

Admission Control in Deadline-Based Network Resource Management

Yanni Ellen Liu
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada R3T 2N2

Johnny W. Wong
School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

Abstract

In our deadline-based network resource management framework, each application data unit (ADU) is characterized by a (size, deadline) pair, which can be used to convey the resource and QoS requirements of the ADU. Specifically, the ADU's bandwidth requirement can be implicitly estimated by the ratio: $\text{size} / (\text{deadline} - \text{current time})$, and the time at which the ADU should be delivered is specified by the deadline. The ADU deadline is mapped onto deadlines at the network layer, which are carried by packets and used by routers for channel scheduling. In an earlier work, we have shown that deadline-based channel scheduling achieves good performance in terms of the percentage of ADUs that are delivered on-time. However, when a network is under heavy load, congestion may occur, and deadline-based scheduling alone may not be sufficient to prevent performance degradation. The level of congestion can be reduced by admission control. In this paper, two application-layer admission control algorithms are developed. The performance of these two algorithms is evaluated by simulation.

1 Introduction

In recent years, we have seen an increasing demand to transport real-time data over packet-switched networks. Examples of real-time data include stock quote updates, bids in an on-line auction, state updates in a multi-player on-line game, time-sensitive business documents in electronic commerce applications, video and audio data in a video conference, and voice data in IP telephony. To ensure timely delivery of real-time data, quality of service (QoS) support at the transport network is required.

We have developed a deadline-based network resource management framework in order to support real-time document delivery [17]. A document may correspond to a file or a frame in audio or video transport. Each document, or application data unit (ADU) is characterized by a (size, deadline) pair. This characterization conveys both the resource and QoS requirements of an ADU. Specifically, an ADU's bandwidth requirement at the transport network can be implicitly estimated by the ratio: $\text{size} / (\text{deadline} - \text{current}$

$\text{time})$, and the time at which the ADU should be delivered at the receiver is specified by the deadline.

In our framework, the ADU deadline is mapped onto deadlines at the network layer, which are carried by packets and used by routers for channel scheduling. In our earlier work, we have developed an efficient deadline-based channel scheduling algorithm which is superior to FCFS (First-Come First-Served) with respect to the percentage of ADUs that are delivered on-time [15, 14].¹

When network traffic is heavy, congestion may occur; queues at bottleneck links may grow significantly. In this case, channel scheduling alone may not be able to prevent degradation of network performance. Performance degradation can be prevented by admission control. In this paper, we develop two application-layer admission control algorithms. The performance of these two algorithms is evaluated by simulation.

This paper is organized as follows. In Section 2, our performance model is described, and the case of no admission control is studied. The two ADU admission control algorithms that we have developed are presented in Section 3, and simulation results on their performance are reported in Section 4. Section 5 reviews the related literature. Finally, Section 6 contains a summary of our findings and a discussion of future work.

2 Congestion in a Network

In this section, we consider the case of no admission control and investigate the network performance as the load increases. The effectiveness of our admission control algorithms will be evaluated by comparing them to the no admission control case.

We first describe our performance model [14]. At the sender, each generated ADU is characterized by: size, source and destination addresses, deadline, and arrival time. Two types of ADUs are considered: real-time and best-effort. Best-effort ADUs are characterized by a delivery deadline of infinity. Segmentation of an ADU into packets is performed at the sender before the packets are

¹The FCFS algorithm is the scheduling algorithm typically found in networks that provide a best-effort service.

admitted to the source node. The maximum packet size at the network layer is 1500 bytes. Packets are routed through the network until they reach their destination node. They are then delivered to the receiver where packet re-assembly is performed. We assume that fixed shortest-path routing is used and there are no transmission errors. For simplicity, the processing times at the sender and the receiver are not included in our model.

The deadline-based channel scheduling algorithm presented in [14] is used at each network router. This algorithm, referred to as T/H- p , is based on the ratio T/H, where T is the time left (or delivery deadline – current time) and H is the number of hops to destination. T/H is calculated when a packet arrives at a router, it can be viewed as the urgency of a packet; specifically, a packet with a smaller T/H means that it is more urgent. There are three priority queues at each outgoing channel: low, medium and high. Best-effort packets always join the low-priority queue. For real-time packets, a fraction p ($0 \leq p \leq 1$) of the more urgent packets (or packets with smaller T/H values) joins the high-priority queue, and the remaining fraction of $(1 - p)$ joins the medium-priority queue. If a real-time packet is already late ($T \leq 0$) upon arrival at a router, it is downgraded to best-effort rather than being dropped.

