A Competitive Analysis for Retransmission Timeout*

Shlomi Dolev† Michael Kate‡ Jennifer L. Welch‡

January 24, 1999

Abstract

Protocols that provide reliable communication on top of a network that can lose packets rely on periodically retransmitting packets. The choice of retransmission timeout critically affects system performance. This paper presents a first step toward a theoretical study of the choice of retransmission timeout, based on competitive analysis. In general, competitive analysis compares the performance of an on-line algorithm to the performance of an optimal off-line algorithm, which has access to more information. In this context, the job of an algorithm is to choose the retransmission timeout interval; an off-line algorithm knows the exact message delays, while an on-line algorithm only knows upper and lower bounds on the delays. The performance measure of interest is the expected value of a linear combination of the number of packets used and the amount of time elapsed. An on-line algorithm for choosing the retransmission timeout is presented that is optimal with respect to the difference between its performance and that of an optimal off-line algorithm. The algorithm is also analyzed with respect to the ratio of its performance and that of an optimal off-line algorithm.

Keywords: retransmission timeout, competitive analysis.

1 Introduction

In communication networks processors exchange information (messages) by sending and receiving packets over communication channels. Typically, a packet transmitted (by the *sender*) over a communication network has a non-zero probability of being lost or corrupted and discarded (by the *receiver*). If this happens, retransmission of the packet that conveys the message

[1]This work was supported by NSF Presidential Young Investigator Award CCR-9396098 and Texas A&M University Engineering Excellence funds. An extended abstract of this work was presented in the 15th International Conference on Distributed Computing Systems, 1995.

[2]Department of Mathematics and Computer Science, Ben-Gurion University, Beer-Sheva, 84105, Israel. Email: `dolev@cs.bgu.ac.il`.

[3]Department of Computer Science, Texas A&M University, College Station, TX 77843, USA. Email: `{mkate,welch}@cs.tamu.edu`.

is required. The packet is repeatedly retransmitted until it is successfully delivered. Consequently, retransmission of packets is at the core of every protocol that guarantees reliable communication.

The key concept used by reliable communication protocols is the retransmission of packets by the sender until an appropriate acknowledgment arrives from the receiver (e.g., the alternating bit protocol [4], the sliding window protocol [11], etc.). The time between two successive retransmissions, called the *retransmission timeout*, is an important parameter of reliable transmission protocols [5]. The efficiency of a reliable transmission protocol is measured in terms of both the time it takes for message delivery through the network and the communication bandwidth used per message delivery. If the retransmission timeout is long, then it takes a long time for the message to be delivered; on the other hand, if the retransmission timeout is short, then the result is high usage of the communication bandwidth. The problem of finding an optimal retransmission timeout is made more complicated when the mutual influence between several senders has to be considered: Choosing a too short retransmission timeout may cause congestion in the network and increase the probability of a packet being lost.

Existing protocols for reliable message transmission use heuristics that depend on manually controlled parameters to tune the choice of retransmission timeout [12, 5]. In this paper, we take a step towards a formal model and analysis that captures the key aspects of the problem. We present a theoretical model and analysis for understanding the influence of the retransmission timeout on system performance. Our theoretical model captures fundamental aspects of the retransmission problem, and still retains the benefits of simplicity. A single message delivery between a sender and a receiver at different sites in the network is considered. A fixed round trip delay and probability of losing a packet during the process of a *single* message delivery is assumed. This is an approximation to the behavior of a multi-user network during a short period of time. In such a network the overall traffic pattern is typically changed gradually.

The efficiency of a protocol is measured with respect to a cost function. We begin by considering a linear combination of the elapsed time and the number of packets required for transmission of a message. The relative costs of time and number of packets are controlled by a parameter. The mutual influence between multiple senders is captured by an appropriate choice of the parameter: when the mutual influence is high, more weight is given to the number of packet used for delivering a message, otherwise more weight is given to the total time elapsed. Finally, we consider the expected value of this linear combination, given the fixed probability of packet loss.

