Alcom-FT Technical Report Series
ALCOMFT-TR-02-60

Packet Bundling*

Jens S. Frederiksen Kim S. Larsen

University of Southern Denmark, Odense

{svalle,kslarsen}@imada.sdu.dk

Abstract

When messages, which are to be sent point-to-point in a network, become available at irregular intervals, a decision must be made each time a new message becomes available as to whether it should be sent immediately or if it is better to wait for more messages and send them all together. Because of physical properties of the networks, a certain minimum amount of time must elapse in between the transmission of two packets. Thus, whereas waiting delays the transmission of the current data, sending immediately may delay the transmission of the next data to become available even more.

We consider deterministic and randomized algorithms for this on-line problem. It turns out that the flow-time cost function, which is the standard measure for problems of this type, cannot distinguish between the different algorithms. This leads us to develop an alternative quality measure under which we analyze our algorithms and determine which are best by obtaining tight results.

It is interesting to note that our results are very different from earlier works on the problem where the physical properties of the networks were emphasized less.

*Supported in part by the Danish Natural Science Research Council (SNF) and in part by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

1 Introduction

We consider point-to-point transmission of data in a network. Transmission of data is in the form of packets, which contain some header information, such as the identification of the receiver and sender, followed by the actual data. For obvious reasons, data is also referred to as messages.

When data to be sent becomes available a little at a time at irregular intervals, the question arises on the sending side whether to send a given piece of data immediately or whether to wait for the next data to become available, such that they can be sent together. Sending the data together is referred to as *Packet Bundling*.

The reason why this is at all an issue is because of physical properties of the networks which imply that after one packet has been sent, a certain minimum amount of time must elapse before the next packet may be sent. Thus, whereas waiting for more data will certainly delay the transmission of the current data, sending immediately may delay the transmission of the next data to become available even more. In addition to reducing the overall transmission delay when bundling messages, we also reduce the bandwidth requirement of the sender, since overhead due to packet headers and network gap is reduced. The problem of making these decisions is referred to as the *Packet Bundling Problem*.

A very similar problem, the Dynamic TCP Acknowledgment Problem, was introduced in [5]. Since it usually does not make sense to delay the transmission of large amounts of data, the focus for packet bundling is small messages, and acknowledgments to the receipt of packets are examples of such. The problem was studied as an on-line problem [3] with the cost function being the number of packets sent plus the sum of the latencies for each message. The latency of a message is the time from when the data is available until it is sent. This is known as the flow-time cost measure. An exact characterization of the optimal algorithms for the deterministic and randomized case can be found in [5] and [7], respectively.

The problem we consider is different from the Dynamic TCP Acknowledgment Problem in two ways: we build on top of an accepted model for distributed computing and, related to this decision, we also choose a different cost function.

To our knowledge, the currently most widely accepted model of computation for distributed computing is now the so-called LogP model [4], which is tractable from a theoretical point of view, but also realistic enough that good theoretical algorithms are likely to be good in practice as well; on many different platforms. The ingredients in the model are the latency L, the processor overhead in sending messages o, the gap imposed by the network between messages g, and the number of processors P. We refer to the original paper for a complete treatment. The LogP model has been extended to improve the treatment of large messages [1] and the model is supported by experimental work, supplying methods for determining the concrete values of the parameters on a given system [9].

Comparing the theoretical work of [5, 7] with the model assumptions of [4], the most noticeable difference is the lack of the gap or overhead parameter in [5, 7]. In their work, they allow packets to be sent arbitrarily small distances apart. This significantly influences the results. With decreasing intervals between messages, the latency of each packet contributes to the cost function. Thus, a good algorithm would send frequently; very likely faster than possible in a real world scenario.

We base our theoretical work on the LogP model, which means we respect the physical gap and overhead. At times when messages become available separated by very small time intervals, the cost function of [5, 7] would impose an unreasonable large penalty, if the decision is to delay transmission, which leads us to consider another cost function: the sum of time intervals when at least one unsent message is available. In our opinion, this measure is just as natural and it has the significant advantage that good and bad algorithms can be distinguished.

Packet bundling has been considered experimentally in [10] for a concrete specialized application, namely real-time simulations. However, the conditions are quite different in that messages have different priorities. The goal is also different since there are some absolute tolerances for the delay of messages of different types.

We analyze natural families of deterministic and randomized algorithms for the problem and find the best. Additionally, we show that these results could not have been obtained using the standard flow-time cost function.

2 Packet Bundling

Referring to the description of the LogP model from the introduction, since we are considering the situation from the perspective of a single processor, there is no reason to distinguish between gap and overhead, since the crucial value is the maximum of the two. In the remainder of the paper, we normalize with respect to this maximum, and assume that it is one (the important fact is that it is different from zero).

With this in mind, we now state the problem in a form which is formal, but also convenient to work with.

Consider of the problem of a person A wishing to send small messages to another person B. All messages have to be sent using the same, single messenger, who uses one time unit for each delivery, i.e., when the messenger has left with one or more messages (a packet), no messages can be sent for the next one time unit.