For a real-time ADU, the delivery deadline is modeled as follows. Let x be the end-to-end latency when there is no queueing and no segmentation. Also let x_p be the end-to-end propagation delay, y the size of the ADU, and c_j the capacity of the j -th channel along the path based on shortest-path routing. Then x can be estimated by $x = x_p + \sum_j y/c_j$. The allowable delay is assumed to be proportional to x . Hence, the delivery deadline for the ADU is given by $d = arrival\ time + kx$, where k is referred to as a “deadline parameter” ($k > 1$). In general, a smaller k means that the ADU has a more urgent deadline.

A 13-node network model is used in our simulation [15, 14]. Its topology is depicted in Figure 1. The capacity of

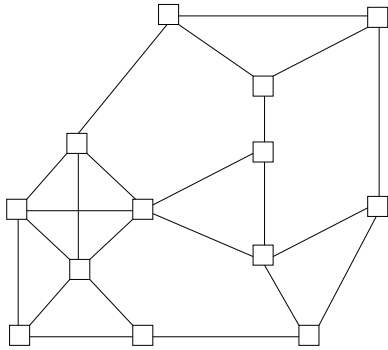


Figure 1: Network model

each channel is assumed to be 155 Mbit/sec. ADUs may

be generated from any point in the network. The ADU interarrival time is assumed to be exponentially distributed, and the aggregate arrival rate is λ (in number of ADUs per second). For each arriving ADU, the source and destination nodes are selected at random. The size of each ADU is assumed to belong to one of two ranges: [500, 1500], and [1500, 500000], in bytes. The first range reflects the sizes of small ADUs, *i.e.*, one packet per ADU. The proportion of small ADUs is kept at 25%. ADU size is assumed to be uniformly distributed within each of these two ranges.

At each outgoing channel, we assume that separate buffers are allocated to real-time traffic and best-effort traffic. Modern routers usually have large buffer sizes that can accommodate between 100 and 200 ms’ worth of data with respect to link capacity [7]. We will consider buffer sizes within this range. For simplicity, we assume that packets dropped due to buffer overflow are not re-transmitted.

The performance measure of interest is throughput. This is defined to be the rate at which real-time ADUs are delivered on time.

The following model parameters were used in our simulation experiments. The ADU arrival rate λ was varied from 200 to 2200 ADUs/sec. For our network model, the bottleneck links are saturated when $\lambda > 1337.5$ ADUs/sec. The deadline parameter k was assumed to be given by $1 + \bar{\epsilon}$ where $\bar{\epsilon}$ is exponentially distributed with mean 0.5. All ADUs generated were of type real-time. For each outgoing channel, the total buffer size was assumed to be 3.1 MByte. This corresponds to 160 ms’ worth of data with respect to the link capacity. 90% of the buffer was allocated to real-time traffic, and the rest to best-effort traffic.

To highlight the effect of congestion, we focus on ADUs that are transmitted over the bottleneck links. In Figure 2, we plot the throughput of these ADUs as a function of

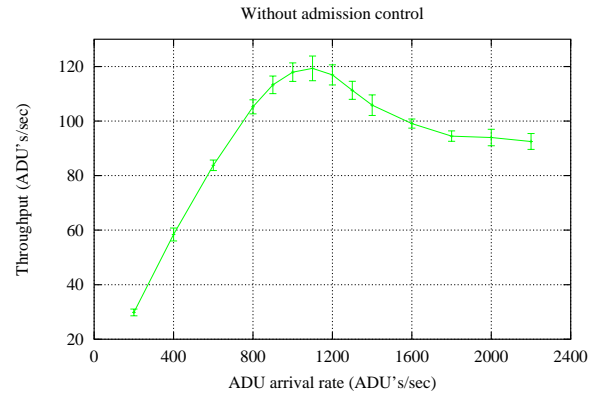


Figure 2: Throughput vs. ADU arrival rate

λ . Both sample means and 99% confidence intervals are shown. It can be observed that, as the load increases, the

throughput continues to increase until $\lambda = 1100$, which corresponds to a utilization of 82% at the bottleneck links. Throughput degradation is observed when λ is increased beyond 1100. When $\lambda > 1337.5$, the system enters an overload condition and throughput continues to suffer.

