On the Relevance of Adversarial Queueing Theory in Practice

Daniel S. Berger
Distributed Computer Systems (DISCO) Lab
University of Kaiserslautern
Germany
berger@cs.uni-kl.de

Martin Karsten
David R. Cheriton School of Computer Science
University of Waterloo
Canada
mkarsten@uwaterloo.ca

Jens Schmitt
Distributed Computer Systems (DISCO) Lab
University of Kaiserslautern
Germany
jschmitt@cs.uni-kl.de

ABSTRACT

Adversarial Queueing Theory (AQT) has shown that seemingly innocent traffic injection rates might lead to unbounded queues in packet-switched networks - depending on scheduling strategies as well as topological characteristics. Little attention has been given to quantifying these effects in realistic network configurations. In particular, the existing AQT literature makes two unrealistic assumptions: infinite buffers and perfect synchrony. Because finite buffers inherently limit queue sizes, adversarial effects ultimately lead to packet loss which we address in this work. In addition, we study the effect of imperfect network synchronization under the packet loss metric. Our results, using analysis and simulation, indicate that classical AQT examples appear harmless under realistic assumptions but for a novel class of adversaries considerably higher loss can be observed. We introduce this class by giving examples of two new AQT concepts to construct loss-efficient network adversaries. Our analysis proves the robustness of these new adversaries against randomized de-synchronization effects in terms of variable link delays and nodal processing.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network topology; C.4 [**Performance of Systems**]: Modeling techniques

Keywords

Adversarial Queueing Theory; network stability; finite buffers

1. INTRODUCTION

Adversarial Queueing Theory (AQT) [7] has been introduced to analyze the inherent stability characteristics of network topologies assuming certain scheduling policies. In particular, it has been shown for FIFO scheduling that seemingly innocent continuous packet injection strategies, where the aggregated arrival rate of requests for each link does not exceed the link capacity, can lead to unbounded queue lengths and thus, unbounded delay. Such a network configuration is termed *unstable*.

In reality, such patterns can be caused by misconfiguration, or unfortunate circumstances and have also been considered as a possible security threat [13]. In fact, descriptions of network adversaries read like a cookbook for stealthy low-rate denial-of-service (DoS) attacks inducing arbitrary long queues in a target network, which in turn cause high delays and loss.

After a period of very high activity in AQT research in the late 1990s and early 2000s, recently, there has been only little work on analytical aspects of adversarial queueing theory. This might be due to the fact that fundamental results about the stability of network systems have been obtained. Unfortunately, these fundamental results are mainly concerned with the notion of *universal stability*[1], which has only been shown for networks that are restricted in topology or employ unrealistic scheduling policies (see Section 2 for details). In contrast, the actual threat potential of adversarial queueing effects in systems that are not universally stable has not been systematically studied. This paper is a first attempt to complement existing AQT research by studying adversarial effects for more realistic network configurations.

Although topological considerations suggest that adversarial instability may be possible in realistic network topologies [20], it is not immediately obvious whether instability effects really pose a practical threat. The most striking theoretical and "analytically convenient" assumptions of AQT are infinite buffers and a synchronized network model. The central question of AQT about the existence of upper bounds on queue lengths ($\rightarrow$ stability) only poses itself under the assumption of unbounded queues. However, device buffers are always finite in length and thus not subject to infinite growth. This paper approaches AQT from a completely different angle than existing literature and investigates the *quantitative* effects of adversarial queueing in finite buffer networks ($\rightarrow$ loss) with some asynchrony due to random effects in nodal processing and link delays. We consider timing variations for adversarial injections and various degrees of randomization for the network model.

This leads to two interesting questions. The first is whether classical AQT examples result in excessive packet loss. We show that this is not the case, but instead, finite buffers

[1] Universal stability refers to stable behavior of a scheduling policy, or a topology – under any adversary (cf. [3, 7])

SIGMETRICS'14, June 16–20, 2014, Austin, Texas, USA.

http://dx.doi.org/10.1145/2591971.2592006.

stabilize the system enough to limit the overall loss rate effectively. This could lead to the conjecture that AQT is a somewhat theoretical artifact with little practical relevance. Thus, the second question is about the existence of more efficient adversaries. This is positively confirmed by presenting new adversarial configurations that do suffer from considerably higher loss rates than previous scenarios and prove robust against randomization effects. In particular, these adversarial scenarios introduce two novel approaches: *interlocking* of adversaries and *reactive* adversaries. These approaches are used to construct *loss-efficient* adversaries. Here, efficiency is seen from the perspective of the network adversary and denotes its ability to effectuate a high loss rate at a low injection rate of adversarial traffic. We quantify the loss behavior of both, the classical and the newly constructed adversaries, using analytical methods and simulation. The new adversarial constructions are more complex than existing AQT scenarios and are not amenable to a straightforward analytical evaluation of their worst-case loss. Therefore, *deterministic simulation* is used to observe their behavior. In addition, simulation is also used to assess the impact of reduced event synchronization by randomizing certain parts of the network model. To this end, we have extended an open-source simulation framework with a (randomized) network model and corresponding adversary implementations which we release as an open-source contribution[2].

Following our goal of assessing AQT in a practical context, this work is focused on FIFO scheduling. While other scheduling schemes have been found to be universally stable in the AQT literature [3], FIFO is essentially the only relevant scheduling discipline in practice among those typically studied.

The rest of the paper is organized as follows. Section 2 surveys the existing literature. Section 3 briefly recapitulates AQT essentials, while Section 4 gives a modified model of adversarial effects in finite buffer settings. Section 5 presents the loss rate analysis for classical AQT scenarios and Section 6 introduces the novel scenarios. In Section 7 we present simulation results on the robustness against randomization and the paper is wrapped up with a brief conclusion in Section 8.