When A decides to send a message, she can either send it immediately (assuming that the messenger is in), or wait some time (probably less than one) to see whether other messages have to be sent, so that these messages can be sent together.

In a people scenario, as opposed to a machine scenario, one can think of a hotel/airport shuttle.

The formal definition of the problem is as follows:

Definition 1 In the *Packet Bundling Problem* one is given a sequence $\sigma = a_1, \ldots, a_n$ of message arrival times and is asked to give a sequence of packet times $p_1, \ldots, p_m$ at which packets of messages are sent. All messages are considered to be small, so that an unlimited number of messages can fit in a packet. The set of messages sent in packet p_i is denoted $\tilde{p}_i$.

The packets should respect the following restrictions:

- All packet times should be at least one unit apart, i.e.,

$$\forall i < m\colon\ p_{i+1} - p_i \geq 1.$$

- All messages should be sent no earlier than their arrival time, i.e.,

$$\forall i \leq n\ \exists j \leq m\colon\ a_i \in \tilde{p}_j \wedge a_i \leq p_j.$$

If a packet is sent at time p_i, then the set of messages contained in the packet are considered delivered at time $p_i + 1$.

The cost function measures the total time elapsed while there are messages which have arrived, but have not been delivered:

$$\sum_{i=1}^{m} \left((p_i + 1) - \max\{(p_{i-1} + 1), \min_{a_j \in \tilde{p}_i} a_j\} \right)$$

where we define $p_0 = -\infty$. For convenience, we identify a message with its arrival time, justifying the notation $a_j \in \tilde{p}_i$, assuming that these are distinct; if not, $\tilde{p}_i$ can be thought of as a multiset. □

We consider on-line algorithms [3]. Thus, the algorithm is given the arrival times, a_i, one at a time, and has to decide the packet time at which the message is sent before the arrival time of the next message (if any) is revealed. If the next message arrives before the previously chosen packet time, the algorithm is allowed to reconsider its choice. For any algorithm, ALG, and input sequence, σ, we let $ALG(\sigma)$ denote the value of the cost function when ALG is run on σ.

The performance of deterministic algorithms is measured in comparison with the optimal off-line algorithm, OPT, using the standard competitive ratio [6, 11, 8, 3]. OPT knows the entire input sequence, when it decides when to send each packet, and can hence achieve a lower cost.

An algorithm ALG is (strictly) c-competitive for a constant c, if for all input sequences, σ, the following holds: $ALG(\sigma) \leq c \cdot OPT(\sigma)$. The infimum of all such values c is called the competitive ratio of ALG.

The performance of randomized algorithms is measured likewise, though using the expected competitive ratio, $E[ALG(\sigma)]$, instead.

3 Deterministic Algorithms

In this section, we consider deterministic on-line algorithms for the problem. Specifically, we consider the following family of algorithms:

A_k: When a message arrives, it is sent together with all messages (if any) arriving after this one at the earliest possible time after k time units.

Without loss of generality, we assume that if A_k decides to let the messenger leave at a certain point in time, and one or more messages arrive exactly when he is about to leave, then he leaves without these new messages. If this is a problem in a proof, then all messages arriving when the messenger leaves can be considered to arrive $\epsilon \in o(1)$ time units later. Because of the infimum which is taken in the definition of the competitive ratio, this will generally not alter the result.

The following theorem states the competitive ratio of this family of algorithms:

Theorem 1 The competitive ratio of A_k is:

$$\mathcal{R}(A_k) = \begin{cases} 1 + \frac{1}{1+k} & , \text{ if } 0 \leq k < \hat{\varphi} \\ 1 + k & , \text{ if } \hat{\varphi} \leq k \end{cases}$$

where $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ and $\hat{\varphi} = \varphi - 1$. The best ratio φ is achieved by $A_{\hat{\varphi}}$.

For a fixed algorithm, A_k, any input sequence for our problem can be divided into phases as follows: Each phase starts with the arrival of the first message after the previous phase has ended, and ends at the earliest possible time when there are no messages to deliver and the messenger is in. In the special case when the messenger returns at the exact same time a new message arrives (and no other messages are due for delivery), the phase ends, and the new message starts the next phase.

In the proof, we need the following two lemmas:

Lemma 1 For a worst-case sequence for A_k, we can assume that the messenger carries only one message at a time.

Proof Consider a worst-case sequence, and assume that A_k has a $\tilde{p}_i$ of size larger than one. We can construct another sequence where $|\tilde{p}_i| = 1$ and the competitive ratio is no better.

Adjusting the sequence by removing messages which do not give rise to the minimum value will not change A_k's behavior and will leave the competitive ratio unchanged, since OPT can send fewer messages without increasing its cost. $\square$

Lemma 2 There exists a worst-case sequence for A_k where, if any messages arrive when the messenger is out, they arrive exactly at the point in time where the messenger leaves or returns.

