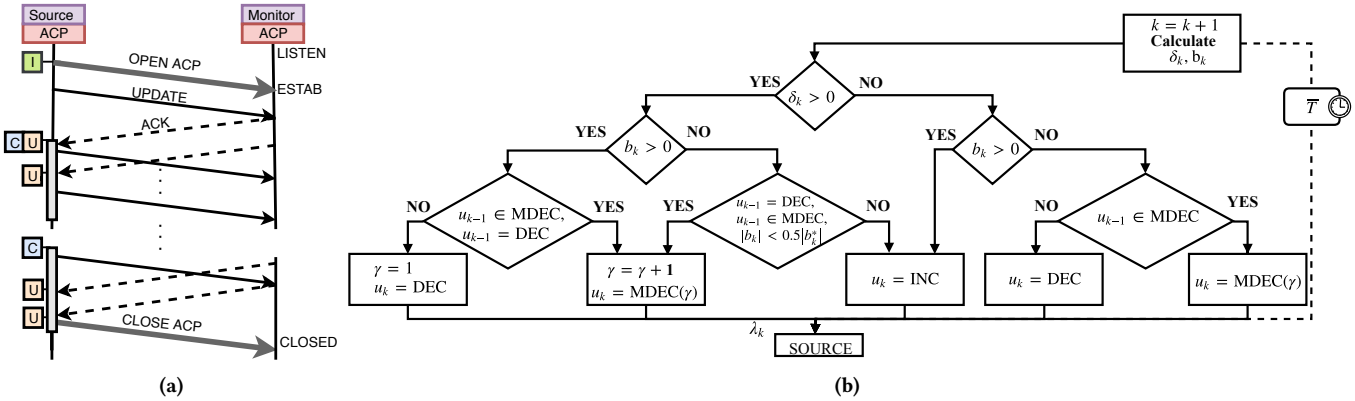

Figure 4: (a) Timeline of an ACP connection. The boxed C denotes the ACP algorithm (Fig. 4b) executed when a new control epoch begins. The boxed U is executed when an ACK is received and updates $\overline{Z}$, $\overline{\mathbf{RTT}}$, and $\mathcal{T}$. (b) Control algorithm at the source.

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.

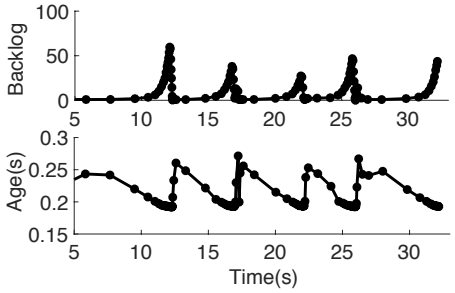

Figure 5: Sample average backlog and age obtained using ACP, for $\kappa = 1$, in a real-world experiment over the Internet.

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.