, where $\delta \geq 1$ is a "trade-off" parameter. Notice that each of these upper bounds converges to the finite quantity $3d_e + 8\varepsilon + c$ as δ approaches infinity; this implies that specializing our generic connection management protocol to the approximately synchronized clocks model yields a bounded protocol for this case. In contrast, the generic connection management protocol of Kleinberg *et al.* [3, Section 5] achieves upper bounds of $(1+2/\delta)d_e+(4+4/\delta)\varepsilon+c$ and $(\delta+2)d_e+(2\delta+6)\varepsilon+c$ on message delivery time and quiesce time, respectively; these bounds imply that the protocol of Kleinberg *et al.* for the approximately synchronized clocks model is not bounded.

We next turn to the case where there are both network and node failures. Assume first that clocks are drifting. We establish a lower bound on message delivery time that must hold for *some* execution. More specifically, we show that for any arbitrary connection management protocol, there exists an execution e of it with $d_e < \mu/(3\rho+1)$ for which a lower bound of $3\rho d_e$ holds. We next turn to the case of approximately synchronized clocks. We present two lower bounds on message delivery time which trade-off strength and generality. First, we show that for any connection management protocol, there exists an execution e of it with $d_e \geq \varepsilon$ for which a lower bound of $d_e + 2\varepsilon$ holds. Second, we show that a stronger assumption on the execution e suffices to show a larger lower bound on message delivery time. More specifically, we show that if, in addition to $d_e \geq \varepsilon$, it is also assumed that $d_e \leq (\mu-6\varepsilon)/5$, then a lower bound of $3d_e+2\varepsilon$ on message delivery time holds. Due to space limitations, many of our definitions and proofs are only sketched in this extended abstract.

2 Definitions and Background

Our definitions of the system architecture closely follow those in [3]. The system we model consists of two nodes S (*sender*) and R (*receiver*), a host at each node, and an unreliable network connecting the two nodes. The sender wishes to transmit a message to the receiver: the receiver is required to eventually deliver the message, but never deliver it for a second time. We model the system as a

collection of I/O automata [5]. The automata U_S and U_R represent the users, one at each node; U_S wishes to transmit a message to U_R. The automata S and R represent the network interfaces for U_S and U_R, respectively. The network is also modeled as an automaton. The automaton U_S provides inputs to S, consisting of messages to be delivered; in correspondence, R provides inputs to U_R representing delivered messages. We model S and R as timed I/O automata [6], augmented with appropriate liveness properties [2]. Each state of S or R contains a special *clock* component, and a special *internal* component. Denote $\Re$ the domain of *real time*. A *clock* is a monotone non-decreasing and unbounded, piece-wise continuous function of real time $\gamma : \Re \rightarrow \Re$; each of S and R can allow any specified amount of time to pass on its clock, and perform certain actions when the clock reaches a specified value. We consider two main clock types: clocks that may "drift" away from real time, and clocks that are approximately synchronized. Fix any constant $\rho \geq 1$. A *ρ-drifting clock,* or *drifting clock* for short, is a clock $\gamma : \Re \rightarrow \Re$ such that for all real times $t_1, t_2 \in \Re$ with $t_1 < t_2$, $1/\rho \leq (\gamma(t_2) - \gamma(t_1))/(t_2 - t_1) \leq \rho$. Roughly speaking, a ρ-drifting clock "runs" at a rate between $1/\rho$ and ρ times that of real time. Since $\rho \geq 1$, it follows that if $t_2 > t_1$, then $\gamma(t_2) - \gamma(t_1) > 0$; thus, a ρ-drifting clock is strictly increasing. Throughout, fix any constant $\varepsilon \geq 0$. An *ε-synchronized clock,* or *approximately synchronized clock* for short, is a clock $\gamma : \Re \rightarrow \Re$ such that for each real time $t \in \Re$, $|\gamma(t) - t| \leq \varepsilon$ Roughly speaking, an ε-synchronized clock is always within ε of real time.