Proof We consider a worst-case sequence where a message arrives in the time interval, $(t, t+1)$, when A_k's messenger is out, and neither is leaving nor returning, and we transform this sequence into an equally bad sequence, where one message fewer arrives in this interval.

The message which is being delivered in the time interval $[t, t+1]$ must have arrived no later than time $t-k$ in order to be sent at time t. Now assume that a message arrives at time τ, where $t < \tau < t+1$. By Lemma 1, we may assume that this is the only message arriving in this time interval, since otherwise A_k would carry two messages next time.

If $\tau \in (t, t+1-k)$ (which may be an empty set, if $k > 1$), then A_k's messenger has to leave with the new message immediately after returning at time $t+1$, and A_k's cost is not influenced by exactly when in the interval $[t, t+1-k]$ the message arrives. If OPT sends the message together with previous messages, then OPT's cost will be minimized for $\tau = t$. If OPT sends the new message on its own, τ can be assumed to be $t+1-k$ as OPT's cost cannot increase if a message arrives later. The time $t+1-k$ is still while A_k's messenger is out, but that will be dealt with in the next case.

If $\tau \in [t+1-k, t+1)$, this message will be sent by A_k at time $\tau + k$. As the previous message arrived at some time before $\tau - 1$, OPT can be assumed not to send the new message together with the previous. Thus, if the new message is shifted together with all later messages by $t+1-\tau$ time units, then OPT's cost will be the same, whereas A_k's cost does not decrease. Consequently, we may assume that $\tau = t+1$. $\square$

Proof [Proof of Theorem 1] Let us first consider the case when $k \leq 1$:

By Lemmas 1 and 2, a worst-case sequence can be assumed to consist of phases of the following form:

$$\sigma_1 = 0 \qquad \text{or} \qquad \sigma_n = 0, k, k+1, k+2, \ldots, k+(n-2)$$

where n is the number of messages. We separate each phase from the next by more than two time units. By definition of A_k, this means that messages from different phases cannot interfere, so relative costs can be calculated separately for each phase.

A_k's cost is $A_k(\sigma_n) = k + n$, whereas OPT's cost is

$$OPT(\sigma_n) = \begin{cases} k+n-1 & , \quad \text{if } n > 1 \\ 1 & , \quad \text{if } n = 1 \end{cases}$$

For $n = 1$, this gives a competitive ratio of $\frac{A_k(\sigma_1)}{OPT(\sigma_1)} = k+1$, and for $n > 1$, it gives a competitive ratio of $\frac{A_k(\sigma_n)}{OPT(\sigma_n)} = \frac{k+n}{k+n-1}$, which is maximized for $n = 2$, where $\frac{A_k(\sigma_2)}{OPT(\sigma_2)} = \frac{k+2}{k+1} = 1 + \frac{1}{1+k}$.

Comparing the two cases, we find that σ_2 is the worst possible for $k \leq \hat{\varphi}$, whereas σ_1 is worst for $\hat{\varphi} \leq k \leq 1$.

Let us then consider the case when $1 < k$. Again by Lemmas 1 and 2, a worst-case sequence can be assumed to consist of phases of the following form:

$$\sigma_n = 0, k, 2k, \ldots, k(n-1)$$

where n is the number of messages. Each phase is separated from the next by more than $1 + k$ time units, so relative costs can be calculated separately for each phase.

A_k's cost is $A_k(\sigma_n) = 1 + kn$, whereas OPT's cost is $OPT(\sigma_n) = n$. This gives a competitive ratio of $\frac{A_k(\sigma_n)}{OPT(\sigma_n)} = \frac{1+kn}{n} = \frac{1}{n} + k$, which is maximized at $n = 1$, where $\frac{A_k(\sigma_1)}{OPT(\sigma_1)} = 1 + k$. □

Finally, we show that no deterministic algorithm has a competitive ratio better than $A_{\hat{\varphi}}$.

Theorem 2 Let ALG be any deterministic algorithm for the *Packet Bundling Problem.* Then $\mathcal{R}(ALG) \geq \mathcal{R}(A_{\hat{\varphi}}) = \varphi$.

Proof We show how to construct an input sequence for ALG, where it has a competitive ratio larger than or equal to φ.

The input will be given in a number of phases, each consisting of either one or two messages. Between each phase there is a time interval large enough so that neither ALG nor $A_{\hat{\varphi}}$ at the end of the interval has any messages to deliver, nor are they at the moment delivering any messages.

Let us first consider phase σ_i, and let the first message in this phase arrive at time a_{i_1}. Let k_i be the length of the time interval ALG waits before it sends the message. Referring to Theorem 1, we know for A_{k_i} whether a worst-case phase for A_{k_i} has one or two messages. If it has two, another message is set to arrive at time $a_{i_2} = a_{i_1} + k_i$, if not, the phase ends.