Throughput degradation can be explained as follows. When the system is at heavy load, the queueing delay at the routers becomes excessive, resulting in more late ADUs. At the same time, buffers start to become full, resulting in dropped packets. This would cause more ADUs to be not delivered on-time. The objective of admission control is to prevent throughput degradation in the presence of increased ADU arrival rates [1]. In Section 4, the effectiveness of our admission control algorithms will be evaluated with respect to this objective.

3 Admission Control Algorithms

In this section, we develop two admission control algorithms within our deadline-based framework. For both algorithms, admission control is performed by an admission control entity residing at the application layer of a sending host. Both algorithms use application-layer feedback information in making admission control decisions. This information is in the form of an acknowledgement (ACK) returned from a receiver indicating whether an ADU was received on-time or not.

Another aspect of our admission control algorithms is that an ADU's destination information is used in admission control decisions. The rationale is as follows. For a network with fixed routing, ADUs to the same destination will traverse the same path inside the network. The admission control entity can therefore estimate the load inside the network for each source-destination pair. If heavy load is imminent, an ADU heading along the same path can be stopped at the source. We now describe our admission control algorithms.

3.1 Algorithm I – Acceptance Probability

Our first admission control algorithm is based on “acceptance probability”. This is the probability that an arriving ADU is admitted. Let $y(src, dst)$ be the acceptance probability for an ADU with sender src and receiver dst . At sender src , the admission control entity maintains an *acceptance probability table*, which contains the $y(src, dst)$'s for all receivers with which the sender is communicating. A receiver, upon receiving an ADU, returns a *positive ACK* to the sender if the ADU is on-time, or it returns a *negative ACK* if the ADU is late.

The Acceptance Probability algorithm consists of two components. The first component is an admission test. This component at sender src is shown in Figure 3. An arriving ADU destined to dst is admitted with probability $y(src, dst)$. If an ADU is not admitted, $y(src, dst)$ is increased by ΔI . In other words, a small credit is awarded if

```

/* At src, upon the arrival of a sending ADU W */
Get its destination dst;
Retrieve y(src,dst) from acceptance_probability_table;
Obtain a random number u in [0, 1];
If ( u <= y(src,dst) )
    W is accepted;
Else
    /* W is not accepted */
    y(src,dst) = y(src,dst) + ΔI;
    If ( y(src,dst) > 1 ) y(src,dst) = 1;

```

Figure 3: Algorithm I: admission test component

an ADU was not accepted. The last line in Figure 3 ensures that an acceptance probability is never larger than one.

The second component is concerned with the adjustment of acceptance probability. The details are shown in Figure 4. At a sender src , upon receiving an ACK from dst , the acceptance probability for this destination is increased by ΔI if the ACK is positive, and decreased by ΔD if the ACK is negative. At no time is the acceptance probability less than or equal to ΔI . This will ensure that

```

/* Adjustment of acceptance probability at src */
All acceptance probabilities are initialized to 1

/* Upon receiving an ACK from dst */
If a positive ACK
    y(src,dst) = y(src,dst) + ΔI ;
    If ( y(src,dst) > 1 ) y(src,dst) = 1;
If a negative ACK
    y(src,dst) = y(src,dst) - ΔD ;
    If ( y(src,dst) <= 0 ) y(src,dst) = ΔI ;

```

Figure 4: Algorithm I: parameter adjustment

transmission to this destination is never blocked, so that some ADUs and their corresponding ACKs may still get through. ΔI and ΔD are tunable parameters for this algorithm.

3.2 Algorithm II – Estimated Bandwidth

Our second admission control algorithm is based on “estimated bandwidth”. In addition to ACKs, the bandwidth requirement of the arriving ADU as well as that of all “outstanding” ADUs are used in admission control decisions. Outstanding ADUs are those that have been admitted but not yet acknowledged, and whose deadlines have not expired. Let z , d , and a denote the size, deadline, and arrival time of an ADU, the bandwidth requirement of this

ADU can be estimated by:

$$c = \frac{z}{d - a}. \quad (1)$$

At each sender *src*, a *committed bandwidth table* is maintained for each receiver. The format of this table is illustrated in Table 1. Each outstanding ADU has an entry

Table 1: Committed bandwidth table

ADU_id	Deadline	Estimated Bandwidth
1	d_1	$c_1 = z_1 / (d_1 - a_1)$
2	d_2	$c_2 = z_2 / (d_2 - a_2)$
3	d_3	$c_3 = z_3 / (d_3 - a_3)$
4	d_4	$c_4 = z_4 / (d_4 - a_4)$
...

in the table. For ADU *i*, the entry contains its deadline d_i and its estimated bandwidth requirement c_i .