Initially, the internal components of S and R are equal to "initial" values $Q_{0,S}$ and $Q_{0,R}$, respectively: no local action is enabled in an initial state. *Quiescence* is modeled as a transition of the internal component of the state R to $Q_{0,R}$; this transition does not affect the clock component of the state of R. *Crash* is modeled identically to quiescence. S and R communicate through *packets* sent across the network. Events $send_S$ and $send_R$ denote packet sending events at S and R, respectively; similarly, events $receive_S$ and $receive_R$ denote packet receive events. We assume a function ϕ, which maps each $receive_S$ event (resp., $receive_R$ event) to a $send_R$ event (resp., $send_S$ event) for the same packet. For a network that can duplicate packets, ϕ need not be one-to-one; that is, $receive_S$ or $receive_R$ may occur more than once for any single packet sent once. However, we will assume that each single packet can be duplicated only a finite number of times. We similarly consider *message transmit* events and *message delivery* events occurring at U_S and U_R as higher primitives. We are concerned with the delivery of a single *message*; thus, we assume that U_S provides the message as a single input to S, at the beginning of an execution. Thus, for any given execution, the inputs to S consist of an initial input u^* from U_S, followed by a sequence of packets $r_1, r_2, \ldots$ from R; the inputs to R consist of a sequence of packets $s_1, s_2, \ldots$ from S. If the network can duplicate and reorder packets, the correctness condition for any *connection management* protocol is as follows. For every execution e beginning with the input of a message from U_S to S, there is exactly one event in e in which R delivers the message to R, and at least one event following this delivery event in which R quiesces. Assume that these events

occur at times $\mathbf{D}(e)$ and $\mathbf{Q}(e)$, respectively.

A *trade-off* connection management protocol $\mathcal{P}$ is a connection management protocol for which there exists a parameter $\delta \geq 0$ such that for any timed execution e of $\mathcal{P}$ both $\mathbf{D}(e)$ and $\mathbf{Q}(e)$ are bounded above by (non-constant) functions of δ, one of which is an ascending function of δ and the other is a descending function of δ. A *bounded* connection management protocol is a trade-off connection management protocol for which the one of the functions bounding $\mathbf{D}(e)$ and $\mathbf{Q}(e)$ that is an ascending function of δ converges to a finite upper bound as δ approaches infinity.

In the *drifting clocks* model, each of γ_S and γ_R is a ρ-drifting clock. In the *approximately synchronized clocks* model, each of γ_S and γ_R is an approximately synchronized clock. The approximately synchronized clocks model models situations where *external synchronization* is available to the processors. The definition of an ε-synchronized clock immediately implies that in the approximately synchronized clocks model both $|(\gamma_S(t_2) - \gamma_S(t_1)) - (t_2 - t_1)| \leq 2\varepsilon$ and $|(\gamma_R(t_2) - \gamma_R(t_1)) - (t_2 - t_1)| \leq 2\varepsilon$. A third immediate implication of the definition of the approximately synchronized clocks model is that $|\gamma_S(t) - \gamma_R(t)| \leq 2\varepsilon$. The *weakly synchronized clocks* model is defined as a weaker version of the approximately synchronized clocks model in which these three implications hold, while relaxing the requirement that the individual clocks be approximately synchronized. For any specific execution e, d_e is the maximum amount of time, over all packets, that can elapse between events $receive_P$ and $send_P$ that are related by ϕ; that is, d_e is the longest packet delay in execution e. The *maximum packet lifetime* μ is defined as the maximum d_e over all executions e; thus, d_e is always at most μ, although it can be substantially less than μ in "normal" executions. We will be assuming, unless otherwise stated, that both S and R "know" μ.

3 Network Failures: A Generic Protocol

In this section, we present a generic protocol $\mathcal{P}$ for connection management, which is based on time stamps. $\mathcal{P}$ solves connection management in the presence of network failures for any model for which processors have monotonically increasing clocks; as we will show later, $\mathcal{P}$ guarantees finite bounds for both message delivery time and quiescence time for the approximately synchronized clocks model and its weaker variants, and the mutually drifting clocks model. The protocol $\mathcal{P}$ is a variation of one proposed by Kleinberg *et al.* [3, Theorem 5], and our analysis closely follows the one in [3, section 5]. Throughout, fix any constant $c > 0$ and let δ be any parameter such that $\delta \geq 1$. Define a constant $c' = c'(\delta) = \delta c/(7\delta + 2)$. The protocol adopts a "time-slicing" technique. Each of S and R uses its local clock to "slice" time into intervals of length c'. At the end of each "slice", Each of S and R sends a "time-stamped" packet to the other; the "time stamp" is the local sending time. A discrete R-time t is a local time at R which is a positive integral multiple of c'; discrete S-time is defined correspondingly.

