Dynamic TCP Acknowledgement: Penalizing Long Delays

Susanne Albers[*] Helge Bals[†]

Abstract

We study the problem of acknowledging a sequence of data packets that are sent across a TCP connection. Previous work on the problem has focused mostly on the objective function that minimizes the sum of the number of acknowledgements sent and the delays incurred for all of the packets. Dooly, Goldman and Scott presented a deterministic 2-competitive online algorithm and showed that this is the best competitiveness of a deterministic strategy. Recently Karlin, Kenyon and Randall developed a randomized online algorithm that achieves an optimal competitive ratio of $e/(e-1) \approx 1.58$.

In this paper we investigate a new objective function that minimizes the sum of the number of acknowledgements sent and the *maximum delay* incurred for any of the packets. This function is especially interesting if a TCP connection is used for interactive data transfer between network nodes. The TCP acknowledgement problem with this new objective function is different in structure than the problem with the function considered previously. We develop a deterministic online algorithm that achieves a competitive ratio of $\pi^2/6 \approx 1.644$ and prove that no deterministic algorithm can have a smaller competitiveness. We also study a generalized objective function where delays are taken to the p-th power, for some positive integer p. Again we give tight upper and lower bounds on the best possible competitive ratio of deterministic online algorithms. The competitiveness is 1 plus an alternating sum of Riemann's zeta function and tends to 1.5 as $p \to \infty$. Finally we consider randomized online algorithms and show that, for our first objective function, no randomized strategy can achieve a competitive ratio smaller than $3/(3-2/e) \approx 1.324$. For the generalized objective function we show a lower bound of $2/(2-1/e) \approx 1.225$.

1 Introduction

Dooly et al. [2, 3] recently initiated the algorithmic study of the *dynamic TCP acknowledgement problem.* In large networks such as the Internet data transmission is performed using the Transmission Control Protocol (TCP). Consider an open TCP connection between two network nodes that wish to exchange data. The data is partitioned into segments or *packets* that are sent across the connection. A node receiving data must acknowledge the arrival of each incoming packet so that the sending node is notified that the transmission was successful; lost packets must be retransmitted. However, data packets do not have to be acknowledged individually. Instead, most TCP implementations employ some delay mechanism that allows the TCP to acknowledge multiple incoming packets with a single acknowledgement and, possibly, to piggyback the acknowledgement on an outgoing data segment. Reducing the number of acknowledgements has several advantages, e.g. the overhead incurred at the network nodes for sending and receiving acknowledgements is reduced and, more importantly, the network congestion is reduced. On the other hand, by reducing the number of acknowledgements, one adds latency to a TCP connection, which is not desirable. The goal is to balance the reduction in the number of acknowledgements with the increase in latency. The decision when to send acknowledgements must usually be made *online*, i.e. without knowledge of the future packet arrival times.

Motivated by the fact that TCP supports dynamic acknowledgement mechanisms, Dooly et al. [2, 3] formulated the following problem. A network node receives a sequence of n data packets. Let a_i denote the arrival time of packet i, $1 \le i \le n$. At time a_i, the arrival times a_j, $j > i$, are not known. We have to partition the sequence $\sigma = (a_1, \dots, a_n)$ of packet arrival times into m subsequences $\sigma_1, \dots, \sigma_m$, for some $m \ge 1$, such that each subsequence ends with an acknowledgement. We use σ_i to denote the set of arrivals in the partition. Let t_i be the time when the acknowledgement for σ_i is sent. We require $t_i \ge a_j$, for all $a_j \in \sigma_i$. If data packets are not acknowledged immediately, there are *acknowledgement delays*. Note that any reasonable objective function must take into account both the number of acknowledgements sent and the incurred acknowledgement delays. Ignoring the number of acknowledgements and considering only delays, it would be optimal to acknowledge each packet immediately, which leads to a large number of acknowledgements sent. On the other hand, ignoring delays and considering only acknowledgements, it would be best to send a single acknowledgement at the end of the packet sequence, which leads to unacceptable delays.

Previous results: Previous work on the dynamic TCP acknowledgement problem [2, 3, 4, 5, 6, 7] has focused mostly on the objective function that minimizes the number of acknowledgements and the sum of the

[*]Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Georges-Köhler-Allee 79, 79110 Freiburg, Germany. `salbers@informatik.uni-freiburg.de` Work supported in part by the EU, projects APPOL and APPOL II.

[†]Lehrstuhl Informatik II, Universität Dortmund, 44221 Dortmund, Germany.

delays incurred for all of the packets, i.e. we wish to minimize $h = m + \sum_{i=1}^{m} \sum_{a_j \in \sigma_i} (t_i - a_j)$. Given a solution generated by an acknowledgement algorithm A on input σ, the resulting objective function value is also referred to as the *cost* $C_A(\sigma)$ of A on σ. Following [8], an online algorithm A is called *c-competitive* if there exists a constant b such that $C_A(\sigma) \leq c \cdot C_{OPT}(\sigma) + b$, for all inputs σ. Here $C_{OPT}(\sigma)$ is the cost incurred by an optimal offline algorithm that knows the entire input σ in advance and can serve it with minimum cost.

Dooly et al. [2, 3] presented a deterministic 2-competitive online algorithm and showed that no deterministic online strategy can achieve a smaller competitive ratio. This performance guarantee also holds if an online algorithm has some bounded lookahead. Most implementations of TCP have a *maximum delay constraint*, i.e. the acknowledgement of a packet may be delayed for at most δ time units, e.g. δ could be 500 ms. Dooly et al. showed that their algorithm can be modified and remains 2-competitive in the presence of such a constraint. Karlin et al. [4] studied randomized online algorithms against oblivious adversaries. They developed a randomized online strategy that achieves a competitiveness of $e/(e-1) \approx 1.58$. Noga [5] and independently Seiden [7] showed that no randomized algorithm can do better.

Dooly et al. also studied the minimization of a second objective function $h' = m + \sum_{i=1}^{m} \max_{a_j \in \sigma_i} (t_i - a_j)$ where one considers the sum of the maximum delays incurred in subsequences σ_i in addition to the number of acknowledgements sent. They showed that the best competitive ratio of a deterministic online algorithm without lookahead is equal to 2.

In general, Dooly et al. and Karlin et al. pointed out that the TCP acknowledgement problem with objective functions h and h' are ski rental type problems.

Our contribution: In this paper we investigate a new family of objective functions that penalize long acknowledgement delays of individual data packets more heavily. TCP is used for both interactive and bulk data transfer. In the first case, consider a TCP connection that is used for communication with a remote interactive program. Here long delays are not acceptable as they are noticeable to the user. In the case of bulk data transfer long delays also have a negative effect and hence, as already mentioned before, most systems work with a maximum delay constraint. Therefore it is desirable to design algorithms that aim at keeping the maximum delay short.