2. RELATED WORK

Previous work investigating the adversarial queueing model is focused on the conditions for network stability. A network is called stable if there exists an upper bound on the number of packets in this network for any arbitrary long time interval [3]. The time evolution is studied as a distributed game between the network system (topology and scheduling) and a hypothetical adversary. The network system is shown to be either stable or unstable, i.e., possessing an upper bound on packet delay or not.

Previous results have explored the boundaries of stability conditions, in particular with respect to the scheduling policy [3], the longest path in the network [17], graph minors [2, 3, 12, 20], and injection rate [3, 5, 10, 14, 17]. Most notably, Bhattacharjee and Goel [5] prove that networks with FIFO scheduling can be unstable at arbitrary small injection rates. Furthermore, the family of stable topologies with FIFO scheduling is restricted to a super-class of directed acyclic graphs, the so-called *decorated cycles* as shown by Weinard [20]. Yet, the class of decorated cycles does not encompass many realistic network topologies, thus indicating a potential threat for the stability of real-world networks.

Simulation has been employed to study adversarial scenarios before, most notably in [9], [11], and recently in [4], which all study unbounded queues. In contrast, our work is focused on quantifying loss in a finite buffer setting. Chroni *et al.* [9] use a simulation model to determine the stability of compositions of different schedulers in a network. They do not assess FIFO and do not report quantitative results, but use simulation to determine whether a network appears to be stable or not. Cespedes *et al.* [11] do not focus on particular adversarial scenarios, but instead choose packet destinations distributed uniformly over the network and investigate the effects of clock asynchrony on timing-based scheduling algorithms.

A related issue to the randomized AQT model are randomized scheduling decisions and their effects on network stability as considered by Lorion and Weinard [16]. Similarly, the asynchronous timers of Cespedes *et al.* [11] only affect algorithms which base scheduling decisions on timers. This is different from the randomization assessed in our work as we focus on deterministic FIFO scheduling and randomness in the *timing of packet injections* and *variability in link delays*.

A finite buffer AQT model is also studied by Aiello *et al.* [1] using a competitive analysis, but they do not consider a network system's total number of dropped packets as a performance metric, but instead use throughput as a metric since no scheduling policy admits a competitive ratio for the number of dropped packets. In the present paper, we propose to use network loss (i.e. the ratio of dropped to successfully delivered packets) as a performance metric and show that previously proposed adversaries appear harmless under this metric.

To the best of our knowledge, no previous work has reported results on network loss in a finite-buffer AQT setting and no results are known under the condition of a network model that is not assumed to proceed in synchronous uniform steps. We appear to be the first to quantify these effects and to propose dedicated network adversaries tailored to this setting.

We remark that a possible perspective on adversarial queueing effects is their interpretation as low-rate DoS attacks [13]. These attacks often rely on synchronizing flows so that packets arrive as correlated bursts as first reported by Kuzmanovic and Knightly [15]. In fact, the temporal evolution of these burst arrivals (shown in Figure 1) appears similar to what can be observed in AQT simulations (shown in Figure 2): an attacker induces periodic waves of packet arrivals which quickly fill up queues. Although the effect is related, our work is concerned with more fundamental aspects of this problem and the analysis is carried out starting from the original theoretical AQT settings where, e.g., no congestion control algorithms exist.

3. BACKGROUND

The network in an adversarial scenario is represented as a directed graph $\mathcal{G} = (V, E)$ with network nodes as vertices V and links as edges E. Time is divided into discrete time steps and each node processes one unit packet per one unit time step. Each edge in the graph is associated with a queue where packets wait if they arrive at an already busy edge

[2]All source code is available on our project page `http://disco.cs.uni-kl.de/content/Aqtmodel`

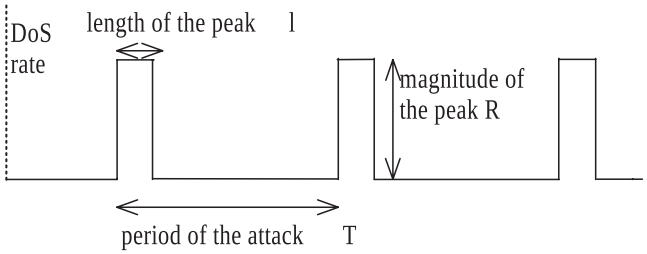

Figure 1: Example of a low-rate DoS (due to Kuzmanovic *et al.* [15], Copyright 2003 ACM).

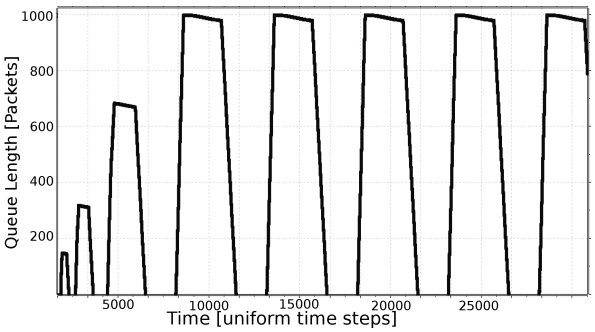

Figure 2: Periodic Bursts in AQT Simulation Trace.

(output queue buffering). If more than one packet is available to be processed at some edge, then the next packet is chosen according to FIFO scheduling. Each time step consists of the following sub-steps: 1) injection of new requests by the adversary, 2) edge traversals of packets already in the system, 3) absorption of packets, if they reach their destination.

