Poster: ACP: Age Control Protocol for Minimizing Age of Information over the Internet

Tanya Shreedhar
Wireless Systems Lab, IIIT-Delhi
tanyas@iiitd.ac.in

Sanjit K. Kaul
Wireless Systems Lab, IIIT-Delhi
skkaul@iiitd.ac.in

Roy D. Yates
WINLAB, Rutgers University
ryates@winlab.rutgers.edu

ABSTRACT

Real-time monitoring is characterized by a source repeatedly sending updates over the Internet to a monitor, which desires the sensed information at it to be as fresh (of small age) as possible, given network constraints. We propose the Age Control Protocol (ACP), which, in a network-transparent manner, enables a source to keep the age at the monitor small. We evaluate it using simulations and real-world experiments.

ACM Reference Format:
Tanya Shreedhar, Sanjit K. Kaul, and Roy D. Yates. 2018. Poster: ACP: Age Control Protocol for Minimizing, Age of Information over the Internet. In *The 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18), October 29-November 2, 2018, New Delhi, India.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3241539.3267740

1 INTRODUCTION

Real-time monitoring applications span across areas like health care and natural environment monitoring. They are distinct from that of file transfer and real-time voice/video. Such applications have a sensor/Internet-of-Things (IoT) device (*source*) that sends sensed information (*update*) to a *monitor*. The monitor desires to have the most currently sensed information at any given time. However, the same may not be feasible given constraints imposed by the communications network. To elucidate, let's say the source sends fresh updates at a rapid rate. However, the faster the rate of updates, the larger the throughput, and the larger is the average delay (Figure 1) with which an update is received at the monitor. This is explained by the larger resulting congestion in the network. As a result, the freshest update at the monitor at any given time will be stale, that is of large age. Figure 1 broadly captures the behavior of the metrics of throughput, delay, and age. *The throughput increases linearly in the rate of updates. This leads to an increase in average packet delay. Large packet delays coincide with large average age.* Large age is also seen for small throughputs (correspondingly, small rate of updates). *At a low rate the monitor receives updates infrequently, and this increases the average age (staleness) of its most fresh update.* Finally, observe that there exists a rate (and corresponding throughput) at which age is minimized.

ACM ISBN 978-1-4503-5903-0/18/10.
https://doi.org/10.1145/3241539.3267740

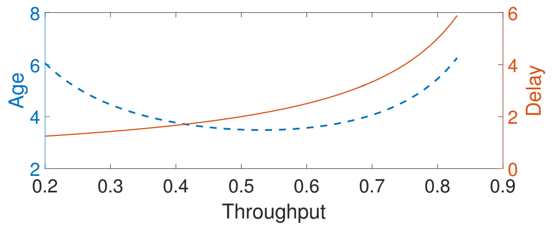


Figure 1: Example interplay of the networking metrics of delay (solid line), throughput (normalized by service rate) and age. Shown for a M/M/1 queue with service rate of 1.

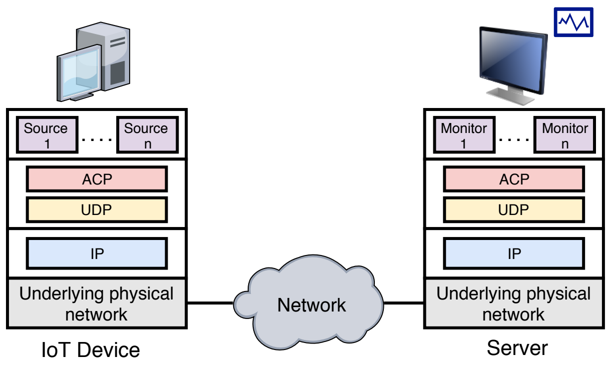


Figure 2: The ACP end-to-end connection.