Referring again to Theorem 1, the following holds for phase σ_i:

$$ALG\left(\sigma_i\right) \geq A_{k_i}\left(\sigma_i\right) \geq A_{\hat{\varphi}}\left(\sigma_i\right) \geq \varphi\, OPT\left(\sigma_i\right)$$

For $k_i \leq \hat{\varphi}$, a phase consists of two messages. Whereas A_{k_i} sends the second message immediately after having sent the first, we cannot be sure that ALG does too. Thus, the costs of A_{k_i} and ALG are not necessarily the same (giving inequality instead of equality between the first two terms).

Thus, for the entire input sequence, we have $ALG\left(\sigma\right) \geq \varphi\, OPT\left(\sigma\right)$. □

4 Randomized Algorithms

We now consider a family of randomized algorithms solving the same problem. Our deterministic algorithm family A_k chose a specific k and sent a message at the earliest possible time after k time units. $RAND_\Delta$ chooses the interval it waits at random.

$RAND_\Delta$: When a new message arrives and no other messages are waiting, this message and later messages (if any) are sent after a period of time chosen uniformly between 0 and Δ, or at the first possible time hereafter.

We only consider algorithms with $\Delta \leq 1$, since any algorithm, $RAND_\Delta$, with $\Delta > 1$ easily can be seen to have a competitive ratio larger than $RAND_1$.

One could also consider other families of randomized algorithms. Instead of using a uniform distribution, we could have used an exponential distribution with parameter Δ varying from zero to infinity (as in [7]), or a cut-off exponential distribution described by the density function: $f(\delta) =$

$\frac{1}{\Delta}e^{-\frac{\delta}{\Delta}}/\left(1-e^{-1}\right)$ for $\delta \in [0,\Delta]$, and zero otherwise. By careful examination both of these are for any Δ easily shown to have a worse competitive ratio than the best member of the $RAND_\Delta$-family.

Without loss of generality, we will as with A_k assume that messages arriving exactly at the randomly chosen time δ when the messenger leaves will not be delivered immediately. For $\Delta > 0$, this does not make any difference to the expected competitive ratio as δ is chosen uniformly at random in the range $[0,\Delta]$. For $\Delta = 0$, $RAND_0$ and A_0 are identical, and we can as for A_0 consider all messages arriving when the messenger leaves, as arriving $o(1)$ time later without any difference.

The competitive ratio for $RAND_\Delta$ is given by the following theorem.

Theorem 3 The expected competitive ratio of $RAND_\Delta$ is

$$\overline{\mathcal{R}}\left(RAND_\Delta\right) = \begin{cases} \frac{1}{2} + \frac{3}{2(\Delta+1)} & , \text{ if } 0 \le \Delta \le \frac{1}{2} \\ \frac{6\Delta^2+4\Delta+1}{4\Delta(1+\Delta)} & , \text{ if } \frac{1}{2} \le \Delta \le \sqrt[3]{\frac{1}{2}} \\ \frac{\Delta}{2}+1 & , \text{ if } \sqrt[3]{\frac{1}{2}} \le \Delta \le 1 \end{cases}$$

The best ratio $\frac{\sqrt[3]{\frac{1}{2}}}{2} + 1 \approx 1.397$ is achieved by $RAND_{\sqrt[3]{\frac{1}{2}}}$.

As for A_k, the theorem is shown by constructing a worst-case input sequence. For a fixed $RAND_\Delta$, any input sequence is divided into phases almost as previously: Each phase starts with the arrival of the first message after the previous phase has ended, and ends exactly when, regardless of the random choices made, there are definitely neither any messages waiting to be sent nor is the messenger out. In the event that a new message arrives at the exact same time as the messenger returns, and where no random choices would have made the messenger arrive later, the phase ends, and the new message starts the next phase. Due to linearity of expectation, it is enough to consider a worst case phase.

Before showing the main theorem, we need the following lemmas:

Lemma 3 Messages in a worst-case phase for $RAND_\Delta$ are not further than one apart, i.e., $\forall i : a_{i+1} - a_i \le 1$.

Proof Let a_i be the first message such that $a_{i+1} > a_i + 1$. As all messages before a_i are at most one apart, OPT can send the messages $a_1, \ldots, a_i$ at time a_i, so that it does not incur any cost between time $a_i + 1$ and a_{i+1}. This means that the arrival of message a_{i+1} (together with all other messages after message a_{i+1}) can be shifted to any point in time further ahead without increasing the cost of OPT.

Since message a_i leaves with the messenger at time $a_i + 1$ at the latest, message a_{i+1} arrives either when the messenger is out or when the messenger has returned (and then no other messages will be waiting at that time). The cost of $RAND_\Delta$ is maximal if the randomly chosen waiting time after a_{i+1} is not shared by time where the messenger is out, i.e., the cost is maximal if message a_{i+1} arrives after the messenger returns, and thereby if the message is not in the same phase, but in the next. $\square$

Lemma 4 For a worst case phase with messages $\sigma = (a_1 = 0), \ldots, a_m$, the expected competitive ratio of $RAND_\Delta$ is at most:

$$\overline{\mathcal{R}}(RAND_\Delta) = \frac{E[RAND_\Delta(\sigma)]}{OPT(\sigma)} \leq \frac{a_m + 2}{a_m + 1} = 1 + \frac{1}{a_m + 1}$$