We study the objective function that minimizes the number of acknowledgements and the maximum delay incurred for any of the data packets. Given an input σ, consider a partitioning $\sigma_1, \ldots, \sigma_m$. Let $d_i = \max_{a_j \in \sigma_i} (t_i - a_j)$ be the maximum delay of any packet in σ_i, $1 \leq i \leq m$. We wish to minimize the function

$$f = m + \max_{1 \leq i \leq m} d_i. \tag{1.1}$$

It turns out that the dynamic TCP acknowledgement problem with objective function f is different in structure than the problem with functions h or h'. In particular our problem is not a ski rental problem. In Section 2 we present a family of deterministic online algorithms and prove that the best strategy in that family achieves a competitive ratio of $\pi^2/6 \approx 1.644$. Note that $\pi^2/6 = \sum_{i=1}^{\infty} 1/i^2$. We also show that this is the best possible competitive ratio. No deterministic online algorithm can achieve a competitiveness smaller than $\pi^2/6$. Additionally, we investigate a generalization of the objective function f where delays are taken to the p-th power and hence are penalized even more heavily. For any integer $p \geq 1$, we wish to minimize

$$f_p = m + \max_{1 \leq i \leq m} d_i^p. \tag{1.2}$$

For the formulation of the competitive ratio, let $\zeta(p) = \sum_{i=1}^{\infty} \frac{1}{i^p}$, for any $p \geq 2$. The function $\zeta(p)$ is known as the Riemann zeta function. We define $\zeta(1) := 1$. Let

$$c_p = 1 + \sum_{q=1}^{p+1} (-1)^{p+1-q} \zeta(q).$$

In Section 3 we give a deterministic online algorithm that is c_p-competitive and prove that no deterministic strategy can achieve a competitiveness smaller than c_p. For $p = 1$, this expression is equal to $\pi^2/6$. In general c_p is decreasing in p and tends to 1.5 as $p \to \infty$.

In Section 4 we consider randomized online algorithms against oblivious adversaries and present lower bounds. We first prove that, given function f, no randomized online algorithm achieves a competitive ratio smaller than $3/(3-2/e) \approx 1.324$. We then show a lower bound of $2/(2-1/e) \approx 1.225$ for function f_p, $p \geq 2$.

We remark that similar to [2, 3], we could consider in f and f_p a linear combination of the number of number of acknowledgements sent and the maximum delay (taken to the p-th power). This does not change the competitive ratios and the upper and lower bound proofs can be modified easily. For simplicity, in this extended abstract we study the functions as defined in (1.1) and (1.2). Moreover we remark that all of our results carry over to the case that there is a maximum delay constraint.

2 Minimizing the maximum delay

2.1 An optimal deterministic online algorithm

We define a general class of algorithms. Let z be a positive real number.

Algorithm Linear-Delay(z): Initially, set $d = z$ and send the first acknowledgement at time $a_1 + d$. In general, suppose that the i-th acknowledgement has just been sent and that j packets have been processed so far. Set $d = (i+1)z$ and send the $(i+1)$-st acknowledgement at time $a_{j+1} + d$.

We analyze the algorithm for values z with $z \geq 1/2$, which give the best performance.

THEOREM 2.1. *For any z with $z \geq 1/2$, Linear-Delay(z) is c-competitive, where $c = \max\{1 + z, (1 + z)/(2 + z - \pi^2/6)\}$.*

COROLLARY 2.1. *Setting $z = \pi^2/6 - 1$, Linear-Delay(z) achieves a competitive ratio of $\pi^2/6$.*

We now prove Theorem 2.1.

Proof. In the following we call the online algorithm LD(z) for short. Suppose that LD(z) serves the input sequence using m acknowledgements. The longest acknowledgement delay is mz and hence the online cost is $C_{LD(z)}(\sigma) = m(1 + z)$.

We have to lower bound the cost incurred by an optimal offline algorithm OPT. In the sequence of n packets we identify a subsequence of m *main packets*, numbered from 0 to $m - 1$. Main packet 0 is the first packet in the input sequence. Main packet i, $1 \leq i \leq m - 1$, is the first packet that arrives after the i-th acknowledgement sent by LD(z), i.e. it is the first packet that arrives after time t_i. The definition of LD(z) implies that the time difference between the $(i-1)$-st and the i-th main packets is larger than iz, for $i = 1, \ldots, m - 1$.

Suppose that the optimum offline algorithm serves the request sequence using l acknowledgements and that the maximum acknowledgement delay is C, $C \geq 0$. Then $C_{OPT}(\sigma) = l + C$. Associated with each acknowledgement α sent by OPT is an *acknowledgement interval* that starts when the first packet acknowledged by α arrives and ends when α is sent. The length of each interval is bounded by C. In the following i always denotes a positive integer.

LEMMA 2.1. *Any acknowledgement interval starting at or after the arrival of main packet $\lfloor \frac{C}{iz} \rfloor$ can contain at most i main packets.*

Proof. Main packet k with $k \geq \lfloor \frac{C}{iz} \rfloor + 1$ has a distance of more than $z(\lfloor \frac{C}{iz} \rfloor + 1)$ to the previous main packet. If the acknowledgement interval contained at least $i + 1$ main packets, then the length of the interval would be at least $iz(\lfloor \frac{C}{iz} \rfloor + 1) > iz(\frac{C}{iz}) = C$, which is impossible. □

Define $i_0 = \lfloor \sqrt[3]{\frac{C}{z}} \rfloor - 1$. In the rest of this proof we assume $i_0 \geq 2$. If $i_0 \leq 1$, then $C \leq 27z$ and OPT must acknowledge each of the last $m - 27z$ main packets with separate acknowledgements. In this case LD(z) is clearly $(1 + z)$-competitive.

LEMMA 2.2. *Let $1 \leq i \leq i_0$. The acknowledgement interval containing main packet k, for $k \geq \lfloor \frac{C}{iz} \rfloor$, must have started after the arrival of main packet $\lfloor \frac{C}{(i+1)z} \rfloor$.*

Proof. We show that the time window starting at main packet $\lfloor \frac{C}{(i+1)z} \rfloor$ and ending with main packet $\lfloor \frac{C}{iz} \rfloor$ is larger than C, which proves the lemma. The number of main packets in this time window is $\lfloor \frac{C}{iz} \rfloor - \lfloor \frac{C}{(i+1)z} \rfloor + 1 > \frac{C}{i(i+1)z} \geq i + 2$. The last inequality is equivalent to $C/z \geq i(i + 1)(i + 2)$ and holds for all $i \leq i_0$. Thus there are at least $i + 2$ main packets in this time window. Each of the last $i + 1$ of these is more than $z(\lfloor \frac{C}{(i+1)z} \rfloor + 1)$ time units away from the previous main packet and thus the length of the window is greater than $(i + 1)z(\lfloor \frac{C}{(i+1)z} \rfloor + 1) > (i + 1)z(\frac{C}{(i+1)z}) = C$. □

We now estimate the number of acknowledgements sent by OPT and use the following charging scheme. An acknowledgement costs 1. We charge this cost to the main packets contained in the associated acknowledgement interval and split the cost evenly among these main packets. More specifically, if an acknowledgement interval contains $i \geq 1$ main packets, then each of these packets is assigned a cost of $1/i$. If an acknowledgement interval does not contain a main packet, then we ignore it in the analysis of OPT's cost. We develop a lower bound on the cost charged to each main packet. Summing over all main packets, we derive a lower bound on the optimum cost incurred for sending acknowledgements.