Our analysis is based on the competitive on-line vs. off-line approach. Originally Sleator and Tarjan [10] suggested the on-line vs. off-line approach to analyze the performance of algorithms for dynamically maintaining a linear list and for paging. In each case, they devised an optimal off-line algorithm assuming (unrealistically) that the entire sequence of future requests (either for operations on the list or for memory references) is known. Then they presented an on-line algorithm for the same problem omitting the unrealistic assumption concerning the availability of future knowledge. The complexity of the on-line algorithm was compared with the complexity of the (optimal) off-line algorithm. Recently, the same approach has been applied to different tasks, e.g., [3, 9, 2, 1].

Our off-line algorithm is devised for the ideal case in which the processors know the exact round-trip delay, and it is optimal. In contrast, the on-line algorithm knows only upper and lower bounds on the current round trip delay. The on-line algorithm uses the bounds on the current round trip delay to determine a time-out period that minimizes the worst case difference in cost relative to the optimal off-line algorithm. An important benefit of competitive analysis is its capability to analyze an algorithm whose performance depends on events which occur with some unknown distribution. For example, in our case, although upper and lower bounds on the round trip delay are known to the on-line algorithm, the distribution of the round trip delay within those bounds is not. Instead, the on-line vs. off-line analysis gives the value of the round trip delay that illustrates the *worst* performance of the on-line algorithm relative to the off-line algorithm.

The retransmission time-out calculated by our on-line algorithm is optimal in the following sense: The maximal difference between the cost measures of the on-line algorithm and the off-line algorithm is minimized. Note that the traditional competitive analysis approach minimizes the (maximal) ratio between the on-line and off-line algorithms. However, we choose to minimize the (maximal) difference, which in many cases provides a stronger guarantee (cf. [7], p. 138). For the sake of completeness, we also present an upper bound on the ratio between the cost measures of the on-line and off-line algorithms. The upper bound on the ratio is a function of the upper and lower bounds on the round trip delay; for instance, if the round trip bounds differ by at most a factor of 4, then the cost measures ratio is at most 2.5.

Several parameters are used by the on-line algorithm to choose the optimal retransmission time-out, including: Lower and upper bounds on the round trip delay, the probability of a packet being lost, and the weighting parameter for the cost function. Throughout the analysis those parameters are assumed to be fixed. This models a short period of time in which only small changes may occur. However, when a sequence of messages is considered, these parameters may change and the solution should adapt in response to those changes. Our algorithm could be combined with procedures (outside the scope of this paper) for adaptively learning good approximations to the values of these parameters, say by using the stream of acknowledgment packets.

The remainder of the paper is organized as follows. Section 2 contains our definitions and assumptions. The optimal off-line algorithm is presented and analyzed in Section 3. The on-line algorithm is studied in Section 4. We conclude in Section 5.

2 Definitions and assumptions

2.1 The system and cost measure

We assume that the system consists of two processors, the *sender* and the *receiver*, that communicate by sending packets over a network in both directions. The processors have access to clocks that accurately measure the passage of time; the clocks need not be synchronized to each other. Each processor's clock ticks discretely and at each tick the processor takes a step.

The sender is assumed to have a single piece of information, called a *message*, that it wants to transfer to the receiver. The sender operates by repeatedly sending copies of the information in *data packets* to the receiver until it receives an acknowledgment from the receiver. The packets are assumed to be sent with a fixed period, called the *retransmission timeout.* The receiver operates simply by sending an *acknowledgment packet* to the sender whenever it receives a data packet from the sender.

The network can lose packets. However, it only delivers packets that were previously sent and it does not duplicate packets. (Thus, the sender alone controls any retransmissions.) We assume that within this single message transmission the network delivers those packets that it does not lose with a fixed delay, and thus delivers packets in FIFO order.

The problem we study is to determine how frequently the sender should retransmit data packets in order to achieve good performance. Thus we must also define an appropriate cost measure for evaluating performance.