Proof Follows directly from Lemma 3. $\square$

The following lemma shows that a worst case phase with $\sigma = (a_1 = 0), \ldots, a_m$ and $a_m \leq 1$ can be assumed to contain at most two messages:

Lemma 5 For any phase with messages $\sigma = (a_1 = 0), \ldots, a_m$ and $a_m \leq 1$, the phase obtained by looking only at the first and the last message of σ, $\sigma' = a_1, a_m$, has an expected competitive ratio which is no better, i.e.,

$$\frac{E[RAND_\Delta(\sigma)]}{OPT(\sigma)} \leq \frac{E[RAND_\Delta(\sigma')]}{OPT(\sigma')}$$

Proof As $a_m \leq 1$, we have $OPT(\sigma) = OPT(\sigma') = a_m + 1$. For $RAND_\Delta$, we consider two cases. Assume that the messenger leaves with the first message (and other messages if any) at time δ. If $a_m < \delta$, then the cost of σ is the same as for σ'. If $\delta \leq a_m$, then message a_m is not sent until the messenger leaves the next time. This point of time is determined by δ, Δ, and the first message, a_i, with $\delta \leq a_i$. If we leave out the messages before message a_m (but after message a_1), then on average the messenger does not leave earlier the second time (but maybe later). $\square$

Furthermore, as the next lemma shows, if $a_m \leq 1$, then in addition to assuming that $m \leq 2$, we can assume that $a_m \in \{0, \Delta\}$. Note that for $\Delta > 0$, due to the definition of our cost function, $a_2 = 0$, i.e., $\sigma = 0, 0$ has the same expected competitive ratio as $\sigma = 0$. For $\Delta = 0$, the sequence $\sigma = 0, 0$ is the same as $\sigma = 0, \Delta$.

Lemma 6 A worst case input sequence for $RAND_\Delta$ with two messages, $\sigma = (a_1 = 0), a_2$, where $a_2 \leq 1$, must have $a_2 \in \{0, \Delta\}$. This gives the following lower bounds, of which at least one is an upper bound for all input sequences $\sigma = a_1, \ldots, a_m$, where $a_m - a_1 \leq 1$:

$\sigma = 0$ has an expected competitive ratio of

$$1 + \frac{\Delta}{2}.$$

$\sigma = 0, \Delta$ has an expected competitive ratio of c, where

$$c = \begin{cases} \frac{1}{2} + \frac{3}{2(\Delta+1)} & , \text{ if } \Delta \leq \frac{1}{2} \\ \frac{6\Delta^2+4\Delta+1}{4\Delta(\Delta+1)} & , \text{ if } \Delta > \frac{1}{2} \end{cases}$$

Proof For $\sigma = 0$, the expected competitive ratio of $RAND_\Delta$ is $\frac{E[RAND_\Delta(\sigma)]}{OPT(\sigma)}$ $= 1 + \frac{\Delta}{2}$.

For $\sigma = 0, a_2$, we prove the result by case analysis.

Let δ be the (now fixed) randomly chosen time at which $RAND_\Delta$ decides to send the first message (and the next, if $a_2 < \delta$). The expected cost of $RAND_\Delta$ can be described as follows:

$$k(a_2, \delta) = \delta + 1 + \begin{cases} 0 & , \text{ if } a_2 \leq \delta \\ 1 & , \text{ if } \delta < a_2 \leq \delta + 1 - \Delta \\ 1 + \frac{(a_2+\Delta)-(\delta+1)}{2} & , \text{ if } \delta + 1 - \Delta < a_2 \leq 1 \end{cases}$$

This can be rephrased to a form which is more convenient in the proof:

$$k(a_2, \delta) = \delta + 1 + \begin{cases} 1 + \frac{(a_2+\Delta)-(\delta+1)}{2} & , \text{ if } 0 \leq \delta < a_2 + \Delta - 1 \\ 1 & , \text{ if } a_2 + \Delta - 1 \leq \delta < a_2 \\ 0 & , \text{ if } a_2 \leq \delta \leq \Delta \end{cases} \quad (1)$$

The expected cost of $RAND_\Delta$ is $E[RAND_\Delta(\sigma)] = \frac{1}{\Delta}\int_0^\Delta k(a_2,\delta)d\delta$, whereas $OPT(\sigma) = a_2 + 1$, giving us an expected competitive ratio of $c(a_2) = \frac{1}{\Delta(a_2+1)}\int_0^\Delta k(a_2,\delta)d\delta$.