The adversary can inject packets with chosen loop-free paths into the network, i.e., each path contains each edge at most once. In order to not trivially overload the system, the adversary's requests are subject to a local load condition. Previous work has introduced different models to express this condition – see [8] for a comparison. We adopt an adversary definition that is less bursty than the original definition [7] but still sufficient to fulfill the requirements of the first result about FIFO instability [3] and all subsequently published instability examples. The *injection rate* $0 < r < 1$ is bounded for every path at any time: for any time interval $[s, t)$, the adversary's injections requiring any particular edge in the graph must not exceed $r(t - s)$ packets. Additionally to this, adversaries are allowed to require an initial configuration of packets in queues in the network. This is allowed only once at the start of time and can be considered as a warm-start assumption. The number of packets required for a warm start is considered to be very small in particular when being compared to the lengths of queues over time. Previous work has introduced transformations of adversaries, so that this requirement can always be satisfied [3, 6]. Additionally, simulation results on the minimal size of initial configurations required for classical adversaries support the claim that already a small number of packets suffices [4].

An adversarial scenario is defined as a specific combination of a topology $\mathcal{G}$ and an adversarial strategy $\mathcal{A}$ that describes a traffic flow pattern. AQT studies the time evolution of the tuple $(\mathcal{G}, \mathcal{A})$ as specific worst-case flow patterns $\mathcal{A}$ repeatedly emerge in $\mathcal{G}$. As such, AQT studies pessimistic effects over

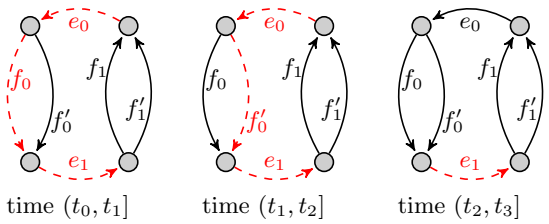

Figure 3: A network called the *Baseball graph* with the corresponding adversarial strategy indicated by dashed edges and in tabular form below.

ADVERSARY 1. *Baseball Adversary* ($\mathcal{A}_1$)

time interval	*set*	*at*	*with path*	*size*
at t_0	S_n	e_0	(e_0)	s_n
$(t_0, t_1 = t_0 + s_n]$	X_n	e_0	(e_0, f_0, e_1)	$r\, s_n$
$(t_1, t_2 = t_1 + r s_n]$	Y_n	e_0	(e_0, f_0', e_1)	$r^2\, s_n$
$(t_1, t_1 + \frac{s_n r}{1+r}]$	C_n	f_0	(f_0)	$\frac{s_n r^2}{1+r}$
$(t_2, t_3 = t_2 + r^2 s_n]$	Z_n	e_1	(e_1)	$r^3\, s_n$

The induction hypothesis (at t_0) indicates an initial set.

long time scales. The evolution of the scenario is given in the form of an induction over time intervals $n \in \mathbb{N}$ which are called *phases*. The induction hypothesis is based on the state of the network at the beginning of a phase n when some queues already hold packets. These queues are called *initial sets* and for the induction base they are satisfied by the initial configuration. In the induction step, traffic flows are generated by injections of the adversary such that for the next phase $n + 1$ the hypothesis of initial sets is satisfied again and thus the same injection pattern can be performed again.

3.1 Example

An important example is the Baseball topology which is due to [3] and called Baseball graph (BB). The topology and the adversary are illustrated in Figure 3 and Adversary 1, respectively. As induction hypothesis in phase n, assume that s_n packets are queued in e_0 at time t_0; this set of packets forms the initial set in this example. One phase corresponds to injections in intervals $(t_0, t_3]$ where we omit the reference to n, the current phase, in order not to clutter notation. At the end of phase n (at time t_3) and for $r \geq 0.85$, s_{n+1} packets with $s_{n+1} > s_n$ are queued in e_1 which can be used to execute the same pattern in the symmetric part of the network in phase $n + 1$.

The adversary's goal is for the two sets X_n and Y_n to arrive simultaneously at e_1. This is achieved by X_n's advance being blocked first by S_n and then by C_n packets while Y_n is injected. For this reason, the size of consecutive injections depends on the size of S_n. Because X_n, Y_n, and Z_n form the eventual queue in e_1 these are called *bottleneck* packets and we call C_n *confinement* packets, as packets of this type confine other approaching packet flows. Over time, the repeated adversarial traffic pattern causes packets to aggregate in the network and the bottleneck buffers e_0 and e_1 face periodic, increasing bursts.

4. FINITE BUFFERS

We now depart from existing adversarial queueing models by assuming that queues are of finite length; for the ease of presentation, we assume a uniform buffer size of b packets throughout the network.

4.1 The Notion of Steady State

For infinite buffers, unstable systems experience ever growing delays; yet, in a model with finite buffers (and thus for any real-world network) this clearly does not hold. At some point, the packet burst s_{m+1} of some phase m does not fit into the queue buffer b any more, and the excess of $s_{m+1}-b$ leaves the system as loss. If the traffic flows continue unchanged, the system enters a *steady state* where any set $s_n, n > m$ is always bounded by the finite buffer size b.

An intuitive example for steady-state behavior can be observed in Figure 2. By means of a queue length plot of a bottleneck queue, the figure shows the transition of the Baseball scenario from a transient state of growing periodic bursts to a steady state of fixed-sized periodic bursts. We capture this intuition by the following definition.

DEFINITION 1 (STEADY STATE). *A network system representing an adversarial scenario $(\mathcal{G}, \mathcal{A})$ with uniform buffer size b is in steady state when all initial sets are of size b from some phase onward.*

For the network adversaries defined in [3, 5, 10, 14, 20] being unstable under the infinite buffer assumption implies to reach such a steady-state in a finite queue setting. For the given examples, instability arises for configurations with r greater than some lower bound r_0 (see, e.g., Section 3.1). In turn, the corresponding finite-queue system with the same configuration reaches steady state.