The *threshold of R at discrete R-time t* [3, Theorem 5] is defined to be the

largest t' for which R has received all S-packets with time stamp at most t'; that is, R has not yet received the S-packet with time stamp $t'+c'$). The *threshold of S at discrete S-time t* is defined analogously. The first packet sent by S contains both the message and the current local time. Subsequent S-packets consist of the current local time and the current threshold of S. Initially, S has received no packets from R and hence reports a trivial threshold; after it receives its first R-packet, it report a non-trivial threshold. R-packets consist simply of the current local time to enable S to compute its threshold. The first R-packet is sent when R first receives the initial S-packet. Assume that S sends its initial packet at discrete S-time 0. Let r_0 denote the discrete R-time at which R first receives the initial S-packets, and hence at which it sent out its first packet to S. R maintains an estimate of the current value of d_e by computing the maximum lag $l^{(t)}$ of any packets observed up to time t; this is equal $c' + M^{(t)}$, where $M^{(t)}$ is the maximum over the following three finite sets: (1) the set of all $r - s$, where the threshold of R at discrete R-time r is equal to s; (2) the set of all $s' - r'$, where the threshold value in the S-packets time- stamped s' is equal to r'; (3) the set of all $s' - r_0$, where the S-packet time-stamped s' reports a trivial threshold. (Note that the $M^{(t)}$ is an ascending function). Clearly, the third rule implies that $l^{(r_0)} > |r_0|$.

We are now ready to present the algorithm. R delivers at the first discrete R-time t' when $t' > (3 - 1/\delta)l^{(t')}$ and quiesces at the first discrete R-time t'' when $t^{(\prime\prime)} > (3 + 1/\delta)l^{(t'')}$ It then sends a done message to S; S quiesces immediately upon receiving this done message. If at any time S report a non trivial threshold that is less than r_0 (i.e. one can conclude that R is hearing replays), R aborts the connection without delivering and sends an error message to S. For any time t, define $r(t)$ to be the discrete R-time at which the maximum value for $l^{(t)}$ was attained; that is, $r(t)$ is largest $r \leq t$ for which the $l^{(r)} = l^{(t)}$. By the three rules, $d_e > \gamma_R^{-1}(r) - \gamma_S^{-1}(r - l^{(t)} + 2c')$. We continue to show that $\mathcal{P}$ is a connection management protocol. We need to prove that R does not deliver any message for a second time. First we argue that R will not quiesce until it has received an S-packet with non-trivial threshold. Let ψ denote the S-packet with minimal time-stamp that reports a non-trivial threshold, and consider discrete R-time r at which R has not yet received ψ. Let $r - u_1$ be the time-stamp of the most recent S-packets, and set $v = r - u_1 - r_0$. Because $r - u_1$ is the time-stamp of the most recent S-packets we have that $r - u_1 \geq r_0$. Since $M^{(t)}$ is an ascending function, it follows that $M^{(r)} \geq M^{(r-u_1)} \geq M^{(r_0)}$ Together with the three rules, this implies that $l^{(r)} \geq u_1$, $l^{(r)} > r_0$, $l^{(r)} \geq u$. Thus, $r = r_0 + u_1 + u \leq 3l^{(r)} \leq (3 + 1/\delta)l^{(r)}$, so R will not yet quiesce. Now let l^* (resp. M^*) denote the maximum value of $l^{(t)}$ (resp. $M^{(t)}$) over all discrete R-times t up to quiescence, and s_1 denote the time-stamp of S-packet ψ. Indeed, the time-stamped $s_1 - c'$ reports a trivial threshold, so by the third rule for estimating the lag, $M^* \geq s_1 - c' - r_0$, it follows $l^* \geq s_1 - r_0$. Because $l^{(t)}$ is an ascending function, $l^* \geq l^{(r_0)} > r_0$, and a manipulation yields that $s_1 < 2l^*$. Finally, suppose $\mathbf{T} > t''$ and a replay of the original message arrives at time $\mathbf{T}$. We will show that if $\mathbf{T}' \geq \mathbf{T}$ is some time at which R has not received a second replay of S-packet ψ, then it will