The obvious cost measures of interest are P, the number of packets sent by the sender, and T, the amount of time measured in clock ticks at the sender until the sender receives an acknowledgment. There are two issues that must be resolved.

The first issue is that considering either P or T in isolation optimizes the performance of the algorithm only in one aspect. If we are only concerned with minimizing P, the number of packets, then the sender should wait a long time before deciding to retransmit, thus costing a lot of time. If we are only concerned with minimizing T, then the sender should send a packet at every step, thus costing a lot of packets. A large number of packets can cause a higher probability of congestion, which in turn may influence the probability of packet loss. Thus, we propose that the cost measure should charge for the number of packets, not only to reflect the bandwidth used but also to discourage congestion. We consider a cost measure that combines P and T and has a parameter f, $0 < f < 1$, that can be tuned to achieve different tradeoffs between P and T. This measure is similar to the phone company's method for billing—a fixed amount is paid monthly regardless of usage (analogous to T) plus charges for actual usage of the network (analogous to P). The definition is:

$$CM = fT + (1-f)P = P + (T-P)f$$

The units for CM are clock ticks. The intuition is that we count the number of ticks at which packets are sent and then add some fraction of the ticks during which the sender is waiting without sending any packets. As f approaches 0, CM approaches P, and as f approaches 1, CM approaches T.

The second issue is that P and T, and thus CM, clearly depend on how many packets are lost.[1] Since the number of lost packets can vary in different executions, we choose to focus on

[1]Unless stated otherwise, throughout this paper the term "packet loss" refers to the loss of either a data packet or its matching acknowledgment packet.

the expected value of *CM*, assuming that each packet has a fixed and independent probability of being lost. Thus the cost measure of interest we study is:

$$ECM = EX[P + (T - P)f]$$

We define the following system parameters:

- t_s, the retransmission timeout, i.e., the sender sends a data packet every t_s ticks,
- f, the parameter for *CM*, $0 < f < 1$,[2]
- p, the probability that either a data packet or its matching acknowledgment is lost, $0 < p < 1$,[3]
- v, the round trip delay for packets that are not lost, $v > 0$.

We first give a formula for *ECM* in terms of f, p, v, and t_s. We will primarily consider f and p as fixed, and vary v and t_s. *ECM* is a linear function of v. When considered as a function of t_s, *ECM* has the form $c_1 \cdot t_s + \frac{c_2}{t_s} + c_3$, for constants c_1, c_2, and c_3.

Theorem 1

$$ECM = \frac{fpt_s}{1-p} + \frac{v(1-f)}{t_s} + \frac{p(1-f)}{1-p} + fv.$$

Proof: Consider an execution in which i packets are lost in a row. Then $T = i \cdot t_s + v$, since the sender will wait t_s time after sending each of the lost packets and will wait v time after sending the $i+1$st packet until receiving the acknowledgment. Furthermore, $P = \frac{T}{t_s} = i + \frac{v}{t_s}$, since the sender sends a packet every t_s time units during a total interval of length T. Let *CM*(i) be the value of *CM* when i packets are lost in a row.

$$ECM = \sum_{i=0}^{\infty} Pr[i \text{ packets in a row are lost}] \cdot CM(i)$$

Substituting gives $ECM =$

$$\sum_{i=0}^{\infty} p^i(1-p)\left(i + \frac{v}{t_s} + \left(i \cdot t_s + v - i - \frac{v}{t_s}\right) f\right)$$

Rearranging, we get

[2]If $f = 0$, then $CM = P$, and the best strategy is for the sender to wait until it sure that the last packet is lost; if $f = 1$, then $CM = T$, and the best strategy is for the sender to resend the packet at every tick.