Let us first consider the case when $\Delta \le \frac{1}{2}$. If $0 \le a_2 \le \Delta$, then the first case of $k(a_2,\delta)$ in equation (1) becomes empty, since $a_2 + \Delta - 1 \le 0$. Thus, the expected competitive ratio is

$$c(a_2) = \frac{\int_0^{a_2}(\delta+2)d\delta + \int_{a_2}^{\Delta}(\delta+1)d\delta}{\Delta(1+a_2)} = \frac{\frac{\Delta^2}{2}+\Delta+a_2}{\Delta(1+a_2)} = \frac{1}{\Delta} + \frac{\frac{\Delta}{2}+1-\frac{1}{\Delta}}{1+a_2}$$

Since $\Delta \le \frac{1}{2}$, this is easily seen to be maximal for a_2 as small as possible, i.e., $a_2 = 0$, and $c(0) = \frac{\Delta}{2}+1$. Let us then consider the case when $\Delta < a_2 \le 1-\Delta$:

$$c(a_2) = \frac{\int_0^{\Delta}(\delta+2)d\delta}{\Delta(1+a_2)} = \frac{\frac{\Delta}{2}+2}{1+a_2}$$

This is again maximal for a_2 as small as possible, i.e., it is at most $c(\Delta) = \frac{1}{2} + \frac{3}{2(1+\Delta)}$. The last case is when $1-\Delta \le a_2 \le 1$:

$$\begin{aligned} c(a_2) &= \frac{\int_0^{a_2+\Delta-1}(\delta+2+\frac{a_2+\Delta-1-\delta}{2})d\delta + \int_{a_2+\Delta-1}^{\Delta}(\delta+2)d\delta}{\Delta(1+a_2)} \\ &= \frac{\frac{\Delta}{2}+2+\frac{(a_2+\Delta-1)^2}{4\Delta}}{1+a_2} \end{aligned}$$

This is maximal for a_2 as large as possible. We have $c(1) = \frac{3}{8}\Delta + 1$, which is, however, smaller than the results found previously.

Let us then consider the case when $\Delta > \frac{1}{2}$. If $0 \le a_2 \le 1-\Delta$, we have:

$$c(a_2) = \frac{\int_0^{a_2}(\delta+2)d\delta + \int_{a_2}^{\Delta}(\delta+1)d\delta}{\Delta(1+a_2)} = \frac{\frac{\Delta^2}{2}+\Delta+a_2}{\Delta(1+a_2)} = \frac{1}{\Delta} + \frac{\frac{\Delta}{2}+1-\frac{1}{\Delta}}{1+a_2}$$

Since $\Delta > \frac{1}{2}$, depending on the enumerator (which changes sign at $\Delta = \sqrt{3}-1$), this is maximal in one of the endpoints, where $c(0) = \frac{\Delta}{2}+1$ and $c(1-\Delta) = \frac{\Delta^2+2}{2\Delta(2-\Delta)}$.

Let us then consider the case when $1-\Delta < a_2 \le \Delta$:

$$\begin{aligned} c(a_2) &= \frac{\int_0^{a_2+\Delta-1}(\delta+2+\frac{a_2+\Delta-1-\delta}{2})d\delta + \int_{a_2+\Delta-1}^{a_2}(\delta+2)d\delta + \int_{a_2}^{\Delta}(\delta+1)d\delta}{\Delta(1+a_2)} \\ &= \frac{3\Delta^2+2\Delta+{a_2}^2+2a_2+2\Delta a_2+1}{4\Delta(1+a_2)} \end{aligned}$$

By differentiation, we have $c'(a_2) = 0 \Leftrightarrow a_2 = -1 \pm \sqrt{3}\Delta$. Since $c''(a_2) > 0$ in this interval, c is maximal in one of the endpoints: $c(1-\Delta) = \frac{\Delta^2+2}{2\Delta(2-\Delta)}$ or $c(\Delta) = \frac{6\Delta^2+4\Delta+1}{4\Delta(1+\Delta)}$

The last case is when $\Delta < a_2 \leq 1$:

$$\begin{aligned} c\,(a_2) &= \frac{\int_0^{a_2+\Delta-1}(\delta + 2 + \frac{a_2+\Delta-1-\delta}{2})d\delta + \int_{a_2+\Delta-1}^{\Delta}(\delta+2)d\delta}{\Delta(1+a_2)} \\ &= \frac{\frac{\Delta}{2} + 2 + \frac{(a_2+\Delta-1)^2}{4\Delta}}{1+a_2} \end{aligned}$$

By differentiation, we have $c'(a_2) < 0$, i.e., this is maximal for a_2 as small as possible, with the same result as in the previous case.

By comparing the equations for $c(0)$, $c(1-\Delta)$, and $c(\Delta)$, we find that $c(1-\Delta) \leq \max\{c(0), c(\Delta)\}$, as desired, with equality for $\Delta \in \left\{\frac{1}{2}, 1\right\}$. □

Lemma 7 Let $\sigma = (a_1 = 0), \ldots, a_m$ be any worst case phase for $RAND_\Delta$ with $\Delta > \frac{1}{2}$ and $1 < a_m \leq 1 + \Delta$, then

$$\frac{E[RAND_\Delta(\sigma)]}{OPT(\sigma)} \leq \frac{\Delta^2 + 2\Delta + a_m - 1}{\Delta(a_m+1)}$$