We assume that $C < m$. If $C \geq m$, then LD(z) is clearly $(1 + z)$-competitive because LD(z)'s cost is $(1 + z)m$ and the optimum offline cost is at least m. In the following we will first analyze the case $C \leq zm$ and then $C > zm$.

Suppose that $C \leq zm$. Each main packet is contained in some acknowledgement interval. Let i be an integer with $1 \leq i \leq i_0$. We analyze the cost charged to main packet k with $k \geq \lfloor \frac{C}{iz} \rfloor$ and $k < \lfloor \frac{C}{(i-1)z} \rfloor$. If $i = 1$, then $k < m$. If the acknowledgement interval containing main packet k started at or after the arrival of main packet $\lfloor \frac{C}{iz} \rfloor$, then by Lemma 2.1 at most i main packets are contained in the interval and main packet k is assigned a cost of at least $1/i$. If the acknowledgement interval started earlier, then by Lemma 2.2 it must have started after the arrival of main packet $\lfloor \frac{C}{(i+1)z} \rfloor$. Applying Lemma 2.1 for $i+1$, we obtain that the interval contains at most $i+1$ main packets and the packet is assigned a cost of at least $1/(i+1)$. There is only one acknowledgement interval that starts before and ends after the arrival of main packet $\lfloor \frac{C}{iz} \rfloor$. Thus for at most $i+1$ main packets considered above, the cost is lower bounded by $1/(i+1)$ instead of $1/i$. We obtain that for $i = 1$, the total cost assigned to all the main packets k with $k \ge \lfloor \frac{C}{z} \rfloor$ is

$$(m - \lfloor \tfrac{C}{z} \rfloor) - 2(1 - \tfrac{1}{2}) = (m - \lfloor \tfrac{C}{z} \rfloor) - 1.$$

For $2 \le i \le i_0$, the total cost assigned to main packet k, $\lfloor \frac{C}{iz} \rfloor \le k < \lfloor \frac{C}{(i-1)z} \rfloor$ is at least

$$\begin{aligned}&(\lfloor \tfrac{C}{(i-1)z} \rfloor - \lfloor \tfrac{C}{iz} \rfloor)\tfrac{1}{i} - (i+1)(\tfrac{1}{i} - \tfrac{1}{i+1})\\ &= \quad (\lfloor \tfrac{C}{(i-1)z} \rfloor - \lfloor \tfrac{C}{iz} \rfloor)\tfrac{1}{i} - \tfrac{1}{i}.\end{aligned}$$

Summing over all i, we obtain that the number of acknowledgements sent by OPT is at least

$$\begin{aligned} l \quad &\ge \quad m - \lfloor \tfrac{C}{z} \rfloor - 1 + \sum_{i=2}^{i_0} \left(\left(\left\lfloor \tfrac{C}{(i-1)z} \right\rfloor - \left\lfloor \tfrac{C}{iz} \right\rfloor \right) \tfrac{1}{i} - \tfrac{1}{i} \right) \\ &= \quad m - \sum_{i=1}^{i_0-1} \lfloor \tfrac{C}{iz} \rfloor \left(\tfrac{1}{i} - \tfrac{1}{i+1} \right) - \left\lfloor \tfrac{C}{i_0 z} \right\rfloor \tfrac{1}{i_0} - H_{i_0}. \end{aligned}$$

Here H_{i_0} denotes the i_0-th Harmonic number. Thus

$$\begin{aligned} l \quad &\ge \quad m - \tfrac{C}{z} \sum_{i=1}^{\infty} \left(\tfrac{1}{i^2} - \tfrac{1}{i(i+1)} \right) - \tfrac{C}{i_0^2 z} - i_0 \\ &\ge \quad m - \tfrac{C}{z} \left(\tfrac{\pi^2}{6} - 1 \right) - \tfrac{C}{i_0^2 z} - i_0 \\ &\ge \quad m - \tfrac{C}{z} \left(\tfrac{\pi^2}{6} - 1 \right) - 10 \sqrt[3]{\tfrac{C}{z}} \\ &\ge \quad m - \tfrac{C}{z} \left(\tfrac{\pi^2}{6} - 1 \right) - 10 \sqrt[3]{\tfrac{m}{z}}. \end{aligned}$$

The second to last inequality follows because $i_0 \ge 2$ and hence $i_0^2 \ge \frac{1}{9}(\frac{C}{z})^{2/3}$. The last inequality follows because $C \le m$ and hence $\sqrt[3]{\frac{C}{z}} \le \sqrt[3]{\frac{m}{z}}$. The total cost incurred by OPT is at least

$$C_{OPT}(\sigma) = l + C \ge m + C - \tfrac{C}{z} \left(\tfrac{\pi^2}{6} - 1 \right) - O\left(\sqrt[3]{m} \right). \tag{2.3}$$

We now distinguish two cases. If $z > \frac{\pi^2}{6} - 1$, then the right hand side of (2.3) is increasing in C. Choosing $C = 0$ we obtain that $C_{OPT}(\sigma) \ge m - O(\sqrt[3]{m})$ and LD(z) is $(1+z)$-competitive because the online cost is $(1+z)m$. If $z \le \frac{\pi^2}{6} - 1$, then the right hand side of (2.3) is decreasing in C. Choosing the largest possible value $C = zm$, we obtain $C_{OPT}(\sigma) \ge (2 + z - \frac{\pi^2}{6})m - O(\sqrt[3]{m})$ and LD(z) achieves a competitive ratio of $(1+z)/(2+z-\frac{\pi^2}{6})$.