We propose the *Age Control Protocol* that resides in the transport layer and operates only on the end hosts. Figure 2 shows an *end-to-end connection* between two hosts, the IoT device, and the server, over the Internet. Sources at a device open ACP connections to their monitors. ACP belongs to the transport layer of the TCP/IP networking stack. Akin to the real-time transport protocol (RTP) [3] that supports voice/video applications, ACP also uses UDP for sending updates generated by the sources. Thus ACP doesn't guarantee delivery of a source update. Inline with the goal of ACP to optimize freshness, sending a new update is better than retransmitting an older one.

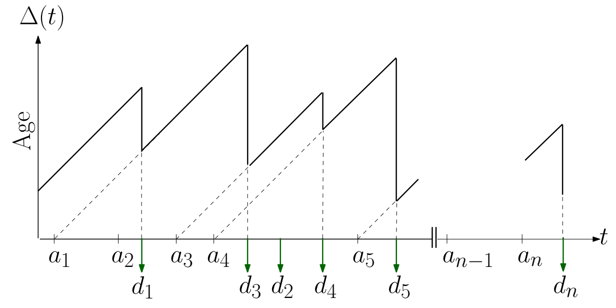


Figure 3: A sample function of the age process $\Delta(t)$. Updates are indexed $1, 2, \ldots, n$. Also, a_i is the time at which update i was sent by the source and d_i is the time the update was received at the corresponding monitor.

ACP suggests to a source the rate at which it must send fresh updates such that the average age at the monitor is minimized. ACP adapts its suggestion to the perceived congestion in the Internet. By suggesting this desired rate to a source, ACP also limits congestion that may be introduced by an unnecessarily fast rate of updating by sources.

Let $u(t)$ be the time at which the freshest update at the monitor at time t was sensed at the source. The age $\Delta(t)$ at the monitor is the age $t - u(t)$ of its freshest update at time t. An example sample function of the age process is shown in Figure 3. Age at the monitor increases linearly in the absence of updates. On reception of an update that is fresher than updates received in the past, the age at the monitor is reset to the age of the update ($d_i - a_i$ for update i in Figure 3), which is simply the time elapsed between its generation at the source and its reception at the monitor. We want to choose the rate λ (updates/second) that minimizes the expected value $\lim_{t\to\infty} E[\Delta(t)]$ of age at the monitor, where the expectation is over any randomness in the network. The expectation is not usually known and must be estimated using measurements. Also, we would like to dynamically adapt this λ to nonstationarities in the network.

Over the last few years, various works [1], [4] have analyzed age under various network model assumptions. Recently in [2], authors use deep Q-learning to optimize age over a given network topology unknown a priori.

2 WORKINGS OF THE ACP

An update sent by the source ACP contains a timestamp a_i that is the time update i was sent (see Figure 3). A more recent timestamp indicates a fresher update. The monitor ACP sends an ACK packet in response to every received update that is fresher than the previously received updates. An ACK for update i contains a_i. ACK(s) that are received in a delayed manner that is after an ACK of a fresher update has been received, are discarded. ACP doesn't assume any time synchronization between the source and the monitor.

Network	Age		λ		Backlog	
Topology	Opt	ACP	Opt	ACP	Opt	ACP
1	0.0143	0.0152	110	107.1	1.07	1.10
1-1	0.0244	0.0250	110	106.9	2.1	2.12
1-5	0.0165	0.0172	110	105.0	1.28	1.28
1-1-1	0.0341	0.0347	110	102.85	3.2	3.0
1-1-5	0.0263	0.0268	110	105.8	2.36	2.26
1-5-5	0.0185	0.0191	110	104.57	1.50	1.49

Table 1: Results from simulations. Step-size $\kappa = 0.1$.

An ACP connection (see Figure 4a) begins with the source opening a UDP socket with the monitor's advertised IP address and port. ACP then sends a few updates to estimate the round-trip-time (RTT) of the connection and sets the initial rate to 1/RTT. Post this, the timeline can be viewed as a sequence of control epochs indexed $1, 2, \ldots$. Epoch k begins at time t_k and ends at t_{k+1}. At t_k, the update rate is set to λ_k. Source updates once every λ_k^{-1} during epoch k.