Proof As before, let δ be the now fixed randomly chosen time at which $RAND_\Delta$ sends message a_1 (and all other messages arriving no later than time δ, if any). The worst-case cost of $RAND_\Delta$ can be described as follows:

$$w(\delta) \leq \begin{cases} a_m + 2 & , \text{ if } \delta + 1 < a_m \\ a_m + \Delta + 1 & , \text{ if } a_m < \delta + 1 \end{cases}$$

This gives an expected worst case cost for $RAND_\Delta$ of at most

$$E[RAND_\Delta(\sigma)] \leq \frac{1}{\Delta}\int_0^{\Delta} w(\delta)d\delta = \frac{\Delta^2 + 2\Delta + a_m - 1}{\Delta}$$

whereas $OPT(\sigma) = a_m + 1$, since σ is a worst-case phase. □

Proof [Proof of Theorem 3] Lemma 6 states some lower bounds for $RAND_\Delta$ which are the best possible for phases $\sigma = (a_1 = 0), \ldots, a_m$ with $a_m \leq 1$.

By Lemma 4, if $a_m > 1$, a worst case input sequence has a competitive ratio less than $\frac{3}{2}$.

For $\Delta \leq \frac{1}{2}$, Lemma 6 is enough, since the input sequence $0, \Delta$ has an expected competitive ratio of $\frac{1}{2} + \frac{3}{2(\Delta+1)} \geq \frac{3}{2}$.

For $\Delta > \frac{1}{2}$, by Lemma 4, any input sequence with $a_m > 1+\Delta$ has an expected competitive ratio of at most $\frac{(1+\Delta)+2}{(1+\Delta)+1}$. Lemma 7 gives a similar bound on the expected competitive ratio for $a_m \in (1, 1+\Delta]$. It can easily be shown that

$$\max\left\{\frac{\Delta+3}{\Delta+2}, \frac{\Delta^2+2\Delta+a_m-1}{\Delta(a_m+1)}\right\} \leq \max\left\{1+\frac{\Delta}{2}, \frac{6\Delta^2+4\Delta+1}{4\Delta(\Delta+1)}\right\}$$

Thus, either 0 or $0, \Delta$ is a worst case input sequence in this case.

Now, Lemma 6 gives the result. □

5 Flow-time cost

The most standard cost function for problems of this nature is flow-time cost, which is the sum of the flow-times for all the messages, where the flow-time of one message is the length of the time interval from when it arrives until it has been delivered. Using our cost function, we only distinguish between whether or not any messages are waiting. Using flow-time cost, the more messages that are waiting, the higher the cost, i.e., if two messages arrive at the same time, the cost of waiting doubles. Formally, the flow-time cost can be defined as follows:

$$\sum_{i=1}^{m} \sum_{a_j \in \tilde{p}_i} (p_i + 1 - a_j)$$

As we will see in the following lemmas, all deterministic on-line algorithms for the Packet Bundling Problem have a competitive ratio of at least two, when using flow-time cost. This competitive ratio is achieved for a family of algorithms we define as *reasonable* algorithms below. The family of randomized algorithms considered earlier, $RAND_\Delta$, also has a competitive ratio of two. This means that flow-time cost is not a good measure if we want to distinguish between different on-line algorithms.

Lemma 8 Let ALG be any deterministic algorithm for the Packet Bundling Problem. Using flow-time cost, the competitive ratio of ALG is at least two.

Proof We show how to construct an input sequence, σ, which has a competitive ratio arbitrarily close to two from below. Let the first message in σ arrive at time 0, and let k be the point of time at which ALG sends this message. Complete σ by using m messages all arriving at time k, i.e., exactly at the point of time when the messenger has left. The competitive ratio of ALG is at least:

$$\frac{k+1+2m}{k+1+m} = 2 - \frac{k+1}{k+1+m}$$

Consequently, for m chosen sufficiently large, the competitive ratio can come arbitrarily close to two from below. □

Definition 2 A reasonable on-line algorithm for the Packet Bundling Problem is a deterministic or randomized algorithm which never lets any message wait more than one time unit before it is sent. □

The two families of on-line algorithms, A_k and $RAND_\Delta$ for respectively $k \leq 1$ and $\Delta \leq 1$ both belong to this family.

The next lemma shows that any reasonable algorithm has a competitive ratio of at most two when using flow-time cost. Thus, any deterministic reasonable algorithm is optimal.

Lemma 9 Using flow-time cost any reasonable on-line (deterministic or randomized) algorithm for the Packet Bundling Problem has a competitive ratio of at most two.

Proof Consider any message a_i. Since the on-line algorithm is reasonable, this message waits at most one time unit before it is sent, which contributes at most two to the total cost. Since OPT has a cost of at least one for each message, the competitive ratio is at most two. □

Finally, every member of the algorithm family $RAND_\Delta$ also has a competitive ratio of two.

Lemma 10 Using flow-time cost, every member of the algorithm family $RAND_\Delta$ for the Packet Bundling Problem has a competitive ratio of two.