We next analyze the case $C > zm$. The only difference in analyzing this case is that there are no main packets k with $k \ge \lfloor \frac{C}{z} \rfloor$ and $k < m$ because C is large. However, there are main packets k with $k \ge \lfloor \frac{C}{2z} \rfloor$ and $k < m$ because $C < m \le 2zm$ since $z \ge 1/2$. Thus the number of acknowledgements sent by OPT is

$$\begin{aligned} l \quad &\ge \quad (m - \lfloor \tfrac{C}{2z} \rfloor) \tfrac{1}{2} - \tfrac{1}{2} \\ &\quad + \sum_{i=3}^{i_0} \left(\left(\left\lfloor \tfrac{C}{(i-1)z} \right\rfloor - \lfloor \tfrac{C}{iz} \rfloor \right) \tfrac{1}{i} - \tfrac{1}{i} \right) \\ &\ge \quad \tfrac{1}{2}m - \sum_{i=2}^{i_0-1} \tfrac{C}{iz} \left(\tfrac{1}{i} - \tfrac{1}{i+1} \right) - \tfrac{C}{i_0^2 z} - (H_{i_0} - 1) \\ &\ge \quad \tfrac{1}{2}m - \tfrac{C}{z} \left(\tfrac{\pi^2}{6} - 1.5 \right) - 10 \sqrt[3]{\tfrac{m}{z}}. \end{aligned}$$

Thus the optimum cost is at least $C_{OPT}(\sigma) \ge \frac{1}{2}m + C - \frac{C}{z}(\frac{\pi^2}{6} - 1.5) - O(\sqrt[3]{m})$. The right hand side of the last inequality is increasing in C because $z \ge 1/2 > (\frac{\pi^2}{6} - 1.5)$. Since $C > zm$, we obtain $C_{OPT}(\sigma) \ge (2 + z - \frac{\pi^2}{6})m - O(\sqrt[3]{m})$ and LD(z) achieves a competitive ratio of $(1+z)/(2+z-\frac{\pi^2}{6})$. □

2.2 Lower bound

THEOREM 2.2. *Let A be a deterministic online algorithm. If A is c-competitive, then $c \ge \frac{\pi^2}{6}$.*

Proof. We construct a family of request sequences σ_l, for any $l \ge 8$. For a fixed l in this range, let $i_0 = \lfloor \sqrt[3]{l} \rfloor - 2$ and $l' = \lfloor \frac{l}{i_0+1} \rfloor$. For convenience we number the packets is σ_l starting with l'. Packet l' is sent at time 0. For any i with $l' < i \le l$, packet i is sent exactly $(\frac{\pi^2}{6} - 1)i$ time units after packet $i-1$. For any i with $i > l$, packet i is sent exactly $(\frac{\pi^2}{6} - 1)l$ time units after packet $i-1$. The adversary stops sending packets as soon as the online algorithm decides to acknowledge an incoming packet together with the preceding packet. If the online algorithm never acknowledges a packet together with a preceding packet, the adversary can force a competitive ratio arbitrarily close to 2 by acknowledging always two packets together. Thus, let m be the number of the last packet sent by the adversary. Note that m is a function of l but, for simplicity, this dependency will not be shown in the notation.

In the following we will first analyze the competitive ratio of the online algorithm for $m \le l$, then we will consider the case $m > l$. If $m \le l$, then the adversary can acknowledge each packet immediately and its cost is $C_{ADV}(\sigma_l) = m - l' + 1$ because the packet numbering in the sequence starts with l'. The online algorithm A serves the first $m - l' - 1$ packets with separate acknowledgements and the last two packets with a joint acknowledgement. The acknowledgement of packet $m-1$ is delayed by $(\frac{\pi^2}{6} - 1)m$ time units. Thus the total online cost is at least $C_A(\sigma_l) = m - l' + (\frac{\pi^2}{6} - 1)m =$

$\frac{\pi^2}{6}m-l' = \frac{\pi^2}{6}(m-\frac{6}{\pi^2}l') \geq \frac{\pi^2}{6}(m-l'+1) = \frac{\pi^2}{6}C_{ADV}(\sigma_l)$. The last inequality holds because $l' \geq \frac{\pi^2}{6}/(\frac{\pi^2}{6}-1)$. To verify this relation we observe that, for $l = 8$, $l' = 8 \geq \frac{\pi^2}{6}/(\frac{\pi^2}{6}-1)$ and l' is increasing in l.

It remains to analyze the case $m > l$. The adversary chooses acknowledgement intervals of length $(\frac{\pi^2}{6}-1)l$, i.e. it sends out an acknowledgement whenever there is an unacknowledged packet waiting for exactly $(\frac{\pi^2}{6}-1)l$ time units. To analyze the number of acknowledgements incurred by the adversary we need the following lemma.

LEMMA 2.3. *Let $1 \leq i \leq i_0$. An acknowledgement interval that ends after the arrival of packet $\lfloor \frac{l}{i} \rfloor$ must have started after the arrival of packet $\lfloor \frac{l}{i+1} \rfloor$.*

Proof. Suppose that an acknowledgement interval ending after the arrival of packet $\lfloor \frac{l}{i} \rfloor$ started at or before the arrival of packet $\lfloor \frac{l}{i+1} \rfloor$. This time interval contains $\lfloor \frac{l}{i} \rfloor - \lfloor \frac{l}{i+1} \rfloor$ packets that are at least $(\frac{\pi^2}{6}-1)(\lfloor \frac{l}{i+1} \rfloor + 1) \geq (\frac{\pi^2}{6}-1)\frac{l}{i+1}$ time units away from the preceding packet. Thus the time interval has a total length of $(\frac{\pi^2}{6}-1)\frac{l}{i+1}(\lfloor \frac{l}{i} \rfloor - \lfloor \frac{l}{i+1} \rfloor) \geq (\frac{\pi^2}{6}-1)\frac{l}{i+1}(\frac{l}{i}-1-\frac{l}{i+1}) = (\frac{\pi^2}{6}-1)\frac{l}{i+1}(\frac{l}{i(i+1)}-1)$. We have $\frac{l}{i(i+1)}-1 > i+1$ because the this inequality is equivalent to $l > i(i+1)(i+2)$, which holds for all $1 \leq i \leq i_0$. Thus the time interval has a total length of greater than $(\frac{\pi^2}{6}-1)l$, contradicting the fact the adversary chooses acknowledgement intervals of length $(\frac{\pi^2}{6}-1)l$. $\square$