not required to deliver. Since ψ has not been received at $\mathbf{T}'$, by the first rule for estimating the lag, $l^{(\mathbf{T}')} \geq \mathbf{T}' - s_1 > \mathbf{T}' - 2l^*$. Since R quiesces by R-time T and l^* denote the maximum value $l^{(t)}$ over all discrete R-times t up to quiescence, we have that $(3 + 1/\delta)l^* \leq \mathbf{T}$, so that $l^* \leq (\delta/(3\delta + 1))\,\mathbf{T}$. Thus, $l^{(\mathbf{T}')} > \mathbf{T}' - 2l^* \geq \mathbf{T}' - \frac{2\delta}{3\delta+1}\,\mathbf{T} \geq \mathbf{T}' - \frac{2\delta}{3\delta+1}\,\mathbf{T}' = \frac{\delta+1}{3\delta+1}\,\mathbf{T}'$, which implies that $\mathbf{T}' < \frac{3\delta+1}{\delta+1}\,l^{(\mathbf{T}')} < (3 - \frac{1}{\delta})\,l^{(\mathbf{T}')}$, Thus R does not deliver at time $\mathbf{T}'$. As we have say T is the R-time which R receives the first replay of ψ report and $\mathbf{T}' \geq \mathbf{T}$ is a R-times in which R has not receives a second replay of ψ yet. It follows that R will receive a replay of ψ before it delivers. But ψ reports a threshold $(= r_0)$ smaller than $\mathbf{T}$,which is the discrete R-time at which R first started sending packets to S following quiescence. By the protocol, R will abort the connection in this case. Thus R is not required to deliver until it receives a replay of ψ. Thus, R never delivers the message a second time, which implies:

Theorem 1. *$\mathcal{P}$ is a connection management protocol.*

4 Network Failures: Drifting Clocks

We establish a trade-off between message delivery time and quiesce time that must hold for *some* execution of any connection management protocol.

Theorem 2. *Consider the drifting clocks model, under network failures. Then, for any connection management protocol $\mathcal{P}$, for any constant δ such that $0 \leq \delta \leq 2$, there exists a timed execution e of $\mathcal{P}$ such that either $\mathbf{D}(e) \geq (3 - \delta\rho)d_e$, or $\mathbf{Q}(e) \geq \rho^2(\mu - (3 - \delta)d_e)$.*

Sketch of proof: Assume, by way of contradiction, that there exists a connection management protocol $\mathcal{P}$ for the drifting clocks model in the presence of network failures, and a constant δ, $0 \leq \delta \leq 2$, such that for every timed execution e of $\mathcal{P}$, both $\mathbf{D}(e) < (3 - \delta\rho)d_e$, and $\mathbf{Q}(e) < \rho^2(\mu - (3 - \delta)d_e)$. We construct a timed execution of $\mathcal{P}$ in which a message is delivered twice. We construct a sequence of timed executions e, f, e', and f', so that R delivers a message twice in f'. f' is the "concatenation" of e' and f. In e and f, the clocks of R and S are "slow", while in e', the clocks of R and S are "fast". We start with e, which terminates immediately after R quiesces. By modifying R's clock, we "perturb" e to obtain f, which S cannot distinguish from e; still, f terminates immediately after R quiesces. We continue to construct e', which S cannot distinguish from e to S, while R still delivers in e' but does not quiescence. Finally, we construct f' as the "concatenation" of e' and f; in f', R first delivers and quiescences, before it receives replays of all packets in a way that R "sees" them arriving as in f. This leads R to deliver again, which contradicts the correctness of $\mathcal{P}$. ■

The lower bounds on message delivery time and quiescence time shown in Theorem 2 are simultaneously non-negative, and, hence, non-trivial, if (and only if) both $3 - \delta\rho \geq 0$ and $\mu - (3 - \delta)d_e \geq 0$. Eliminating δ and assuming $\rho > 1$

yields $d_e \leq \frac{\rho}{\rho-1}\frac{\mu}{3}$ as a necessary condition for any timed execution e for which the trade-off lower bounds shown in Theorem 2 are non-trivial; Kleinberg *et al.* [3, Theorem 4] argue that $d_e \leq \mu/3$ is a corresponding necessary condition. Since $\frac{\rho}{\rho-1}\frac{\mu}{3} \geq \frac{\mu}{3}$, for $\rho > 1$, this implies that the trade-off lower bound shown in Theorem 2 is non-trivial for a wider range of timed executions than the trade-off lower bound shown in [3, Theorem 4].