The fundamental adversarial mechanism underlying the known adversarial queueing scenarios is an incremental build-up of packet bursts relying on the congestion caused by the initial sets in each phase. Because of this incremental build-up, the worst-case behavior of such scenarios can be obtained by setting the cardinality of all initial sets to their maximal value, i.e., to b. For the scenarios analyzed in Sections 5 and 6, the long-term behavior of the system converges asymptotically to this worst case.

4.2 Loss as the New Threat Measure

To assess the severity of adversarial effects quantitatively, we propose to measure loss, which is a common indicator of congestion in a network. Instead of computing the loss for individual network segments though, the number of lost packets is compared to the total number of packets injected by the adversary:

$$(network)\ loss = \frac{\#\ packets\ lost}{\#\ packets\ injected}$$

On a network-system level this ratio can be considered as the loss rate over time or as the probability to lose any injected packet independent from its path. Other threat measures could easily exaggerate the severity of adversarial effects. For example the maximum burst size or the length of a single congestion event can grow arbitrarily large, if no topological parameters are considered. However, in such cases the total relative loss usually remains small, because the adversary has to inject a large number of packets in order to create these bursts; while, many of these packets are unaffected by congestion (see Section 5.2).

Besides indicating inherent global risk, the total relative loss also helps to quantitatively estimate the feasibility of adversarial network attacks. Attackers may inject packets into the network to artificially create adversarial effects, which would cause denial of service in the target network. It is desirable to obtain a lower bound on the effort needed to launch attacks of this type, assuming that an attacker attempts to use available resources as efficiently and stealthily as possible.

For the loss analysis, assume a system $(\mathcal{G}, \mathcal{A})$ that is unstable for $r \geq r_0$ (assuming an infinite amount of buffer). The analysis is limited to injection rates greater than r_0, since the system will endure loss only in this parameter range. As the adversarial queueing scenarios analyzed in the next section reach their respective steady states for $r \geq r_0$ (proofs omitted), their analysis proceeds by setting the cardinality of all initial sets to b.

5. LOSS IN CLASSICAL AQT EXAMPLES

This section presents an analytical study of the loss for two well-known examples of adversarial scenarios given in [3] and [5]. To estimate network loss for a system $(\mathcal{G}, \mathcal{A})$ the number of packets lost and the number of packets injected need to be determined. Loss depends on the injection scheme of the adversary $\mathcal{A}$, the cardinality of initial sets, and on $\mathcal{G}$. The total number of injections can be derived from the description of $\mathcal{A}$.

5.1 Loss in the Baseball Graph

The first example is the system $(\mathcal{G}_{BB}, \mathcal{A}_1)$ shown in Figure 3. We assume the system to be in some phase n and that the initial set has the cardinality s_n $(= b)$. $\mathcal{A}_1$ induces a queue of $s_{n+1} = r^3 s_n + r^2 s_n/(r+1)$ packets in the bottleneck at the end of the phase, as stated in [3]. To get the total number of injections, it is necessary to find the exact cardinality of the confinement set C_n (see Section 3.1), because this cardinality is not stated in the original description of $\mathcal{A}_1$ [3]. For efficiency reasons, the cardinality of C_n must be such that throughout $(t_1, t_2]$ packets compete with X to traverse f_0, but after t_2 no packets of C_n may remain in the queue of f_0. In other words, $|C_n| = \frac{s_n r^2}{1+r}$. As all cardinalities stated here ignore static offsets, this leads to an upper bound as follows:

$$\begin{aligned} loss \leq & \frac{s_{n+1} - s_n}{|X_n \cup Y_n \cup Z_n \cup C_n|} \\ = & \frac{s_n(r^3 + \frac{r^2}{1+r} - 1)}{s_n(r + r^2 + r^3 + \frac{r^2}{1+r})} \\ = & \frac{r^4 + r^3 + r^2 - r - 1}{r^4 + 2r^3 + 3r^2 + r} \end{aligned}$$

It is interesting to note that the loss scales independently of the buffer size b. This is due to the length of individual phases being directly dependent on the initial set's cardinality s_n and our worst-case assumption of steady-state loss, i.e., that $s_n = b$. For $r < 0.85$ the system $(\mathcal{G}_{BB}, \mathcal{A}_1)$ does not necessarily reach a steady state and insignificant loss occurs (see also simulation results in Section 6.4). With growing injection rate r, the loss increases approximately linear to $\frac{1}{7} \approx 14.23\%$ for $r = 1.0$.

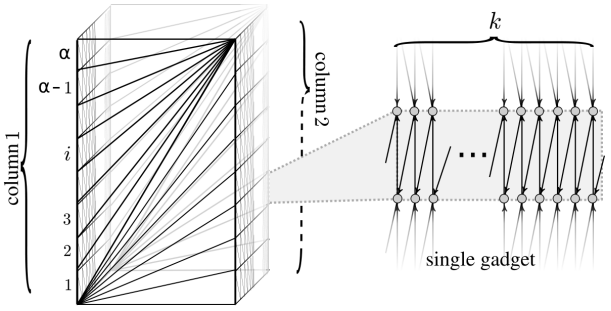

Figure 4: FIFO-unstable topology at arbitrary small injection rates.

Another observation that can be drawn from this example is that the small in-degree of e_0 and e_1 inherently limits the arrival rate and thus the numerator of the loss equations. If there were another incoming edge, the numerator could comprise additional terms (besides r^3 and $\frac{r^2}{1+r}$). However, greater in-degrees amplify the role of another factor that can limit packet loss as shown in the next example.