[3]If $p = 0$, then packets are never lost, and the best strategy is for the sender to send the packet once and wait for the acknowledgment; if $p = 1$, then every packet is lost and the message can never be transferred.

$$ECM = (1-p)(1+f\cdot t_s - f)\sum_{i=0}^{\infty} p^i \cdot i$$
$$+(1-p)\left(\frac{v}{t_s} + fv - \frac{v}{t_s}\cdot f\right)\sum_{i=0}^{\infty} p^i$$

From calculus, we get

$$ECM = (1-p)(1+f\cdot t_s - f)\frac{p}{(1-p)^2}$$
$$+(1-p)\left(\frac{v}{t_s} + fv - \frac{v}{t_s}\cdot f\right)\frac{1}{1-p}$$

This simplifies to the required quantity. ■

2.2 On-line and off-line algorithms

The algorithms we study are those for choosing the value of t_s.

We assume that an off-line algorithm knows the value of v, the round trip delay for non-lost packets, while an on-line algorithm only knows a range $[v_l, v_u]$ within which v lies. Both types of algorithms know f and p.

An off-line algorithm is *optimal* if, given any fixed values for v, p, and f, it calculates a value for t_s that minimizes the value of *ECM*.

We next define the "competitive difference" for an on-line algorithm. Given any fixed values for v_l, v_u, p, and f, the on-line algorithm is to calculate a value t_s^{on} for t_s. The *competitive difference* of the algorithm is

$$\max_{v\in[v_l,v_u]}\{ECM(t_s = t_s^{on}) - ECM(t_s = t_s^{off})\}.$$

$ECM(t_s = t_s^{on})$ represents *ECM* evaluated when t_s equals t_s^{on}; the value of t_s^{on} is the same no matter what the value of v is. $ECM(t_s = t_s^{off})$ represents *ECM* evaluated when t_s equals t_s^{off}, the value that minimizes *ECM*; the value of t_s^{off} varies with v. Thus we are interested in the difference between the performance of the on-line algorithm and the performance of the optimal off-line algorithm.

An on-line algorithm is *optimal* if, for every v_l, v_u, p and f, it computes a value for t_s that minimizes the competitive difference.

For completeness, we also define the *competitive ratio* of an algorithm to be

$$\max_{v\in[v_l,v_u]}\left\{\frac{ECM(t_s = t_s^{on})}{ECM(t_s = t_s^{off})}\right\}.$$

3 Optimal off-line algorithm

Let A be the off-line algorithm that uses the following quantity for t_s:

$$t_s^{off} = \sqrt{\frac{v(1-f)(1-p)}{fp}}.$$

In order to match our model assumptions, t_s^{off} must be an integer; for simplicity, we approximate the discrete case with the continuous. We also assume that f and p are such that t_s^{off} lies in the range $[1, v]$; if instead the above square root evaluates to less than 1, then 1 is used for t_s^{off}, and if it evaluates to more than v, then v is used for t_s^{off}. The reason is that the nature of the problem makes it clear that the extreme choices for t_s^{off} are 1 (send every tick) and v (wait until it is definite that the packet has been lost).

We now prove that A is optimal.

Theorem 2 *ECM is minimized when $t_s = t_s^{off}$.*

Proof: First we differentiate *ECM* with respect to t_s.

$$\frac{d}{dt_s}ECM = \frac{fp}{1-p} - v(1-f)\frac{1}{t_s^2}.$$

Setting this equal to 0 and solving for t_s produces t_s^{off}.

To check that this is indeed a minimum, we take the second derivative of *ECM* with respect to t_s:

$$\frac{d^2}{dt_s^2}ECM = \frac{2v(1-f)}{t_s^3}.$$

Clearly substituting t_s^{off} into this expression produces a positive quantity. ■

Evaluating *ECM* at t_s^{off} produces the minimum value of *ECM*, given by the following corollary.

Corollary 3 *The minimum value of ECM is*

$$ECM(t_s = t_s^{off}) = 2\sqrt{\frac{fpv(1-f)}{1-p}} + \frac{p(1-f)}{1-p} + fv.$$

We can make the following observations about t_s^{off}.