Proof Since any member of $RAND_\Delta$ is a reasonable algorithm as defined in Definition 2, an upper bound of two has already been shown in Lemma 9.

For the lower bound, we construct an input sequence, σ, which has a competitive ratio arbitrarily close to two. The first part of σ consists of m messages arriving with distance one. Without loss of generality, we assume that the first message arrives at time $-m+1$. Thus, the first part of σ is $-m+1, -m+2, \ldots, 0$. For $\delta \in [0, \Delta]$, the probability that message m leaves with the messenger at time δ or earlier is $F_m(\delta) = \left(\frac{\delta}{\Delta}\right)^m$, so the density function describing when the messenger leaves is $f_m(\delta) = \frac{d}{d\delta}F_m(\delta) = \frac{m}{\delta}\left(\frac{\delta}{\Delta}\right)^m$. The last part of σ consists of n messages all arriving at time Δ. The expected cost of $RAND_\Delta$ can then be bounded from below by the following:

$$\begin{aligned} E\left[RAND_\Delta(\sigma)\right] &\geq m + n\left(\int_0^\Delta f_m(\delta)(\delta + 1 - \Delta)d\delta + 1\right) \\ &= m + n\left(2 - \frac{\Delta}{m+1}\right) \end{aligned}$$

The cost for OPT is $OPT(\sigma) = m + \Delta + n$. Consequently, by choosing m large enough and n even larger, the expected competitive ratio of σ can come arbitrarily close to two. □

Lemma 10 can be extended to any randomized on-line algorithm behaving like the members of $RAND_\Delta$, except that it has another continuous distribution on its interval $[0, \Delta], \Delta \leq 1$.

6 Concluding Remarks

We have considered a new cost function instead of the cost function which is almost a standard in theoretical analysis of this type of problems, namely flow-time. With the new cost function, algorithms can be distinguished effectively, whereas using flow-time, this is not possible while respecting the LogP model assumptions. The behavior of the optimal off-line algorithm can be a little peculiar, however. If we consider sequences where n messages arrive less than one unit apart, nothing in our cost function encourages the optimal off-line algorithm to send any messages until the nth message has arrived.

While the behavior of an off-line optimal algorithm is secondary to the ability of the total set-up to distinguish between good and bad on-line algorithms, our results are robust enough that the behavior of OPT could be altered. Assume that we change the cost function such that when a message has been waiting for one time unit (or equivalently, has not been delivered two units after it became available), a strictly higher penalty is imposed. This will encourage a different behavior, where messages are sent earlier. However, OPT can still send all messages with the same cost. It will send at time t_n immediately after the nth message has arrived (as before), but it could also send at all times in the set $\{t_n - i | i \in \mathbb{N}, t_n - i \geq 0\}$.

The cost of all reasonable on-line algorithms as defined in the previous section, including the cost of A_k for $k \leq 1$ and $RAND_\Delta$ for $\Delta \leq 1$, will also be unchanged since none of them will wait more than one time unit before sending. Thus, our results hold for this more general type of cost function.

Finally, our algorithms can in principle be built into any operating system, though the ease with which this can be done depends on the exact design of the operating system in question, in particular on the availability of an extra timer to support interrupts from our algorithm. We have concrete plans to try out one of our algorithms by building it into the new Occam Operating System [2].

Acknowledgments

We would like to thank Brian Vinter for drawing our attention to the Packet Bundling Problem and for initial discussions regarding the cost function.

References

[1] Albert Alexandrov, Mihai F. Ionescu, Klaus E. Schauser, and Chris Scheiman. LogGP: Incorporating long messages into the LogP model for parallel computation. *Journal of Parallel and Distributed Computing*, 44(1):71–79, 1997.

[2] Fred Barnes, Brian Vinter, and Peter H. Welch. The Occam Operating System project. Work in progress.

[3] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[4] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken. LogP: Towards a realistic model of parallel computation. In *Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 1–12, 1993.

[5] Daniel R. Dooly, Sally A. Goldman, and Stephen D. Scott. TCP dynamic acknowledgment delay: Theory and practice. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 389–398. ACM Press, 1998.

[6] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal*, 45:1563–1581, 1966.

[7] Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic TCP acknowledgement and other stories about $e/(e-1)$. In *Proceedings of the 33th Annual ACM Symposium on Theory of Computing*, pages 502–509. ACM Press, 2001.

[8] Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.

[9] Thilo Kielmann, Henri E. Bal, and Kees Verstoep. Fast measurement of LogP parameters for message passing platforms. In *Proceedings of the Workshop on Run-Time Systems for Parallel Programming*, pages 1176–1183, 2000. Satelite workshop of the International Parallel and Distributed Processing Symposium.

[10] Lawrence A. H. Liang, Wentong Cai, Bu-Sung Lee, and Stephen J. Turner. Performance analysis of packet bundling techniques in DIS. In *Proceedings of the 3rd International Workshop on Distributed Interactive Simulation and Real-Time Applications*. IEEE Computer Society Press, 1998.

[11] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.