Let $\overline{\Delta}_k$ be the estimate at the source ACP of the time average of the age of updates at the monitor over (t_{k-1}, t_k). To calculate this average, the source ACP must construct its estimate of the age sample function (see Figure 3), over the interval, at the monitor. It knows the time a_i a source sent a certain packet i. However, it needs the time i was received by the monitor, which it approximates by the time an ACK for the packet i was received. On receiving the ACK, it resets its estimate of age to the RTT of i.

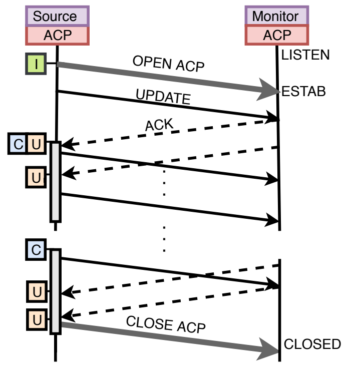
Let $\overline{B}_k$ be the time average of backlog calculated over (t_{k-1}, t_k). The backlog increases by 1 when the source sends a new update. When an ACK corresponding to an update i is received, update i and any unacknowledged updates older than i are removed from the current backlog count.

We state without detail that in addition to $\overline{\Delta}_k$ and $\overline{B}_k$ the source ACP also maintains moving averages $\overline{Z}$ of the inter-update reception time at the monitor and $\overline{\text{RTT}}$ and adapts the length $\overline{T}$ of an epoch as an integral multiple of $\mathcal{T} = min(\overline{Z}, \overline{\text{RTT}})$. At start of control epoch k, at time t_k, the source ACP calculates the changes $\delta_k = \overline{\Delta}_k - \overline{\Delta}_{k-1}$ and $b_k = \overline{B}_k - \overline{B}_{k-1}$ in average age and backlog, respectively.

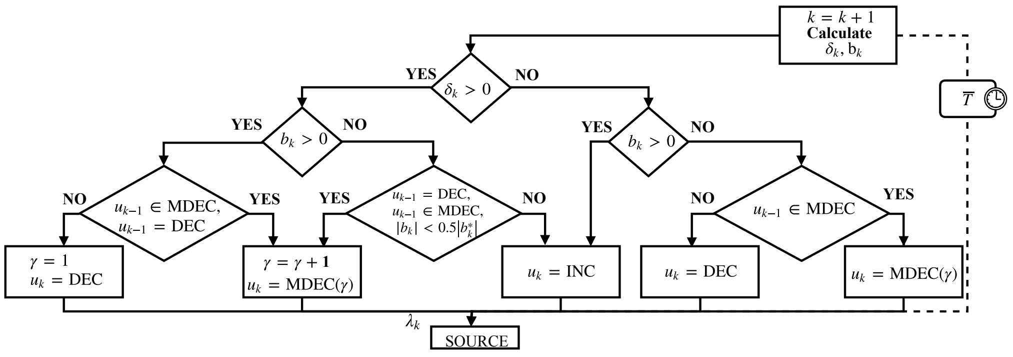
ACP at the source chooses an action u_k at t_k that targets a change b^*_{k+1} in average backlog over (t_k, t_{k+1}) with respect to (t_{k-1}, t_k). The actions, may be broadly classified into (a) additive increase (INC), additive decrease (DEC), and multiplicative decrease (MDEC). Here MDEC corresponds to a set of actions MDEC(γ), where $\gamma = 1, 2, \ldots$. We have

$$\text{INC: } b^*_{k+1} = \kappa;\ \text{DEC: } b^*_{k+1} = -\kappa;\ \text{MDEC}(\gamma)\text{: } b^*_{k+1} = -(1 - 2^{-\gamma})B_k,$$

where $\kappa > 0$ is a step-size parameter, which we choose empirically. ACP attempts to achieve b^*_{k+1} by setting λ_k appropriately. It can be shown that (we skip details), $\lambda_k = (1/\overline{Z}) + (b^*_{k+1}/\mathcal{T})$. Figure 4b summarizes how ACP chooses its action u_k as a function of b_k and δ_k.