To estimate the total number of acknowledgements incurred by the adversary we use a charging scheme similar to that employed in the upper bound. If an acknowledgement interval contains i packets, then the cost of 1 is distributed evenly among the packets, i.e. each packet is assigned a cost of $\frac{1}{i}$. An acknowledgement interval that ends no later than the arrival of packet $\lfloor \frac{l}{i} \rfloor, 1 \leq i \leq i_0$, contains at least $i+1$ packets because each of the packets is a distance of at most $\lfloor \frac{l}{i} \rfloor(\frac{\pi^6}{6}-1)$ away from the preceding packet. Hence packets k with $\lfloor \frac{l}{i+1} \rfloor < k \leq \lfloor \frac{l}{i} \rfloor$ are charged a cost of at most $\frac{1}{i+1}$. However this is not completely correct because a packet k in the latter range may be contained in an acknowledgement interval that ends after the arrival of packet $\lfloor \frac{l}{i} \rfloor$. By the above lemma, such an acknowledgement interval cannot end after the arrival of packet $\lfloor \frac{l}{i-1} \rfloor$, if $i \geq 2$. Thus the packet k is assigned a cost of $\frac{1}{i}$ instead of $\frac{1}{i+1}$. At most $i+1$ packets can have this slightly higher cost because each packet k with $\lfloor \frac{l}{i+1} \rfloor < k \leq \lfloor \frac{l}{i} \rfloor$ has a distance of at least $(\frac{\pi^2}{6}-1)(\lfloor \frac{l}{i+1} \rfloor + 1) > (\frac{\pi^2}{6}-1)\frac{l}{i+1}$ to its preceding packet. For any $1 \leq i \leq i_0$, the total cost charged to all the packets k with $\lfloor \frac{l}{i+1} \rfloor < k \leq \lfloor \frac{l}{i} \rfloor$ is $(\lfloor \frac{l}{i} \rfloor - \lfloor \frac{l}{i+1} \rfloor)\frac{1}{i+1} + (i+1)(\frac{1}{i}-\frac{1}{i+1}) = (\lfloor \frac{l}{i} \rfloor - \lfloor \frac{l}{i+1} \rfloor)\frac{1}{i+1} + \frac{1}{i}$. Any packet k with $l < k \leq m$ is charged a cost of $\frac{1}{2}$ because these packets are a distance of exactly $(\frac{\pi^2}{6}-1)l$ apart. In the worst case, the last packet m is charged a cost of 1. Moreover packet l' is assigned a cost of $\frac{1}{i_0+1}$. In summary, the total cost charged to all of the packets, which is equal to the total number of acknowledgements sent by the adversary, is upper bounded by

$$
\begin{aligned}
& (m-l-1)\tfrac{1}{2} + 1 + \sum_{i=1}^{i_0}\left(\left(\left\lfloor \tfrac{l}{i} \right\rfloor - \left\lfloor \tfrac{l}{i+1} \right\rfloor\right)\tfrac{1}{i+1} + \tfrac{1}{i}\right) \\
& + \tfrac{1}{i_0+1} \\
\leq \; & \tfrac{m}{2} - \tfrac{l}{2} + \tfrac{1}{2} + \sum_{i=1}^{i_0}\left(\tfrac{l}{i} - \tfrac{l}{i+1} + 1\right)\tfrac{1}{i+1} + H_{i_0+1} \\
\leq \; & \tfrac{m}{2} - \tfrac{l}{2} + \sum_{i=1}^{\infty}\tfrac{l}{i(i+1)} - \sum_{i=1}^{\infty}\tfrac{l}{(i+1)^2} + 2H_{i_0+1} \\
= \; & \tfrac{m}{2} + \tfrac{l}{2} - l\left(\tfrac{\pi^2}{6}-1\right) + O(\log l),
\end{aligned}
$$

where H_k is the k-th Harmonic number.

Since the maximum acknowledgement delay incurred by the adversary is $(\frac{\pi^2}{6}-1)l$, its total cost is $\frac{1}{2}(m+l) + O(\log l)$. On the other hand the total cost incurred by the online algorithm A is $m - l' + (\frac{\pi^2}{6}-1)l$ because the input consists of $m-l'+1$ data packets, the last two of which are acknowledged together. We conclude that the ratio of the online cost to the adversary's cost is

$$\frac{\frac{\pi^2}{6}l + m - l - l'}{l + \frac{1}{2}(m-l) + O(\log l)}.$$

Since $l' = o(l)$ and $O(\log l) = o(l)$, this ratio approaches a value of at least $\frac{\pi^2}{6}$ as $l \to \infty$, no matter how the online algorithm chooses $m, m > l$. $\square$

3 Minimizing the maximum delay taken to the p-th power

We first show that c_p is decreasing in p and tends to 1.5 as $p \to \infty$. For $p \geq 1$, let $g(p) = \sum_{i=1}^{\infty} \frac{1}{i^p(i+1)}$. Then, for $p \geq 2$,

$$
\begin{aligned}
g(p) &= \sum_{i=1}^{\infty}\frac{1}{i^p(i+1)} = \sum_{i=1}^{\infty}\frac{1}{i^p} - \sum_{i=1}^{\infty}\frac{1}{i^{p-1}(i+1)} \\
&= \zeta(p) - g(p-1).
\end{aligned}
$$

Applying this recurrence repeatedly we obtain $g(p) = \sum_{q=1}^{p}(-1)^{p-q}\zeta(q)$. Note that $g(1) = 1 = \zeta(1)$. Thus $c_p = 1 + g(p+1)$. We have $g(p+1) = \frac{1}{2} + \sum_{i=2}^{\infty}\frac{1}{i^{p+1}(i+1)}$. The last sum is always positive and tends to 0 as $p \to \infty$. Table 1 shows the value of c_p, for small p.

p	c_p
1	1.6449
2	1.5571
3	1.5252
4	1.5117
5	1.5056
6	1.5027
7	1.5013
8	1.5007
9	1.5003
10	1.5002

Table 1: Some values of c_p

3.1 An optimal deterministic online algorithm

We generalize the algorithm given in Section 2. Let z be a positive real number.

Algorithm Delay(z,p)**:** Set the initial delay to $d = \sqrt[p]{z}$ and send out the first acknowledgement at time $a_1 + d$. In general, assume that i acknowledgements have been sent and that j packets have been processed so far. Set $d = \sqrt[p]{(i+1)z}$ and send the $(i+1)$-st acknowledgement at time $a_{j+1} + d$.

THEOREM 3.1. *Setting $z_p = c_p - 1$, the algorithm Delay(z_p,p) is c_p-competitive.*

Proof. We denote the algorithm by $\mathrm{D}(z_p,p)$ for short. Suppose that the online algorithm serves the input sequence using m acknowledgements. Then its total cost is $C_{D(z_p,p)}(\sigma) = m + (\sqrt[p]{m\,z_p})^p = (1+z_p)m = c_p m$. Let C be the maximum acknowledgement delay incurred by the optimum offline algorithm OPT. If $C > \sqrt[p]{m}$, then the optimum offline cost is at least m and $\mathrm{D}(z_p,p)$ is clearly c_p-competitive. Therefore we assume $C \le \sqrt[p]{m}$.

In analyzing the optimum offline cost we use the terms *main packet* and *acknowledgement interval* as introduced in the proof of Theorem 2.1. Again we number the m main packets in the input from 0 to $m-1$. Let $i_0 = \lfloor \sqrt[2p+1]{C^p/z_p} \rfloor - 1$. In the following we assume $i_0 \ge 4$. If $i_0 \le 3$, then $\sqrt[2p+1]{C^p/z_p} \le 5$, which is equivalent to $C \le \sqrt[p]{5^{2p+1} z_p}$. Thus C is upper bounded by a constant and all but a constant number of the m main packets require a separate acknowledgement by OPT. Thus $\mathrm{D}(z_p,p)$ is c_p-competitive.