5 Network Failures: Approximate Clocks

We start by showing:

Theorem 3. *Consider the approximately synchronized clocks model, in the presence of network failures. Fix any parameter $\delta > 1$. Then, for any connection management protocol $\mathcal{P}$, there exists an execution e of $\mathcal{P}$ for which either* $\mathbf{D}(e) \geq (3 - \frac{2}{\delta})d_e + \varepsilon$, *or* $\mathbf{Q}(e) \geq \frac{\delta}{\delta-1}d_e + \varepsilon$.

Sketch of proof: Assume, by way of contradiction, that there exists a connection management protocol $\mathcal{P}$ for the approximately synchronized clocks model under network failures, and a constant $\delta > 1$ such that for every execution e of $\mathcal{P}$, both $\mathbf{D}(e) < (3 - 2/\delta)d_e + \varepsilon$, and $\mathbf{Q}(e) < \delta/(\delta - 1)d_e + \varepsilon$. We construct an execution of $\mathcal{P}$ in which a message is delivered twice. We construct a sequence of executions e, f, e' so that R delivers a message twice in e'. We start with execution e which terminates with R's quiescence following its delivery. We continue to construct f which is indistinquishable from e to S, while R only delivers. The incurred delays on packets are larger in f than in e. Finally, we construct e' as the "concatenation" of e and f. In e', R first delivers and immediately quiesces, and it next receives replays of all packets in such a way that R "sees" all packets arriving as in f. By construction of f, R delivers again, constradicting the correctness of $\mathcal{P}$. ■

We continue with upper bounds. We use the timing assumptions made in the approximately synchronized clocks model to carry out a careful timing analysis of our generic connection management protocol and show:

Theorem 4. *Consider the approximately synchronized clocks model in the presence of network failures. Then, for any constants $\delta \geq 1$ and $c > 0$, there exists a connection management protocol $\mathcal{P}$ such that for every timed execution e of $\mathcal{P}$,* $\mathbf{D}(e) < (3 - \frac{1}{\delta})\,d_e + (4 - \frac{1}{\delta})\,2\varepsilon + c$, *and* $\mathbf{Q}(e) < (3 + \frac{1}{\delta})\,d_e + (4 + \frac{1}{\delta})\,2\varepsilon + c$.

The bounds shown in Theorem 4 still hold if the timing model is relaxed to the weakly synchronized clocks model, to imply a bounded and generic connection management protocol for this model, too.

6 Network and Node Failures

In this section, we present our lower bounds under both network and node failures. We start with the drifting clocks model. We show:

Theorem 5. *Consider the drifting clocks model, in the presence of network and node failures. Then, for any connection management protocol $\mathcal{P}$, there exists an execution e of $\mathcal{P}$ with $d_e < \mu/3$ for which $\mathbf{D}(e) \geq 3\rho d_e$.*

Sketch of proof: Assume, by way of contradiction, that there exists a connection management protocol $\mathcal{P}$ such that for every execution e with $d_e < \mu/3$, $\mathbf{D}(e) < 3\rho d_e$. We construct an execution of $\mathcal{P}$ in which a message is delivered twice. We construct a sequence of executions e, e', f, and f' so that R delivers a message twice in f'. In all of these executions, clocks run at a rate of $1/\rho$ times that of real time. We start with any execution e which terminates with R delivering a message and immediately crashing. We "perturb" e to obtain e' which is indistinguishable from e to either S or R, while all messages from R to S take time μ in e'; so, R still delivers and immediately crashes by the end of e'. We continue to construct f which is indistinguishable from e' to S, while R only delivers in f but does not crash; the construction uses the fact that communication from R to S is slow in e'. Finally, we construct f' as the "concatenation" of e' and f; in f', R first delivers and crashes and next receives replays of all packets in such a way that R "sees" all packets arriving as in f. By the construction of f, R delivers again, contradicting the correctness of $\mathcal{P}$. ■

We continue with our two lower bounds for the approximately synchronized clocks model; these lower bounds trade-off strength and generality. We start with the more general but less strong lower bound.