(a)



(b)

Figure 4: (a) Timeline of an ACP connection. The boxed C denotes the ACP algorithm (Fig. 4b) executed when a new control epoch begins. The boxed U is executed when an ACK is received and updates $\overline{Z}$, $\overline{\text{RTT}}$, and $\mathcal{T}$. (b) Control algorithm at the source.

The source ACP targets a reduction in average backlog over the next epoch in case (a) $b_k > 0, \delta_k > 0$ or (b) $b_k < 0, \delta_k < 0$. The first condition indicates that the update rate is such that update packets are experiencing larger than optimal delays. ACP attempts to reduce backlog multiplicatively to reduce congestion delays and in the process reduce age quickly. Every consecutive occurrence of this case (tracked by increasing γ by 1 every time) attempts to decrease backlog even more aggressively, which is by a larger power of 2.

The second condition $b_k < 0, \delta_k < 0$ captures a reduction in both age and backlog. ACP greedily aims at reducing backlog further hoping that age will reduce too. It attempts multiplicative decrease if the previous action did so. Else, it attempts an additive decrease.

The source ACP targets an increase in average backlog over the next control epoch in case (a) $b_k < 0, \delta_k > 0$ or (b) $b_k > 0, \delta_k < 0$. The first condition hints at too low an update rate causing an increase in age. So, ACP additively increases backlog. On the occurrence of the second condition ACP greedily increases backlog.

When the condition $b_k < 0, \delta_k > 0$ occurs, ACP checks if the previous action attempted to reduce the backlog. If yes, and if the actual change in backlog was much smaller than the desired, ACP reduces backlog multiplicatively. This helps counter situations where the increase in age is in fact because of increasing congestion.

3 RESULTS AND FUTURE WORK

Simulation Results: Our source sent constant sized (1024 bytes) updates to a monitor 1 to 3 hops away. Each hop was assigned a link rate of either 1 or 5 Mbps. To exemplify, in Table 1, 1-1-5 corresponds to a network with three hops, where the first and second hops have rates of 1 Mbps and the last hop (to the monitor) has a rate of 5 Mbps. The table compares the average age, backlog, and update rate λ to which ACP converges with those found to be optimal (Opt) via Monte-Carlo simulations over a range of rates λ.

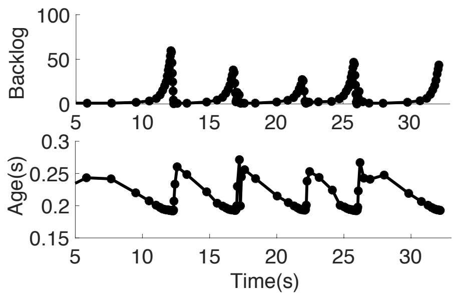


Figure 5: Sample average backlog and age obtained using ACP, for $\kappa = 1$, in a real-world experiment over the Internet.

Real-World Experiments: The source sent updates to a monitor over the Internet (15-20 hops). We did about 50 experiments of 1000 packets each, during different times of the day. Figure 5 illustrates backlog and age as a function of time. ACP increases backlog conservatively while this results in reduction of age and starts decreasing backlog aggressively once age increases.

In the future, we plan on simulating significantly more complex network topologies including multiple routes to the monitor, packet errors due to congestion and link errors, random update packet sizes, and presence of other traffic flows. In addition, we are working toward a better analytic understanding of age control in the Internet.

REFERENCES

[1] Maice Costa et al. 2016. On the age of information in status update systems with packet management. *IEEE Transactions on Information Theory* 62, 4 (2016), 1897–1910.
[2] E. Sert et al. 2018. Optimizing age of information on real-life TCP/IP connections through reinforcement learning. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*. 1–4.
[3] Henning Schulzrinne et al. 2003. *RTP: A transport protocol for real-time applications*. Technical Report.
[4] Yin Sun et al. 2017. Update or wait: How to keep your data fresh. *IEEE Transactions on Information Theory* 63, 11 (2017), 7492–7508.