In the following we first concentrate on the case $C < \sqrt[p]{z_p m}$, then we consider $C \ge \sqrt[p]{z_p m}$. To estimate the number of acknowledgements sent by OPT, we apply the usual charging scheme. If an acknowledgement interval contains i main packets we charge a cost of $\frac{1}{i}$ to each of these. Using ideas similar to that in the proof of Theorem 2.1 we can show that if an acknowledgement interval starting at or after the arrival of main packet $\lfloor \frac{C^p}{i^p z_p} \rfloor$ can contain at most i main packets. Secondly, an acknowledgement interval containing main packet k, with $k \ge \lfloor \frac{C^p}{i^p z_p} \rfloor$ must have started after packet $\lfloor \frac{C^p}{(i+1)^p z_p} \rfloor$. These two statements imply that the total cost assigned to main packets k with $k \ge \lfloor \frac{C^p}{z_p} \rfloor$ is at least $m - \lfloor \frac{C^p}{z_p} \rfloor - 1$ and the cost assigned to main packets k with $\lfloor \frac{C^p}{i^p z_p} \rfloor \le k < \lfloor \frac{C^p}{(i-1)^p z_p} \rfloor$ and $2 \le i \le i_0$ is at least

$$\left(\left\lfloor \frac{C^p}{(i-1)^p z_p} \right\rfloor - \left\lfloor \frac{C^p}{i^p z_p} \right\rfloor \right) \frac{1}{i} - \frac{1}{i}.$$

Hence the number l of acknowledgements sent by OPT is at least

$$\begin{aligned} l &\ge m - \left\lfloor \frac{C^p}{z_p} \right\rfloor - 1 \\ &\quad + \sum_{i=2}^{i_0} \left(\left(\left\lfloor \frac{C^p}{(i-1)^p z_p} \right\rfloor - \left\lfloor \frac{C^p}{i^p z_p} \right\rfloor \right) \frac{1}{i} - \frac{1}{i} \right) \\ &= m - \sum_{i=1}^{i_0-1} \left\lfloor \frac{C^p}{i^p z_p} \right\rfloor \left(\frac{1}{i} - \frac{1}{i+1} \right) - \left\lfloor \frac{C^p}{i_0^p z_p} \right\rfloor \frac{1}{i} - H_{i_0} \\ &\ge m - \frac{C^p}{z_p} \sum_{i=1}^{\infty} \left(\frac{1}{i^{p+1}} - \frac{1}{i^p(i+1)} \right) - \frac{C^p}{i_0^{p+1} z_p} - H_{i_0} \\ &= m - \frac{C^p}{z_p}(\zeta(p+1) - g(p)) - \frac{C^p}{i_0^{p+1} z_p} - H_{i_0} \\ &= m - \frac{C^p}{z_p} z_p - \frac{C^p}{i_0^{p+1} z_p} - H_{i_0}. \end{aligned}$$

The last equation holds because $\zeta(p+1) - g(p) = g(p+1) = c_p - 1 = z_p$.

LEMMA 3.1. *The term $\frac{C^p}{i_0^{p+1} z_p}$ is $o(m)$.*

Proof. By definition the term equals

$$(3.4)\quad \frac{C^p}{\left(\left\lfloor \sqrt[2p+1]{\frac{C^p}{z_p}} \right\rfloor - 1 \right)^{p+1} z_p} \le \frac{C^p}{\left(\sqrt[2p+1]{\frac{C^p}{z_p}} - 2 \right)^{p+1} z_p},$$

where the inequality holds because the denominator is non-negative by the choice of $i_0 \ge 4$. Moreover, the assumption $i_0 \ge 4$ implies $2 \le i_0/2 \le \sqrt[2p+1]{\frac{C^p}{z_p}}/2$ and hence the last expression in (3.4) can be upper bounded by

$$\begin{aligned} &\frac{C^p}{\left(\sqrt[2p+1]{\frac{C^p}{z_p}} - \sqrt[2p+1]{\frac{C^p}{z_p}}/2 \right)^{p+1} z_p} \\ &= C^{p - \frac{p(p+1)}{2p+1}} z_p^{\frac{p+1}{2p+1} - 1} 2^{p+1} = dC^{p - \frac{p(p+1)}{2p+1}}, \end{aligned}$$

for some constant d. As $C \le \sqrt[p]{m}$ and the exponent $p - \frac{p(p+1)}{2p+1}$ is strictly smaller than p, the term under consideration is $o(m)$. □

Using the above lemma we obtain that the number of acknowledgements sent by OPT is at least $m - C^p - o(m)$ and the total cost is $C_{OPT}(\sigma) \geq m - o(m)$. This implies that $\mathrm{D}(z_p, p)$ is c_p-competitive.

We finally analyze the case $C \geq \sqrt[p]{z_p m}$. Since C is large, there are not necessarily main packets k with $k \geq \lfloor \frac{C^p}{z_p} \rfloor$ and $C < m$. However there are packets $k \geq \lfloor \frac{C^p}{2^p z_p} \rfloor$ and $k < m$ because $C^p/(2^p z_p) < m$ is equivalent to $C < \sqrt[p]{m 2^p z_p}$ and this holds because $C \leq \sqrt[p]{m}$ and $z_p = g(p+1) = \frac{1}{2} + \sum_{i=2}^{\infty} \frac{1}{i^{p+1}(i+1)} > \frac{1}{2}$. Thus the number of acknowledgements l sent by OPT is at least

$$\begin{aligned}
l &\geq \left(m - \left\lfloor \frac{C^p}{2^p z_p} \right\rfloor\right) \frac{1}{2} - \frac{1}{2} \\
&\quad + \sum_{i=3}^{i_0} \left(\left(\left\lfloor \frac{C^p}{(i-1)^p z_p} \right\rfloor - \left\lfloor \frac{C^p}{i^p z_p} \right\rfloor\right) \frac{1}{i} - \frac{1}{i}\right) \\
&\geq \frac{m}{2} - \frac{C^p}{z_p} \sum_{i=2}^{\infty} \left(\frac{1}{i^{p+1}} - \frac{1}{i^p(i+1)}\right) - \frac{C^p}{i_0^{p+1} z_p} - H_{i_0} \\
&= \frac{m}{2} - \frac{C^p}{z_p}(z_p - \tfrac{1}{2}) - o(m).
\end{aligned}$$

We conclude that the optimum offline cost is at least $C_{OPT}(\sigma) \geq \frac{m}{2} + \frac{C^p}{2 z_p} - o(m) \geq m - o(m)$ because $C \geq \sqrt[p]{z_p m}$ and $\mathrm{D}(z_p, p)$ is c_p-competitive. □

3.2 Lower bound

THEOREM 3.2. *Let A be a deterministic online algorithm. If A is c-competitive, then $c \geq c_p$.*

Proof. We construct a family of input sequences σ_l, for any integer $l \geq 1$. For a fixed l, let $i_0 = \lfloor \sqrt[2p+1]{l} \rfloor - 1$ and $l' = \lfloor \frac{l}{(i_0+1)^p} \rfloor$. Note that $l' = \Theta(l^{1-\frac{p}{2p+1}}) = o(l)$. We number the packets in σ_l starting with l'. Packet l' is sent at time 0. Packet k, for $l' < k \leq l$, is sent $\sqrt[p]{z_p k}$ time units after packet $k-1$. For $k > l$, packets k and $k-1$ are separated by exactly $\sqrt[p]{z_p l}$ time units. The adversary stops sending packets when the online algorithm decides to acknowledge two packets with the same acknowledgement. Let m be the number of the last packet sent.