The Estimated Bandwidth algorithm also consists of two components. The first component is again an admission test. The details of this component are shown in Figure 5. Upon the arrival of a real-time ADU, the committed

```

/* Upon the arrival of a sending ADU W (z,d,a) */
Get destination dst and dst's committed bandwidth table;
Delete all obsolete ADU's (entries) from table;
Let c = z / (d - a);
If ( c < Cmin )
    W is accepted;
    Add ADU W to table;
Else
    Let tmp = c + Σci, for all entries in table;
    If ( tmp < Cmax )
        W is accepted;
        Add ADU W to table;
    Else
        W is not accepted;

```

Figure 5: Algorithm II: admission test component

bandwidth table for its destination is retrieved. This table is first scanned and all ADUs with expired deadlines are removed. The bandwidth requirement of the arriving ADU (denoted by c) is then calculated using Equation (1). If c is smaller than C_{min} , the ADU is accepted for transmission without further consideration, and its estimated bandwidth as well as its deadline are entered into the committed bandwidth table. The idea here is that ADUs with a very small bandwidth requirement are always admitted. If $c \geq C_{min}$, the sum of c and the total bandwidth requirement of all

outstanding ADUs in the table is calculated. If this sum is less than C_{max} , the arriving ADU is admitted, otherwise it is rejected. C_{max} serves as an upper bound on committed bandwidth for this destination.

The second component is concerned with updates to C_{max} . The details are shown in Figure 6. Initially, C_{max} is

```

/* Adjustment of Cmax */
Cmax = Cmax0 at algorithm initialization

/* Upon receiving an ACK for ADU W */
Delete entry for W from table if it still exists;
If a positive ACK
    Cmax = Cmax + ΔI;
    If ( Cmax > Cmax0 ) Cmax = Cmax0;
If a negative ACK
    Cmax = Cmax - ΔD;
    If ( Cmax < Cmin ) Cmax = Cmin;

```

Figure 6: Algorithm II: parameter adjustment

set to a pre-specified upper bound C_{max0} . When an ACK for an ADU is returned from a receiver, the committed bandwidth table for that receiver is retrieved. The table entry corresponding to the acknowledged ADU is removed. The value of the parameter C_{max} is also adjusted. Specifically, C_{max} is increased by ΔI if the ACK is positive, and decreased by ΔD if the ACK is negative. At no time can C_{max} be larger than the pre-specified bound C_{max0} . Also, it can never be smaller than C_{min} .

4 Performance Evaluation

Simulation is used to evaluate the performance of the two admission control algorithms. Admission control is implemented in our model as follows. At the sender, upon the arrival of an ADU, an admission test, as specified by the admission control algorithm, is performed to determine whether the ADU is to be admitted or not. At the receiver, an ACK is generated and returned to the sender under the following two conditions: (i) if all packets of an ADU are received on-time, a positive ACK is returned when the last packet of this ADU has been received; or (ii) if one or more packets of an ADU are received after their deadline, a negative ACK is returned to the sender when the first such packet has been received. Note that it is also possible that no packets of an ADU ever arrive at the receiver because of buffer overflow. In this case, no ACK will be returned to the sender. For simplicity, we assume that such an ADU is simply lost and lost ADUs are not re-transmitted.

For convenience, the details of transmission of ACKs are not modeled. Instead, an ACK is assumed to arrive at the sender T_{ACK} seconds after it has been sent. T_{ACK} is

determined by: $T_{ACK} = prop_delay + ACK\ tran_delay$, where $prop_delay$ is the propagation delay along the shortest path from receiver to sender, and $ACK\ tran_delay$ is given by the sum of queuing delay and transmission time. $ACK\ tran_delay$ is assumed to be uniformly distributed between $[0.8, 20]$ ms.

For Algorithm I, ΔI and ΔD are tunable parameters. We assume that at each sender, ΔI is set to the same value for all destinations; similarly for ΔD . Similar assumptions are made for the tunable parameters in Algorithm II, namely C_{min} , C_{max0} , ΔI , and ΔD .

A good admission control algorithm should not only prevent performance degradation at heavy load, but also not be overly restrictive when the load is light. We therefore evaluate the performance of our algorithms under both light-load and heavy-load scenarios.