5.2 Loss at Extremely Small Injection Rates

Following the initial baseball graph instability example, further work on FIFO instability has sought to decrease the bound at which instability occurs (see also the comprehensive survey in [8]). The ultimate goal has been reached in [5] where Bhattacharjee and Goel present a system which is unstable at arbitrary small injection rates, i.e., $r = 0 + \epsilon$ for infinitesimal $\epsilon > 0$. The corresponding graph is constructed in a parameterized way from basic building blocks and can therefore compensate low injection rates by increasing graph size.

We use the following notation, which is slightly adapted from the original one used by Bhattacharjee and Goel. The network consists of two columns of α concatenated gadgets of width k (Figure 4); label the gadgets $i \in 1, \ldots, \alpha$. Additionally each gadget i in column one is connected to the first gadget of column two by an additional chain of gadgets, a so-called connector; the same holds vice versa. The parameters α and k are actually functions of r, for details we refer to [5].

In each phase n an initial set S_n travels through one column. Every time it traverses a gadget i, its advance is blocked by k single-edge injections. The resulting queue S_n^i at each gadget i is then used to inject bottleneck packets into the connectors towards the symmetric column's first gadget.

As a rough bound on the number of lost packets we assume that all injected packets arrive at the bottleneck at the same time (which is not the case and largely overrates the adversary's efficiency). Furthermore, we rely on the bound on $s_n^i = |S_n^i|$ from Lemma 5.1 in [5]: $s_n/2 < s_n^i < s_n$, where $s_n = |S_n|$. In each sub-phase i a set X_n^i of cardinality $x_n^i \leq s_n^i\, r/k$ packets is injected into each path towards the bottleneck. As each gadget has k paths, and there are α such sub-phases, it holds that:

$$s_{n+1} \leq \sum k\, x_n^i \leq \sum_{n=1}^{\alpha} s_n^i\, r < \alpha\, s_n\, r. \qquad (1)$$

For a rough upper bound on the loss, it is sufficient to bound additional injections used to hold back each X_n^i's packets until all of them are injected. This is achieved by k additional single-edge injections into each gadget chain of each X_n^i; after it is is injected, it needs to be held back for another $\alpha - i$ sub-phases. Now, each X_n^i needs at least $s_n^i > \frac{s_n}{2}$ time steps for injection, and hence one sub-phase produces at least $(r s_n^i)\, r\, k$ confinement packets. Thus, the total number of packets injected is at least:

$$\sum_{i=1}^{\alpha} (r s_n^i)\, r\, k\, (\alpha - i) > \sum_{i=1}^{\alpha-1} r^2 \frac{s_n}{2}\, k\, i = \alpha\, s_n\, r \left(\frac{1}{4} r k (\alpha - 1) \right)$$

Finally, we use the upper bound from (1) $\alpha s_n r$ on the number of bottleneck packets and obtain $loss < \frac{1}{\frac{1}{4} r k (\alpha - 1)}$.

For the limit $r \to 1$ and the best choice of parameters, the loss of this system stays below 0.3%. The reason why this rough estimation gives such an expressive bound is that the number of confinement packets – injected by the adversary to delay packet flows in order for them to arrive at the bottleneck link at the same time – is excessively large. We conclude that loss can be safely ignored for this system.

5.3 Conclusion for Loss in Classical Examples

We have studied several additional adversarial scenarios from the literature and found that the baseball graph actually has the worst loss characteristics, despite its simplicity. Accordingly, we argue that classical AQT examples do not show excessive loss. However, these examples were not targeted at a finite buffer model and cannot necessarily be expected to deliver worst-case results for loss. Therefore, we reformulate the goal of the adversary to *maximize loss*, that is, be as *effective* (induce instability) and *efficient* (use the least number of packets) in terms of injections as possible. We propose two appropriate adversarial scenarios that are described in the next section.

6. NEW ADVERSARIAL SCENARIOS

The analysis presented in the previous section shows that for small topologies like the baseball graph, small in-degrees limit the arrival rate at bottleneck buffers and as such the loss. For larger graphs, on the other hand, the number of confinement packets (defined in Section 3.1) required by classic adversaries is so high that it always keeps the global loss rate low.

We propose a new adversarial concept called *interlocking* which alleviates the need for large numbers of confinement packets: several adversarial components are interweaved, such that the bottleneck packets of one component act as confinement packets for the others and vice versa. This idea turns previously unused confinement packets into bottleneck packets enabling highly loss-efficient adversaries. For readability we introduce the concept via two examples (as is typical for AQT results, e.g., [3, 5, 10, 14, 17]) but point out that many adversarial components can be reused within this concept.

6.1 Interlocked Baseballs

As the first example, the interlocking concept is applied to two baseball networks denoted as *Outer Adversary* and *Inner Adversary* (see Figure 5). These adversaries are synchronized, so that the injections of the Outer Adversary are held back by packets of the Inner Adversary, and vice versa. In such a scenario, significantly fewer dedicated confinement packets are needed, which results in higher network loss. To see the

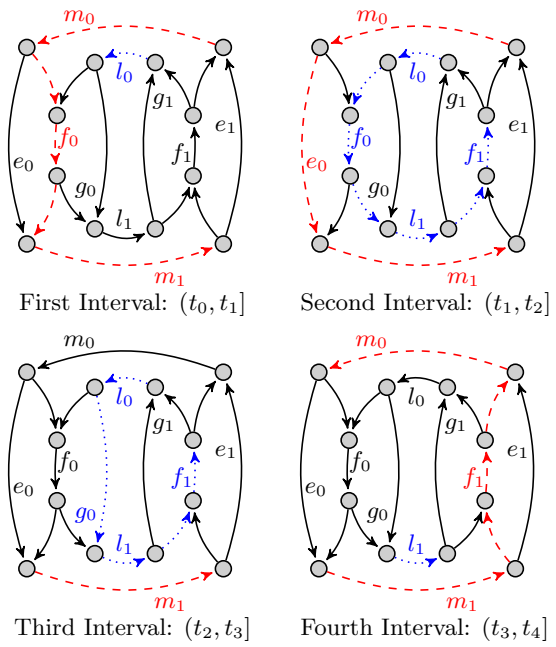

Figure 5: The Interlocked Baseballs (IB) scenario comprises two baseball graphs. Corresponding injection patterns are shifted by one interval and are indicated by dashed and dotted edges. The phase overlap due to each sub-adversary being completed in three intervals can be observed in $(t_0, t_1]$ and in $(t_3, t_4]$. Only the subset of edges needed for uniqueness of path description is labeled.