- The optimal retransmission timeout for a fixed p and f is $O(\sqrt{v})$.

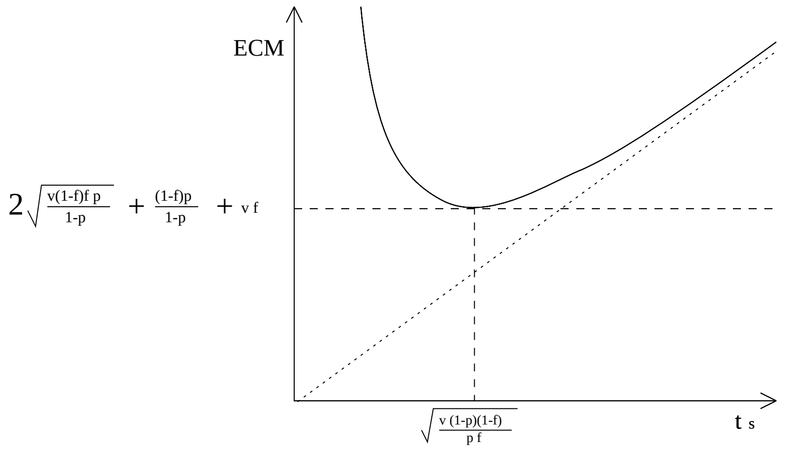

Figure 1: Optimal off-line retransmission timeout

- When the probability p of losing a packet is high, then the value of t_s^{off} is small. This implies that when most packets are lost, it is best to retransmit the packet very frequently.
- When the probability p of losing a packet is low, then the value of t_s^{off} is high. This implies that since packets are rarely lost, it is best to wait a "long time" because most likely the acknowledgment will arrive.
- When the fraction f is close to 1, implying that the cost measure should be time optimal (and no congestion problem exists), the value of t_s^{off} is small. Since the goal is to be time optimal, it is best to retransmit the packet very frequently.
- When the fraction f is close to 0, implying that the cost measure should be packet optimal (and/or a congestion problem exists), the value of t_s^{off} is high. Since the goal is to be packet optimal, it is best to wait "a long time" to avoid sending unnecessary packets.

4 On-line algorithm

Recall that an on-line algorithm can only use knowledge of v_l and v_u, between which v lies, instead of knowing v exactly. We are interested in finding a choice for t_s that will minimize the competitive difference, no matter where in the interval $[v_l, v_u]$ v actually lies.

Let B be the on-line algorithm that uses the following value for t_s:

$$t_s^{on} = \frac{1}{2}(\sqrt{v_l} + \sqrt{v_u})\sqrt{\frac{(1-f)(1-p)}{fp}}.$$

This is similar to t_s^{off}, but with $\sqrt{v}$ replaced by $\frac{1}{2}(\sqrt{v_l} + \sqrt{v_u})$.

Analogously to the off-line algorithm, we only consider values of t_s^{on} that lie between 1 and v_u.

We next show that this algorithm is optimal, i.e., that it minimizes the competitive difference. We proceed as follows. First we define a function *DIFF* of t_s which is the difference between the value of *ECM* for t_s and the optimal value of *ECM*. Then we show that *DIFF* is maximized when v is at one of the endpoints of its range (either v_l or v_u). Then we observe that in order to minimize this maximum difference, we should choose t_s so that the differences at the two endpoints are equal. Finally we check that t_s^{on} achieves this.

Let *DIFF* be defined as follows (v_l, v_u, f, and p are fixed, while t_s and v can vary):

$$DIFF = ECM - ECM(t_s = t_s^{off})$$

Theorem 4 *For a fixed t_s, within the range $[v_l, v_u]$ DIFF is maximized when $v = v_l$ or $v = v_u$.*

Proof: Using calculus shows that the only critical point of *DIFF* is $v = \frac{fpt_s^2}{(1-f)(1-p)}$, but that this is a minimum, not a maximum. Thus, over a finite range, *DIFF* is maximized at one of the two endpoints. The details follow.