4.1 Algorithm I – Acceptance Probability

For Algorithm I, admission control becomes more restrictive when ΔD is increased. This is due to the fact that a late ADU would result in a larger fraction of ADUs being rejected. A larger ΔD should therefore be used when the load is heavy. Conversely, a larger ΔI means that the control is less restrictive, and such an adjustment is appropriate when the load is light.

We again focus on ADUs that are transmitted over bottleneck links. In Figure 7, the throughput of these ADUs

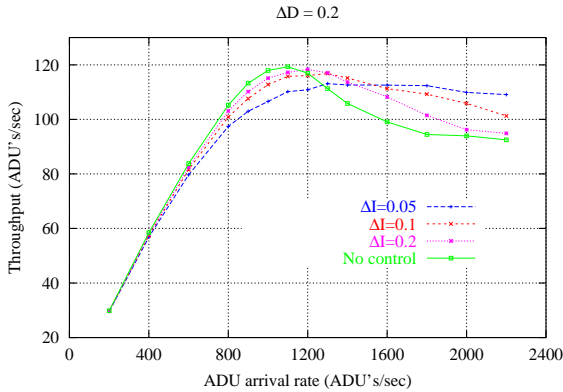


Figure 7: Effect of ΔI in Algorithm I

is plotted against the ADU arrival rate λ for different values of ΔI . ΔD is kept at 0.2. The corresponding results for the case of no admission control are included for comparison (these results are taken from Figure 2). For convenience, only sample means are shown. It can be observed that when $\lambda > 1600$, a larger ΔI results in worse performance than a smaller ΔI . This is a consequence of the admission control being not very restrictive.

In Figure 8, we plot the corresponding throughput results for different values of ΔD . ΔI is fixed at 0.05. It can

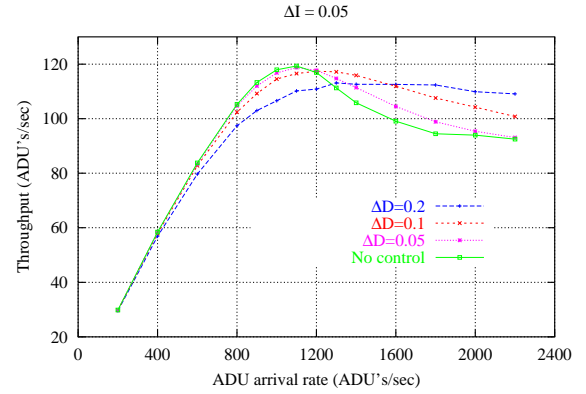


Figure 8: Effect of ΔD in Algorithm I

be observed that a larger ΔD results in better performance when the load is heavy. This is consistent with our earlier comment that at heavy load, a larger ΔD should be used.

We also observe from Figures 7 and 8 that, when compared with no admission control, our admission control algorithms result in (i) noticeable throughput improvement at heavy load (when $\lambda > 1300$), (ii) some throughput degradation at medium load (when $600 < \lambda < 1300$), and (iii) very little throughput degradation at light load ($\lambda < 600$). This is a desirable behavior for a network that employs admission control.

We have also investigated the issue of how to choose ΔI and ΔD , and found that system performance is affected by the ratio $r_d = \Delta D / \Delta I$ [14]. Furthermore, we found that a relatively large value of $r_d = \Delta D / \Delta I$, e.g., $r_d \approx 4$, should be used when the load is heavy, and at light load, a small r_d , e.g., $r_d \approx 1$ should be used. In either case, ΔI should be relatively small (e.g., in the range of 0.025 to 0.05) to avoid large fluctuations in acceptance probability.

We conclude that even with a simple admission control algorithm such as our Algorithm I, throughput improvement can be realized at heavy load when compared to the case of no admission control.

4.2 Algorithm II – Estimated Bandwidth

Our second admission control algorithm, Estimated Bandwidth, has four tunable parameters: C_{min} , C_{max0} , ΔI , and ΔD . All four parameters are in terms of bandwidth, measured in bits per second.

C_{min} is a threshold where ADUs with estimated bandwidth smaller than C_{min} are always accepted. Intuitively, the value of C_{min} should correspond to a small fraction of the bottleneck bandwidth, otherwise too much traffic may be admitted into the network, jeopardizing the purpose of admission control. In our network model, all links have capacities of 155 Mbps. Three values of C_{min} , 5, 20, and 30 Mbps, were investigated in our simulation experiments.