If $m \leq l$, the cost incurred by the online algorithm A is at least $m - l' + (z_p m)^{\frac{p}{p}} = c_p m - l'$. The adversary can acknowledge each packet immediately, incurring no delays so that its cost is at most $m - l' + 1$. The ratio of the cost incurred by A to the cost incurred by the adversary is at least

$$\frac{c_p m - l'}{m - l' + 1} = c_p + \frac{z_p l' - c_p}{m - l' + 1}$$

and this expression is at least c_p if $l \geq 2^{2p+1}$. In this case $l' \geq 4$ and $z_p l' - c_p \geq 0$ because $z_p > 1/2$ and $c_p < 2$.

We concentrate on the case $m > l$. The adversary chooses an acknowledgement interval of $\sqrt[p]{z_p m}$ time units. Using the familiar charging scheme we can show that, in order to upper bound the number of acknowledgements incurred by the adversary, the total cost charged to packets k with $\lfloor \frac{l}{i^p} \rfloor < k \leq \lfloor \frac{l}{(i+1)^p} \rfloor$ and $1 \leq i \leq i_0$ is at most $(\lfloor \frac{l}{i^p} \rfloor - \lfloor \frac{l}{(i+1)^p} \rfloor)\frac{1}{i+1} + \frac{1}{i}$. The total cost charged to packets k with $k > l$ is at most $(m-l)\frac{1}{2} + \frac{1}{2}$. Hence the total number of acknowledgements sent by the adversary is at most

$$\begin{aligned}
&(m-l)\tfrac{1}{2} + \tfrac{1}{2} + \sum_{i=1}^{i_0} \left(\left(\left\lfloor \frac{l}{i^p} \right\rfloor - \left\lfloor \frac{l}{(i+1)^p} \right\rfloor\right) \frac{1}{i+1} + \frac{1}{i}\right) \\
&\quad + \frac{1}{i_0+1} \\
&\leq (m-l)\tfrac{1}{2} + \tfrac{l}{2} - \sum_{i=2}^{\infty} \frac{l}{i^{p+1}(i+1)} + 2H_{i_0+1} \\
&= (m-l)\tfrac{1}{2} + l - z_p l + O(\log l).
\end{aligned}$$

The total cost paid by the adversary is at most $(m-l)\frac{1}{2} + l + O(\log l)$ and the ratio of the cost incurred by A to the cost incurred by the adversary is at least

$$\frac{c_p l + m - l - l'}{l + \frac{1}{2}(m-l) + O(\log l)}.$$

This ratio approaches a value of at least c_p as $l \to \infty$ because $l' = o(l)$. □

4 Randomization

THEOREM 4.1. *For the dynamic TCP acknowledgement problem with objective function f, no randomized online algorithm can achieve a competitive ratio smaller than $c \geq 3/(3 - \frac{2}{e})$ against any oblivious adversary.*

Proof. We apply Yao's principle [1, 9] and construct a probability distribution on input sequences σ_l, for any integer $l \geq 1$, such that for any deterministic online algorithm D,

$$\lim_{l \to \infty} \frac{E[C_D(\sigma_l)]}{E[C_{ADV}(\sigma_l)]} \geq \frac{3}{3 - 2/e}$$

and

$$\lim_{l \to \infty} E[C_{ADV}(\sigma_l)] = \infty.$$

Here $E[C_{ADV}(\sigma_l)]$ and $E[C_D(\sigma_l)]$ denote the costs incurred by the adversary and the deterministic online algorithm, respectively. An input σ_l consists of *triples*. A triple is a set of three data packets that are separated by l time units each. More precisely, the second packet is sent exactly l time units after this first packet of the triple; the third packet is sent l time units after the second packet. Thus a triple has a total length of $2l$. The adversary sends triples, where the distance between triples is chosen so large that it does not make sense to acknowledge packets in two different triples with one acknowledgement. With probability $p_i = q(1-q)^{i-1}$, where $q = 1/l$, the adversary sends exactly i triples, for any $i \geq 1$. Note that $\sum_{i=1}^{\infty} q(1-q)^{i-1} = 1$. Triple i and $i+1$ are separated by $3l/p_{i+1}$ time units.

If a deterministic online algorithm on this input acknowledges packets of different triples together and if this happens for the first time for packets from triples i and $i+1$, then the expected cost of the algorithm is at least $p_{i+1}(3l/p_{i+1}) = 3l$. In the following we concentrate on the case that a deterministic online algorithm on this input never acknowledges packets from different triples together. We characterize an algorithm by two non-negative integers l_1, l_2, with $l_1 < l_2$ such that $l_1 + 1$ is the first triple where the algorithm acknowledges at least two packets together and $l_2 + 1$ is the index of the first triple where all the three packets are acknowledged together. We refer to this strategy as $D(l_1, l_2)$, $l_1 \leq l_2$. Algorithm $D(l_1, \infty)$, $l_1 \geq 0$, never acknowledges all the packets of one triple together and $D(\infty, \infty)$ never acknowledges any packets together. To analyze the expected cost, we need the following lemma.

LEMMA 4.1. *a) If $l_1 < l_2$, then*
$E[C_{D(l_1,l_2)}(\sigma_l)] = E[C_{D(l_1+1,l_2)}(\sigma_l)]$.

b) If $l_1 \leq l_2$, then
$E[C_{D(l_1,l_2)}(\sigma_l)] = E[C_{D(l_1,l_2+1)}(\sigma_l)]$.

c) For any $l_1 \geq 0$,
$E[C_{D(l_1,\infty)}(\sigma_l)] = E[C_{D(l_1+1,\infty)}(\sigma_l)]$.