Theorem 1 and Corollary 3 imply

$$DIFF = \left(\frac{fpt_s}{1-p} + \frac{v(1-f)}{t_s} + \frac{p(1-f)}{1-p} + fv\right)$$
$$- \left(2\sqrt{\frac{fpv(1-f)}{1-p}} + \frac{p(1-f)}{1-p} + fv\right).$$

Canceling gives

$$DIFF = \frac{fpt_s}{1-p} + \frac{v(1-f)}{t_s} - 2\sqrt{\frac{fpv(1-f)}{1-p}}.$$

Differentiating with respect to v gives

$$\frac{d}{dv}DIFF = \frac{1-f}{t_s} - \sqrt{\frac{fp(1-f)}{v(1-p)}}.$$

Setting this equal to 0 and solving for v gives

$$v_0 = \frac{fpt_s^2}{(1-f)(1-p)}.$$

To determine whether v_0 is a minimum or maximum, we take the second derivative:

$$\frac{d^2}{dv^2}DIFF = \frac{1}{2} \cdot \frac{1}{v\sqrt{v}}\sqrt{\frac{fp(1-f)}{1-p}}.$$

Clearly the second derivative is greater than zero when evaluated at v_0, and hence v_0 is a minimum. ■

In order to minimize the maximum value of *DIFF*, t_s should be chosen so that the values of *DIFF* at the two endpoints, v_l and v_u, are equal. (*ECM* is a linear function of v in the on-line case, *ECM* is $O(v + \sqrt{v})$ in the off-line case, and the two are tangent at the value of v that minimizes *DIFF*.)

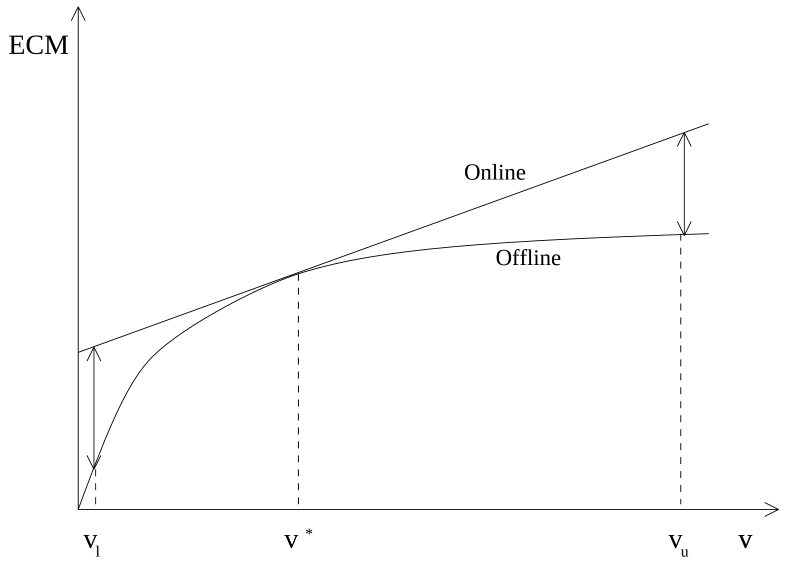

Figure 2: Optimal on-line retransmission timeout

The next theorem shows that t_s^{on} does this.

Theorem 5 $DIFF(t_s = t_s^{on}, v = v_l) = DIFF(t_s = t_s^{on}, v = v_u)$.