For each sender-receiver pair, C_{max0} represents the maximum rate at which ADUs with estimated bandwidth $> C_{min}$ are admitted. In our experiments, C_{max0} was selected to be 120 Mbps, which corresponds to approximately 80% of link capacity. ΔI and ΔD should be small and were selected to be 5 and 20 Mbps respectively, corresponding to a ratio of 4 for $\Delta D/\Delta I$.

In Figure 9, the throughput of ADUs that traverse bot-

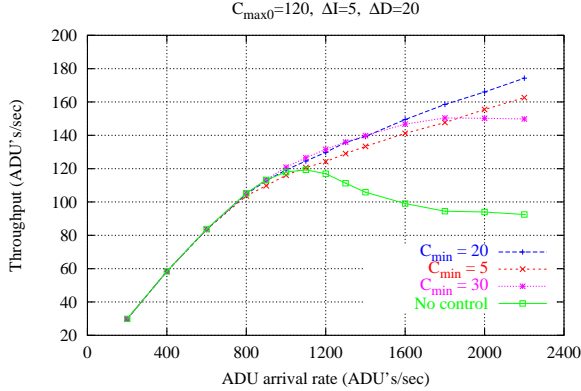


Figure 9: Effect of C_{min} in Algorithm II

tleneck links is plotted against the ADU arrival rate. The corresponding results for the case of no admission control are also shown. It can be observed that all three values of C_{min} resulted in much improved performance over no admission control when $\lambda > 1200$ ADUs/sec. In addition, there was minimal performance degradation at medium load, and practically no degradation at light load. Our results also suggest that a value of C_{min} between 5 and 20% of link capacity would result in good performance in most cases.

We next consider how one may select the value of C_{max0} . Experiments for two values of C_{max0} were performed (50 and 120 Mbps). The values of the other parameters are: $C_{min} = 20$ Mbps, $\Delta I = 5$ Mbps, and $\Delta D = 20$ Mbps. In Figure 10, the throughput at bottleneck links is plotted against the ADU arrival rate. It can be observed that at light load, $C_{max0} = 50$ is more restrictive, and as a result, a lower throughput is realized than when $C_{max0} = 120$. On the other hand, $C_{max0} = 120$ does not perform as well as $C_{max0} = 50$ when the load is heavy ($\lambda > 1200$). This is due to the fact that at $C_{max0} = 120$, the bottleneck links are congested, leading to fewer ADUs being delivered on-time. Further experiments indicated that a good selection of C_{max0} is 50% of link capacity [14].

As to the values of ΔI and ΔD , we found that, at light load, both ΔI and ΔD do not have significant impact on performance. However, at heavy load, a small ΔI (e.g., less than 5% of link capacity) should be used, and ΔD should

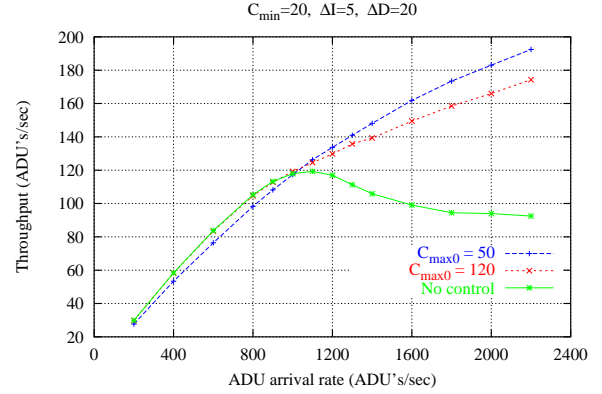


Figure 10: Effect of C_{max0} in Algorithm II

be larger than ΔI . A reasonable selection is such that the ratio $\Delta D/\Delta I$ is between 4 and 8.

4.3 Comparison of Algorithms I and II

The results obtained in the last two subsections indicate that for the scenarios considered, Algorithm II (Estimated Bandwidth) is more effective than Algorithm I (Acceptance Probability). This is further supported by the results shown in Figure 11 where we present the best performance