analogy of the Outer Adversary with the original Baseball graph, compare m_0, e_0, f_0, m_1, e_1, and f_1 in Figure 5 with e_0, f_0, f_0', e_1, f_1', and f_1 in Figure 3. A respective analogy exists between the Inner Adversary and the original Baseball graph. This mapping is illustrated in Figure 6.

Both Baseball-type components in this adversarial scenario execute in lockstep during alternating phases. Therefore, the adversary is described with two induction hypotheses: an initial set O_n in m_0 for the Outer Adversary and I_n in l_0 for the Inner Adversary. In terms of the overall scenario, numbered phases are divided into even and odd phases; each phase n is subdivided into intervals of length o_n, where $o_n = |O_n|$. We describe roughly what happens to the injections, which are detailed in Adversary 2. As before, we omit the description for the symmetric part of the network because the adversary uses the same strategy for odd phases $n+1, n+3, n+5, \ldots$.

In the first interval, the initial set O_n starts traversing m_0 at $t_0 + 1$ and by doing so, it blocks the path of O_n^1 until $t_1 + 1$. Next, O_n^1 competes with X_n to traverse the edge f_0. This competition is the basic mechanism connecting the two interlocked adversaries with each other. At the same time, O_n^2 is injected and its packets are queued up behind O_n^1. After $r\,o_n + 1$ time steps, O_n^1 will have left the queue of m_0 and O_n^2 starts traversing m_0. Simultaneously, I_n^1 is injected into l_0. Its long path (and the path of I_n^2) is necessary to form X_{n+1} for the next phase. The packets in I_n^1 are buffered behind the initial set I_n at l_0 and start traversing l_0 at time step $t_1 + i_n$. In the third interval, O_n^1 and O_n^2 arrive

ADVERSARY 2. *Interlocked Baseballs Adversary*

time interval	*set*	*at*	*with path*	*size*
at t_0 (ind. hypothesis)	O_n	m_0	(m_0)	o_n
$(t_0, t_1 = t_0 + o_n]$	O_n^1	m_0	(m_0, f_0, m_1)	$o_n\, r$
at t_1 (ind. hypothesis)	I_n	l_0	(l_0)	i_n
at t_1 (ind. hypothesis)	X_n	l_0	(l_0, f_0)	x_n
$(t_1, t_2 = t_1 + o_n]$	O_n^2	m_0	(m_0, e_0, m_1)	$r\, o_n$
(t_1, t_2)	I_n^1	l_0	(l_0, f_0, l_1, f_1)	$r\, o_n$
$(t_2, t_3 = t_2 + o_n]$	O_n^3	m_1	(m_1)	$r\, o_n$
(t_2, t_3)	I_n^2	m_0	(l_0, g_0, l_1, f_1)	$r\, o_n$
$(t_3, t_4 = t_3 + o_n]$	I_n^3	(l_1)	(l_1)	$r\, o_n$

The induction hypothesis (ind. hypothesis) indicates an initial set. Note that X_n is a subset of I_n – both are initial sets that reside in the same queue, but differ in that the packets in I_n require to traverse the path (l_0, f_0).

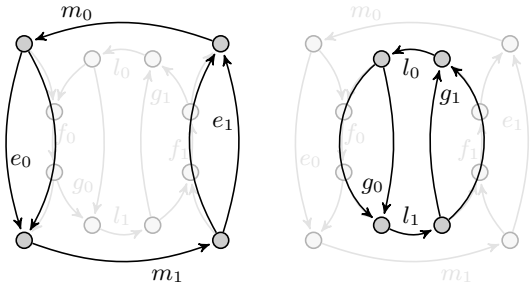

Figure 6: Mapping the Interlocked Baseballs (IB) scenario to its two baseball graphs. Left: the Outer Adversary, right: the Inner Adversary.

at m_1, while O_n^3 gets directly injected into the same edge. These arrivals fill the queue of m_1 rapidly and form O_{n+1}, which becomes the initial set of the next phase. Still in the same time interval, another set I_n^2 is injected by the inner adversary while I_n^1 is still queued behind O_n^1. Finally, in the fourth time interval, I_n^1 and I_n^2 arrive at l_1 together with an additional injection I_n^3. Note that the phases n and $n+1$ overlap. While the arrivals at l_0 form I_{n+1} in (t_3, t_4), the next phase of the outer adversary has already started. In other words, $(t_0, t_1]$ of phase $n+1$ is already active.

6.2 Loss in the Interlocked Baseballs Scenario

In the analysis, we ignore constant offsets towards lower bounds. For example, Set O_n^1 competes to traverse f_0 at $t_1 + 3$ with Set X_n but we count as if it arrived at t_1. Also, some expressions of set cardinalities may become negative, but we omit the $[\ldots]^+$ operator, because set sizes cannot become negative.