Proof: $DIFF(v = v_l) = DIFF(v = v_u)$ implies

$$\left(\frac{fpt_s}{1-p} + \frac{v_l(1-f)}{t_s} + \frac{p(1-f)}{1-p} + fv_l\right)$$
$$-\left(2\sqrt{\frac{fpv_l(1-f)}{1-p}} + \frac{p(1-f)}{1-p} + fv_l\right)$$
$$= \left(\frac{fpt_s}{1-p} + \frac{v_u(1-f)}{t_s} + \frac{p(1-f)}{1-p} + fv_u\right)$$

$$-\left(2\sqrt{\frac{fpv_u(1-f)}{1-p}}+\frac{p(1-f)}{1-p}+fv_u\right).$$

Canceling and rearranging gives

$$\frac{v_l(1-f)-v_u(1-f)}{t_s}=2\left(\sqrt{v_l}-\sqrt{v_u}\right)\sqrt{\frac{fp(1-f)}{1-p}}.$$

Solving for t_s gives t_s^{on}. ■

The next theorem states the competitive difference of the on-line algorithm, namely, the maximum difference between the on-line algorithm and the optimal off-line algorithm. By the previous argument, the maximum difference is found by evaluating *DIFF* with t_s equal to t_s^{on} and v equal to v_l or v_u.

Theorem 6 *The competitive difference for the on-line algorithm is*

$$\left(\frac{1}{2}\frac{\left(\sqrt{v_u}-\sqrt{v_l}\right)^2}{\sqrt{v_u}+\sqrt{v_l}}\right)\sqrt{\frac{fp(1-f)}{1-p}}.$$

Proof: $DIFF(t_s=t_s^{on}, v=v_l)$ equals

$$ECM(t_s=t_s^{on}, v=v_l)-ECM(t_s=t_s^{off}, v=v_l)$$

$$=\left(\frac{fpt_s^{on}}{1-p}+\frac{v_l(1-f)}{t_s^{on}}+\frac{p(1-f)}{1-p}+fv_l\right)$$
$$-\left(2\sqrt{\frac{fpv_l(1-f)}{1-p}}+\frac{p(1-f)}{1-p}+fv_l\right).$$

This simplifies to the required quantity. ■

This quantity is the maximum difference between the on-line algorithm B and the optimal off-line algorithm. We can make the following observations about this difference.

- When the probability of losing a packet is high, the value of $DIFF$ is large, so the on-line algorithm's performance is much less than optimal.
- When the probability of losing a packet is low, the value of $DIFF$ is small, so the on-line algorithm's performance is close to optimal.
- When the fraction f is close to one (i.e., the emphasis is on time and/or no congestion problem exists) the value of $DIFF$ is small, so the on-line algorithm's performance is close to optimal.
- When the fraction f is close to zero (i.e., the emphasis is on number of packets and/or to avoid congestion) the value of $DIFF$ is small, so the on-line algorithm's performance is close to optimal.

4.1 Competitive ratio analysis

We have previously shown that algorithm B is optimal (with respect to the competitive difference). We now analyze the competitive ratio of B. This is done by studying the ratio between the performance of the on-line algorithm and the performance of the off-line algorithm when $v = v_l$. We have previously shown that the difference is maximized and equal at the endpoints v_l and v_u. Since the functions are increasing, the ratio is maximized at the smaller endpoint.

Theorem 7 *The competitive ratio for algorithm B is at most $\frac{1}{2}\sqrt{\frac{v_u}{v_l}} + 1.5$.*

Proof:

$$\frac{ECM(t_s = t_s^{on}, v = v_l)}{ECM(t_s = t_s^{off}, v = v_l)}$$

simplifies to an expression of the form

$$\frac{a \cdot b + c}{d \cdot b + c}$$

where

$$a = \frac{1}{2}(\sqrt{v_l} + \sqrt{v_u}) + \frac{2v_l}{\sqrt{v_l} + \sqrt{v_u}}, b = \sqrt{\frac{fp(1-f)}{1-p}}, c = \frac{p(1-f)}{1-p} + fv_l, d = 2\sqrt{v_l}.$$

We can upper bound this expression:

$$\frac{a \cdot b + c}{d \cdot b + c} = \frac{a}{d + c/b} + \frac{c/b}{d + c/b} \le \frac{a}{d + c/b} + 1 \le \frac{a}{d} + 1.$$