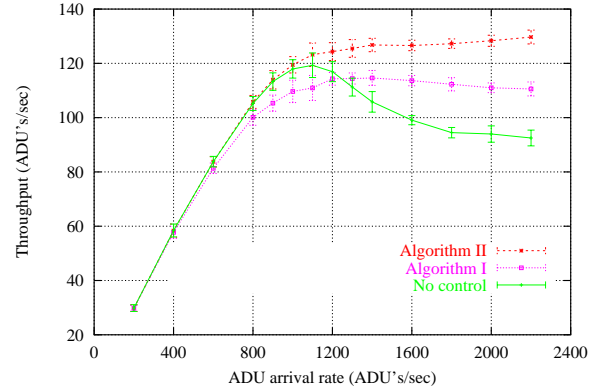


Figure 11: Algorithm I vs. Algorithm II

of Algorithm I and the worst performance of Algorithm II among all scenarios that we have investigated [14]. Note that Algorithm II makes use of the bandwidth requirement of an arriving ADU, as given by size / (deadline - current time), as well as the bandwidth requirements of outstanding ADUs in making admission control decisions. This additional information may have allowed the admission control entity to make more informed decisions.

4.4 Effect of ACK Transmission Delay

Our results so far are based on the assumption that the ACK transmission delay is uniformly distributed within the

range [0.8, 20] ms. This corresponds to a relatively fast return of the ACK. Within our deadline-based framework, each ACK is sent like an ADU with an appropriately specified deadline. An ACK may therefore experience a longer transmission delay when the corresponding packet is transmitted at the network layer. To study the effect of ACK transmission delay, we assume that this delay is uniformly distributed within the range of [0.8, x] ms where x takes on the values of 10, 20, 60, 100, and 150. Simulation results on the throughput at bottleneck links for Algorithms I and II are shown in Figures 12 and 13 respectively. Four val-

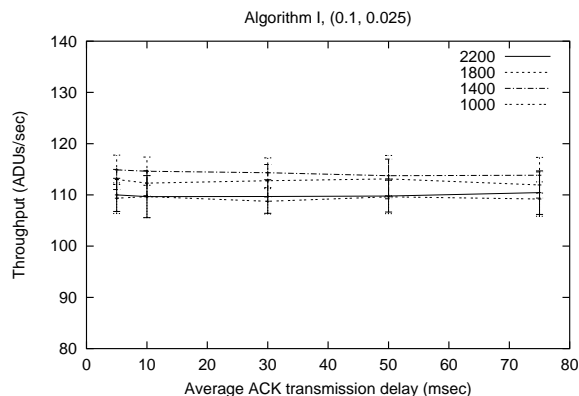


Figure 12: Algorithm I: Effect of ACK transmission delay

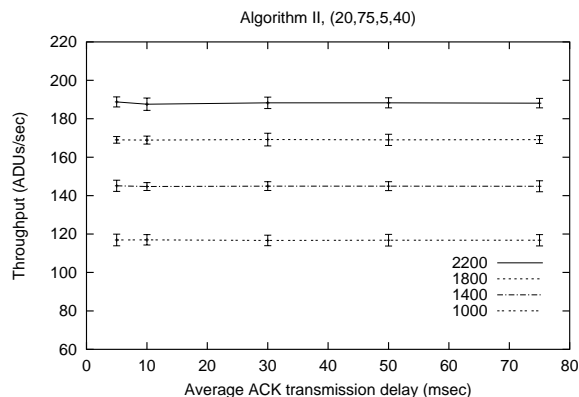


Figure 13: Algorithm II: Effect of ACK transmission delay

ues of the ADU arrival rate λ were considered. They are 1000, 1400, 1800, and 2200 ADUs/sec. Also, for Algorithm I, ΔI and ΔD are set to 0.025 and 0.1 respectively, while for Algorithm II, the parameter settings are: $C_{min} = 20$ Mbps, $C_{max0} = 75$ Mbps, $\Delta I = 5$ Mbps, and $\Delta D = 40$ Mbps.

It is observed that for both admission control algorithms, the performance is not sensitive to the ACK trans-

mission delay. In other words, promptness of ACKs does not have much impact on performance. This is a desirable feature when one considers the robustness of our algorithms under a range of traffic conditions.

5 Related Work

Admission control for packet-switched networks has been a popular subject of research. Admission control schemes at both the network layer and the application layer have been studied.

At the network layer, many schemes have been proposed for call admission control in ATM networks (see [16] for a review of such schemes) and for QoS provisioning in IP-based networks (see for example [8, 10, 4, 5, 2, 13]). Network-layer admission control typically adopts a connection-oriented approach. Before data transmission, a signaling protocol is used to set up a connection along a flow path. During connection setup, each router along the path determines if the newly arriving flow with given traffic specification and QoS requirements can be accepted without violating the QoS commitment that has been made to already accepted flows. In contrast, our admission control algorithms reside at the application layer, the admission decisions are made at end-systems rather than at the routers along a flow path. There is no notion of flow at the network layer in our framework.