LEMMA 1.

$$o_{n+1} = \frac{(4\,o_n\,r - 2\,o_n)\;x_n + 3\,o_n\,i_n\,r - 2\,o_n\,i_n}{x_n + i_n} \tag{2}$$

$$i_{n+1} = o_n(4\,r - 3) + x_n \tag{3}$$

$$x_{n+1} \geq o_n\;(4\,r - 3) + x_n - \frac{2\,o_n\,r}{r+2} \tag{4}$$

PROOF. In $(t_1, t_2]$, X_n competes with O_n^1 to traverse f_0. X_n arrives on average at rate x_n/i_n and O_n^1-packets arrive

at rate 1. Thus, X_n traverses f_0 at rate $\frac{x_n/i_n}{x_n/i_n+1}$, and O_n^1 at rate $\frac{1}{x_n/i_n+1}$. Overall, $\frac{o_n\,(x_n/i_n)}{x_n/i_n+1}$ X_n-packets, and $\frac{o_n}{x_n/i_n+1}$ O_n^1-packets traverse f_0 as this interval is o_n time steps long.

At t_2, $r\,o_n + x_n - o_n$ packets from O_n^1 and X_n are left in queue f_0 to block the advancing packets of I_n^1. Thus, until t_3, another $o_n - (r\,o_n + x_n - o_n) = 2o_n - r\,o_n - x_n$ packets will traverse f_0 and therefore, I_n^1 will decrease by that number of packets.

While O_n^2 gets injected into m_0 in $(t_1, t_2]$, the queue of m_0 holds only $r\,o_n$ packets of O_n^1. Therefore, $o_n(1-r)$ packets from O_n^2 traverse m_0 before the set is fully injected. These packets arrive at m_1 with rate 1. Concurrently, packets from O_n^1 arrive at m_1 with rate $\frac{1}{x_n/i_n+1}$, while traversals at m_1 occur at rate 1, so that in $(t_2 + ro_n, t_3]$, these arrivals will form an additional set E_n at m_1. This set grows at rate $\frac{1}{x_n/i_n+1}$.

Similarly to O_n^2, I_n^2 diminishes by $i_n(1-r)$ packets at l_0 in $(t_2, t_3]$. Also, by the same mechanism as E_n, a set F_n builds up in $(t_3 + ro_n, t_4]$ in the queue of l_1, as early traversals of the sets I_n^1 and I_n^2 arrive at edge l_1 with an arrival rate of 2 while traversals occur at rate 1.

At t_4 the queue of m_1 consists of the remaining packets of new injections (denoted by rem), the additional set E_n and traversals in $t_4 - t_3 = o_n$:

$$\begin{aligned} o_{n+1} =& rem(O_n^1) + rem(O_n^2) + O_n^3 + E_n - o_n \\ =& r\,o_n - \frac{o_n}{x_n/i_n + 1} + r\,o_n - (1-r)o_n + r\,o_n \\ &+ (1-r)o_n \left(\frac{1}{\frac{x_n}{i_n}+1}\right) - o_n \\ =& \frac{(4\,o_n\,r - 2\,o_n)\;x_n + 3\,o_n\,i_n\,r - 2\,o_n\,i_n}{x_n + i_n} \end{aligned}$$

Analogously, the queue length of l_1 at time t_5 can be derived from the remaining arrivals and traversals:

$$\begin{aligned} i_{n+1} =& rem(I_n^1) + rem(I_n^2) + I_n^3 + F_n - o_n \\ =& r\,o_n - (2o_n - r\,o_n - x_n) + r\,o_n - (1-r)o_n \\ &+ r\,o_n + (1-r)o_n - o_n \\ =& x_n + 4\,o_n\,r - 3\,o_n \end{aligned}$$

Finally, we estimate a lower bound on X_{n+1} – the initial packets at l_0 which include f_0 (resp. at l_1 and f_1). This set is only composed of packets from I_n^1 and I_n^2. Both sets' path includes f_1 whereas I_n^3's path does not. Because of this, only a fraction of the traversals at l_1 diminishes X_{n+1}. This fraction can be upper-bounded by considering the corresponding rates of arrivals.

In $[t_4, t_5]$, I_n^1 and I_n^2 each arrive at rate 1 at l_0 while I_n^3 is injected at rate r. Packets from the former two sets traverse at most with rate $\frac{2}{2+r}$. The set F_n only decreases the number of traversals but does not directly count towards X_{n+1}:

$$\begin{aligned} x_{n+1} \geq& rem(I_n^1) + rem(I_n^2) - \left(\frac{2}{2+r}\right)(o_n - F_n) \\ =& o_n\;(4\,r - 3) + x_n - \frac{2\,o_n\,r}{r+2} \end{aligned}$$

because F_n-packets appear first in the queue l_1. □

The ratio x_n/i_n plays a central role to assess instability and loss, but its recursive nature prevents a straightforward evaluation. The following lemma addresses this issue via an auxiliary function, which captures the variety of dependencies from Eqs. 2-4 with a single fixed-point iterator.

We restrict the analysis to $r \geq 0.9$ for ease of presentation.

LEMMA 2. *Assume $r \in [0.9, 1]$, $i_0 \geq o_0$, and $x_0 \geq i_0/3$. Then $i_n \geq o_n$ and $x_n \geq i_n/3$. Furthermore, it holds that $\lim_{n\to\infty} x_n/i_n = (1191/500)\,r - (361/200)$.*

PROOF. The proof is by induction on n. From Lemma 1, i_n/o_n is monotonic in r. Assume $i_{n-1} \geq o_{n-1}$ and $x_{n-1} \geq i_{n-1}/3$. Then, for $r = 0.9$ the ratio i_n/o_n evaluates to a value greater than 1.009. For $r = 1.0$ the ratio is greater than 16/15. This shows the first claim $i_n \geq o_n$ for step n.

It remains to be shown that $x_n \geq i_n/3$ is satisfied. This is done by showing that for the fixed-point of x_n/i_n the ratio is always greater than 1/3 for $r \in [0.9, 1]$. In particular, the fixed-point equation is monotonic and the claim follows by induction.