Theorem 6. *Consider the approximately synchronized clocks model, under both network and node failures. Then, for any connection management protocol $\mathcal{P}$, there exists a timed execution e of $\mathcal{P}$ with $d_e \geq \varepsilon$ for which $\mathbf{D}(e) \geq d_e + 2\varepsilon$.*

Sketch of proof: Assume, by way of contradiction, that there exists a connection management protocol $\mathcal{P}$ such that for any execution e with $d_e \geq \varepsilon$, $\mathbf{D}(e) < d_e + 2\varepsilon$. We construct an execution of $\mathcal{P}$ in which a message is delivered twice. We construct a sequence of executions e, e', and f', so that R delivers a message twice in f'. In all of these executions, the clock of R "lags" by ε that of S. We start with any execution e which terminate with R delivering a message and immediately crashing. We "perturb" e to abtain e' which is indistinguishable from e to either S, while all messages incur a delay larger than the corresponding one in e. Finally, we continue to construct f' as the "concatenation" of e and e'; in f', R first delivers and crashes and next receives replays of all packets in such a way that R "sees" all packets arriving as in e'. By construction of e', R delivers again, contradicting the correctness of $\mathcal{P}$. ■

We continue to show a stronger but less general lower bound.

Theorem 7. *Consider the approximately synchronized clocks model, under both network and node failures. Then, for any constant δ, $0 \leq \delta \leq \delta$, for any connection management protocol $\mathcal{P}$, there exists an execution e of $\mathcal{P}$ with $d_e \leq (\mu - 2\delta\varepsilon)/5$ for which $\mathbf{D}(e) \geq 3d_e + \delta\varepsilon$.*

Sketch of proof: Assume, by way of contradiction, that there exists a connection management protocol $\mathcal{P}$ such that for any execution e with $d_e < (\mu - 2\delta\varepsilon)/5$, $\mathrm{D}(e) < 3d_e + \delta\varepsilon$. We construct an execution of $\mathcal{P}$ in which a message is delivered twice. We construct a sequence of executions e, e', f and f' so that R delivers a message twice in f'. In all of these executions, the clock of R "lags" by $\delta\epsilon/2$ that of S, for some constant δ, $0 \leq \delta \leq 2$. We start with execution e which terminates with R delivering a message and immediately crashing. We "perturb" e to obtain e' which is indistinguishable from e to either S or R, while all messages from R to S take time μ in e'; so, R still delivers and immediately crashes by the end of e'. We continue to construct f which is indistinguishable from e' to S, while R only delivers in f but does not crash; the construction uses the fact that communication from R to S is slow in e'. Finally, we construct f' as the "concatenation" of e' and f; in f', R first delivers and crashes and next receives replays of all packets in such a way that R "sees" all packets arriving as in f. By the construction of f, R delivers again, which contradicts the correctness of $\mathcal{P}$. ∎

Clearly, as δ increases, the lower bound of $3d_e + \delta\varepsilon$ is raised, but the upper bound of $(\mu - 2\delta\varepsilon)/5$ on d_e drops, thus narrowing the range of executions e for which the lower bound on message delivery time is valid. Since the weakly synchronized clocks model is no stronger than the approximately synchronized clocks model, Theorems 6 and 7 immediately imply corresponding results for that model too.

References

1. D. Belsnes, "Single-Message Communication," *IEEE Transactions on Communications,* Vol. 24, No. 2, February 1976.
2. R. Gawlick, R. Segala, J. Sogaard-Andersen and N. Lynch, "Liveness in Timed and Untimed Systems," *Proceedings of the 21st International Colloqium on Automata, Languages and Programming,* July 1994.
3. J. Kleinberg, H. Attiya and N. Lynch, "Trade-offs Between Message Delivery and Quiesce Times in Connection Management Protocols," *Proceedings of the 3rd Israel Symposium on the Theory of Computing and Systems,* pp. 258–267, January 1995.
4. B. Liskov, L. Shrira and J. Wroclawski, "Efficient At-Most-Once Messages Based on Synchronized Clocks," *ACM Transactions on Computer Systems,* Vol. 9, No. 2, pp. 125–142, 1991.
5. N. Lynch and M. Tuttle, "An Introduction to Input/Output Automata," *CWI Quarterly,* Vol. 2, No. 3, pp. 219–246, September 1989.
6. N. Lynch and F. Vaandrager, "Forward and Backward Simulations for Timing-Based Systems," in *Real-Time: Theory in Practice,* Lecture Notes in Computer Science, Vol. 600 (J. W. de Bakker, C. Huizing, W. P. de Roever and G. Rozenberg, eds.), pp. 397–446, Springer-Verlag, June 1991.
7. C. Sunshine and Y. Dalal, "Connection Management in Transport Protocols," *Computer Networks,* Vol. 2, pp. 454–473, 1978.
8. A. Tanenbaum, *Computer Networks,* Prentice Hall, 1988.