The literature in application-layer admission control includes end-to-end measurement-based admission control (EMBAC) [3, 6, 12] and distributed admission control [9, 11]. In EMBAC schemes, before transmission of user data, an end-system first sends probe packets to the receiver. The receiver observes the performance experienced by these probe packets and returns a report to the sender. The sender uses these reports to infer the level of congestion in the network and makes admission decisions accordingly. In distributed admission control, packets are “marked” by routers when approaching overload. The receiver returns these marks to the sender. The sender decides whether or not to accept a new call according to its recent experience of marking probabilities. Probing and marking by routers are not used in our admission control schemes.

6 Summary

Two application-layer ADU admission control algorithms have been developed and evaluated within the deadline-based network resource management framework. We found that both algorithms can be used to prevent performance degradation at heavy load. Between the two algorithms, Algorithm II, which makes use of ADU bandwidth requirements, yields better performance. This indicates that the availability of ADU bandwidth requirement information allows the admission control entity to

make more informed decisions. As a direction for future research, more sophisticated admission control algorithms can be considered. For example, if an ADU does not pass an admission test, rather than being treated as rejected, the admission control entity may give a “hint” about when to re-submit the ADU, or it may suggest a different but more feasible deadline.

Acknowledgment

This work was supported by the Natural Sciences and Engineering Research Council of Canada.

References

- [1] Dimitri Bertsekas and Robert Gallager. *Data networks*. Prentice Hall, second edition, 1992.
- [2] S. Bhatnagar and B. R. Badrinath. Distributed Admission Control to Support Guaranteed Services in Core-Stateless Networks. In *Proc. of IEEE Infocom'2003*, 2003.
- [3] G. Bianchi, A. Capone, and C. Petrioli. Throughput analysis of end-to-end measurement-based admission control in IP. In *Proc. of IEEE Infocom'2000*, pages 1461–1470, March 2000.
- [4] L. Breslau, S. Jamin, and S. Shenker. Measurement-based admission control: what is the research agenda? In *Proc. IWQOS*, pages 3–5, London, UK, May 1999.
- [5] L. Breslau, S. Jamin, and S. Shenker. Comments on the performance of measurement-based admission control algorithms. In *Proc. of IEEE Infocom'2000*, pages 1233–1242, March 2000.
- [6] V. Elek, G. Karlsson, and R. Ronngren. Admission control based on end-to-end measurements. In *Proc. of IEEE Infocom'2000*, pages 623–630, March 2000.
- [7] Internet end-to-end interest mailing list. Queue size of routers. <http://www.postel.org/pipermail/end2end-interest/2003-January/>.
- [8] R. J. Gibbens and F. P. Kelly. Measurement-based connection admission control. In *Proc. of 15th International Teletraffic Congress*, June 1997.
- [9] R. J. Gibbens and F. P. Kelly. Distributed connection acceptance control for a connectionless network. In P. Key and D. Smith, editors, *Teletraffic Engineering in a Competitive World, Proc. of the International Teletraffic Congress (ITC-16)*, pages 941–952. Elsevier, Amsterdam, 1999.
- [10] S. Jamin, P. B. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. *IEEE/ACM Transactions on Networking*, 5(1):56–70, February 1997.
- [11] F. P. Kelly, P. B. Key, and S. Zachary. Distributed admission control. *IEEE Journal on Selected Areas in Communications*, 18:2617–2628, 2000.
- [12] P. Key and L. Massouli. Probing strategies for distributed admission control in large and small scale systems. In *Proc. of IEEE Infocom'2003*, 2003.
- [13] T. K. Lee, M. Zukerman, and R. G. Addie. Admission Control Schemes for Bursty Multimedia Traffic. In *Proc. of IEEE Infocom'2001*, 2001.
- [14] Y. E. Liu. *Deadline based network resource management*. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 2003.
- [15] Y. E. Liu and J. W. Wong. Deadline based channel scheduling. In *Proc. of the IEEE Globecom'2001*, pages 2358–2362, San Antonio, Texas, November 2001.
- [16] H. Perros and K. Elsayed. Call admission control schemes: A review. *IEEE Communications Magazine*, 34:82–91, November 1996.
- [17] J. W. Wong and Y. E. Liu. Deadline based network resource management. In *Proc. of ICCCN'2000*, pages 264–268, 2000.