The expression $\frac{a}{d}$ simplies to

$$\frac{\sqrt{v_l} + \sqrt{v_u}}{4\sqrt{v_l}} + \frac{\sqrt{v_l}}{\sqrt{v_l} + \sqrt{v_u}}.$$

The first term is at most $\frac{1}{2} \cdot \sqrt{\frac{v_u}{v_l}}$. The second term is at most $\frac{1}{2}$. Thus the total is at most $\frac{1}{2} \cdot \sqrt{\frac{v_u}{v_l}} + 1.5$. ■

5 Conclusion

We have presented a first step toward a theoretical study of the choice of retransmission timeout, based on competitive analysis. We have presented definitions for an appropriate cost measure, which combines number of packets and elapsed time in a tunable way. We have made choices for what information is available to on-line and off-line algorithms respectively. We described an off-line algorithm and showed that it is optimal. We presented an on-line algorithm and proved that it is optimal with respect to the competitive difference. For completeness, we also analyzed the on-line algorithm with respect to the competitive ratio.

Many interesting questions remain. This paper has studied one part of the overall problem of reliable message transfer in communication networks. An important generalization of this work is to handle the case of transmitting a sequence of messages. Since this can take place over a long period of time, generally p and v will change. Consequently, the choice of t_s should change dynamically. There is also potential feedback between lessening t_s (i.e., sending packets more frequently) and increasing p (i.e., the probability of packets being lost). This feedback could perhaps be modeled by changing f in response to observed changes in v_l, v_u, and p. Adaptive learning techniques could be used to compute current estimates for the parameters f, v_l, v_u, and p; then, for the estimated parameters our on-line algorithm performs optimally.

Acknowledgments: Many thanks to Shmuel Aharon and Zohar Yaakov for simulating the results.

References

[1] James Aspnes, Yassi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts, "On-Line Load Balancing with Applications to Machine Scheduling and Virtual Circuit Routing," *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing,* pp. 623–631, May 1993.

[2] Yair Bartal, Amos Fiat, and Yuval Rabani, "Competitive Algorithms for Distributed Data Management," *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing,* pp. 39–50, May 1992.

[3] A. Borodin, N. Linial, and M. Saks, "An Optimal Online Algorithm for Metrical Task Systems," *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing,* pp. 373-382, 1987.

[4] K. Bartlett, R. Scantlebury, and P. Wilkinson, "A Note on Reliable Full-Duplex Transmission over Half-Duplex Links," *Communications of the ACM,* 12(5):260-261, May 1969.

[5] Douglas Comer, *Internetworking with TCP/IP, Volume I: Principles, Protocols and Architecture,* Prentice-Hall, Englewood Cliffs, NJ, 1991.

[6] David Feldmeier and Ernst Biersack, "Comparison of Error Control Protocols for High Bandwidth-Delay Product Networks," *Proceeding of the IFIP Workshop on Protocols for High Speed Networks,* 1990.

[7] Michael R. Garey and David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W. H. Freeman and Company, New York, 1979.

[8] Raj Jain, "Divergence of Timeout Algorithms for Packet Retransmissions," *Fifth Annual International Phoenix Conference on Computers and Communications,* pages 174–179, March 1986.

[9] Mark Manasse, Lyle McGeoch, and Daniel Sleator, "Competitive Algorithms for On-line Problems," *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing*, pp. 323–333, May 1988.

[10] Daniel D. Sleator and Robert E. Tarjan, "Amortized Efficiency of List Update and Paging Rules," *Communications of the ACM*, 28(2):202–208, 1985.

[11] Andrew Tanenbaum, *Computer Networks,* Prentice Hall, Englewood Cliffs, NJ, 1988.

[12] Lixia Zhang, "Why TCP Timers Don't Work Well," *Proceedings of the ACM SIGCOMM: Communications, Architectures, and Protocols*, pp. 397–409, August 1986.