Proof. We prove part a). The other parts can be proved similarly. We have

$$\begin{aligned}
&E[C_{D(l_1,l_2)}(\sigma_l)] \\
&= \sum_{i=1}^{l_1} 3ip_i + \sum_{i=l_1+1}^{l_2} (l + l_1 + 2i)p_i \\
&\quad + \sum_{i=l_2+1}^{\infty} (2l + l_1 + l_2 + i)p_i \\
&= \sum_{i=1}^{l_1+1} 3ip_i - 3(l_1+1)p_{l_1+1} \\
&\quad + \sum_{l_1+2}^{l_2} (l + l_1 + 1 + 2i)p_i \\
&\quad + (l + l_1 + 2(l_1+1))p_{l_1+1} - \sum_{i=l_1+2}^{l_2} p_i \\
&\quad + \sum_{i=l_2+1}^{\infty} (2l + l_1 + 1 + l_2 + i)p_i - \sum_{i=l_2+1}^{\infty} p_i \\
&= E[C_{D(l_1+1,l_2)}(\sigma_l)] + (l-1)p_{l_1+1} - \sum_{i=l_1+2}^{\infty} p_i \\
&= E[C_{D(l_1+1,l_2)}(\sigma_l)]. \square
\end{aligned}$$

Parts a) and b) of the above lemma imply that $E[C_{D(0,0)}(\sigma_l)] = E[C_{D(l_1,l_2)}(\sigma_l)]$ for any $0 \leq l_1 \leq l_2$. Hence it suffices to compute $E[C_{D(0,0)}(\sigma_l)] = \sum_{i=1}^{\infty}(2l + i)p_i = 2l + 1/q = 3l$. Part c) of the Lemma 4.1 implies $E[C_{D(0,\infty)}(\sigma_l)] = E[C_{D(l_1,\infty)}(\sigma_l)]$, for any $l_1 \geq 0$. We have $E[C_{D(0,\infty)}(\sigma_l)] = \sum_{i=1}^{\infty}(l + 2i)p_i = 3l$. Finally $E[C_{D(\infty,\infty)}(\sigma_l)] = \sum_{i=1}^{\infty} 3ip_i = 3l$. Thus, in any case the expected online cost is at least $3l$. It remains to analyze the expected cost incurred by the adversary. If the input consists of at most l triples, the adversary acknowledges the packets individually; otherwise it incurs a delay of $2l$ and acknowledges the packets of each triple together. Thus

$$\begin{aligned}
&E[C_{ADV}(\sigma_l)] \\
&= \sum_{i=1}^{l} 3ip_i + \sum_{i=l+1}^{\infty} p_i(i + 2l) \\
&= l + 2\sum_{i=1}^{l} ip_i + 2\sum_{i=l+1}^{\infty} p_i l \\
&= l + 2q\tfrac{1-(l+1)(1-q)^l + l(1-q)^{l+1}}{q^2} + 2l(1-q)^l \\
&= 3l - 4l(1-1/l)^l + 2l(1-1/l)^l
\end{aligned}$$

Thus $\lim_{l\to\infty} E[C_{ADV}(\sigma_l)]/l = 3 - 2/e$ and the theorem follows. □

THEOREM 4.2. *For the dynamic TCP acknowledgement problem with objective function f_p, no randomized online algorithm can achieve a competitive ratio smaller than $c \geq 2/(2 - \frac{1}{e})$ against any oblivious adversary.*

Proof. An input σ_l, for any integer $l \geq 1$, consists of *pairs*. A pair are two packets that are $\sqrt[p]{l}$ time units apart. With probability $p_i = q(1-q)^{i-1}$, $q = 1/l$, the input consists of i pairs, for any $i \geq 1$. Pairs i and $i+1$ are separated by $\sqrt[p]{2l/p_{i+1}}$ time units. If a deterministic online algorithm acknowledges packets of different intervals together and if this happens for the first time for packets from pairs i and $i+1$, then the expected cost is at least $p_{i+1}(\sqrt[p]{2l/p_{i+1}})^p = 2l$. In the following we consider algorithms that never acknowledge packets from different pairs together and denote by $D(l')$, $l' \geq 1$, the algorithm that acknowledges packets in the first l' pairs separately and the packets in the $(l'+1)$-st pair together. $D(\infty)$ is the algorithm that never acknowledges packets together. We have $E[C_{D(\infty)}(\sigma_l)] = \sum_{i=1}^{\infty} 2ip_i = 2q/q^2 = 2l$. For any $l \geq 0$,

$$\begin{aligned}
E[C_{D(l')}(\sigma_l)] &= \sum_{i=1}^{l'} 2ip_i + \sum_{i=l'+1}^{\infty} (l' + i + l)p_i \\
&= E[C_{D(l'+1)}(\sigma_l)] - 2(l'+1)p_{l'+1} \\
&\quad + (2l' + 1 + l)p_{l'+1} - \sum_{i=l'+2}^{\infty} p_i
\end{aligned}$$

$$= \quad E[C_{D(l'+1)}(\sigma_l)]$$

and hence $E[C_{D(0)}(\sigma_l)] = [C_{D(l')}(\sigma_l)]$, for any $l' \geq 0$. We have $E[C_{D(0)}(\sigma_l)] = \sum_{i=1}^{\infty}(l+i)p_i = 2l$ and conclude that the expected online cost is at least $2l$.

The adversary acknowledges the packets of pairs separately if at most l intervals are sent; otherwise it always acknowledges the packets of pairs together. Hence

$$\begin{aligned} E[C_{ADV}(\sigma_l)] &= \sum_{i=1}^{l} 2ip_i + \sum_{i=l+1}^{\infty} p_i(i+l) \\ &= l + \sum_{i=1}^{l} ip_i + \sum_{i=l+1}^{\infty} p_i l \\ &= l + q\frac{1-(l+1)(1-q)^l + l(1-q)^{l+1}}{q^2} + lq^l \\ &= 2l - 2l(1-1/l)^l + l(1-1/l)^l \end{aligned}$$

adn $\lim_{l\to\infty} E[C_{ADV}(\sigma_l)]/l = 2 - 1/e$. The theorem follows. □

References

[1] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[2] D.R. Dooly, S.A. Goldman, and S.D. Scott. TCP dynamic acknowledgement delay: Theory and practice. *Proc. 30th Annual ACM Symposium on Theory of Computing*, 389–398, 1998.

[3] D.R. Dooly, S.A. Goldman, and S.D. Scott. On-line analyis of the TCP acknowledgment delay problem. *Journal of the ACM*, 48:243–273, 2001.

[4] A.R. Karlin, C. Kenyon and D. Randall. Dynamic TCP acknowledgement and other stories about $e/(e-1)$. *Proc. 31st ACM Symposium on Theory of Computing*, 502–509, 2001.

[5] J. Noga. Private communication.

[6] J. Noga, S.S. Seiden, G.J. Woeginger. A faster off-line algorithm for the TCP acknowledgement problem. *Information Processing Letters*, 81:71-73, 2002.

[7] S.S. Seiden. A guessing game and randomized online algorithms. *Proc. 32nd ACM Symposium on Theory of Computing*, 592–601, 2000.

[8] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

[9] A.C.-C. Yao. Probabilistic computations: Towards a unified measure of complexity. *Proc. 18th Annual Symposium on Foundations of Computer Science*, 222–227, 1977.