Even if x_{n-1} and i_{n-1} are known from previous computation steps, computing the ratio of x_n/i_n is not trivial and approximations are used. Assume $R(k-1)$ is the ratio produced by the fixed-point iterator in iteration $k-1$ and $k \leq n$. Then, for the k-th step, we estimate $R(k) = x_k/i_k$ by using ${x_{k-1}}' = R(k-1)\,o_{k-1}$. Observe that ${x_{k-1}}' \leq x_{k-1}$ because $k \leq n$, $R(k-1)\,i_{k-1} = x_{k-1}$, and because we can prove that $i_n \geq o_n$ for the current induction step n (first paragraph in this proof). This approximation leads to the following algebraic simplification:

$$R(n) \geq \frac{(r+2)\,R(n-1) + 4\,r^2 + 3\,r - 6}{(r+2)\,R(n-1) + 4\,r^2 + 5\,r - 6}$$

Using $R(n-1) \geq \frac{1}{3}$ and then evaluating for $r \in [0.9, 1]$ shows that $x_k \geq i_k/3$, which completes the induction proof.

The injection rate r remains as a parameter of $R(n)$. It can be used to obtain a better lower bound on $R(n)$ by expressing it as a linear lower bound below the fixed-point iteration curve for $r \in [0.9, 1.0]$. This is possible by interpolating at the limit values, 0.9 and 1, due to the fixed-point curve being monotonically increasing in r. The result from this interpolation is the limit given in the second part of the lemma. □

The base case, $i_0 \geq o_0$ can be assumed without loss of generality, because it is a constant offset and i_0 and o_0 are small compared to i_n and o_n. The same holds for $x_0 \geq i_0/3$.

PROPOSITION 1. *Under infinite buffers, the Interlocked Baseballs scenario is unstable for $r > 0.916$.*

PROOF. From Lemma 1 we know an equation for o_{n+1}. We use $i_n \geq o_n$ and the bound on x_n to simplify this equation as follows:

$$\begin{aligned} o_{n+1} =& \frac{(4\,o_n\,r - 2\,o_n)\;x_n + 3\,o_n\,i_n\,r - 2\,o_n\,i_n}{x_n + i_n} \\ \geq& \frac{9528\,r^2 - 8984\,r + 1610}{(2382\,r - 805)}\,o_n \end{aligned}$$

This proves that o_n grows without bound for $r \geq 0.916$. Because $i_n \geq o_n$ from Lemma 2 this also applies to i_{n+1}, so that the number of packets in the Inner Adversary's grows without bound, too. □

The Interlocked Baseball adversary's initial sets will thus reach the buffer size b. We additionally claim that the Interlocked Baseballs scenario reaches steady state. This can be

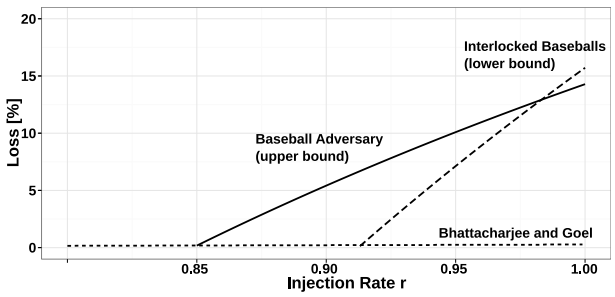

Figure 7: Analytical loss bounds of two classical scenarios in comparison to the Interlocked Baseballs scenario show that there exist new scenarios that can exceed the loss of classical scenarios. Note that the actual difference is more pronounced as we here compare upper to lower bounds (cf. Figure 11).

seen by reconsidering the proof of Lemma 1 under the condition that initial sets of phase n are of cardinality b. Observe that all flows retain their timing (and cardinality) because intermediate buffers (e.g., f_0) do not overrun. Therefore, the expressions from the Lemma hold also in the finite buffer system and can be used in the following proposition.

PROPOSITION 2. *For $r > 0.916$, the lower bound on the loss for the Interlocked Baseballs scenario is*

$$\frac{8243308\,r^2 - 10110340\,r + 2362675}{4764000\,r^2 - 1610000\,r} \tag{5}$$

PROOF. From the description of Adversary 2 we obtain that overall $6\,o_n\,r$ packets are injected in one phase. The loss for one phase can be derived from Lemma 1 and 2 similarly to the proof of Proposition 1:

$$loss = \frac{[\widetilde{o}_{n+1} - b]^+ + [\widetilde{i}_{n+1} - b]^+}{6\,o_n\,r}$$

Here, $\widetilde{o}_{n+1}, \widetilde{i}_{n+1}$ denote the cardinality of the initial sets in the (hypothetical) infinite buffer system. These can be expanded using Lemma 1 and using Lemma 2's fixed point estimate for x_n. Under the assumption of steady state, where $o_n = b$ and $i_n = b$, the closed form of the loss can be derived. □

This analytical result shows that the classical examples can be "outperformed" in terms of loss. As an illustration, Figure 7 shows that the loss bound of the classical Baseball adversary (BB) is eventually surpassed by the lower bound of the Interlock Baseballs adversary (IB). While the difference in this case remains small, note that we are comparing upper to lower bounds where both are not tight (cf. Figures 9 and 10). A more powerful reactive adversary is introduced in the next section and evaluated using simulation in Section 6.4.

6.3 The Reactive Adversary

A refinement of the interlocking concept is possible with three BB-adversaries. Conceptually, interlocking greater numbers of adversaries is a straightforward task. However, to preserve the precise timing of packet flows the construction requires caution, because the different adversaries find different network conditions at the start of each phase.

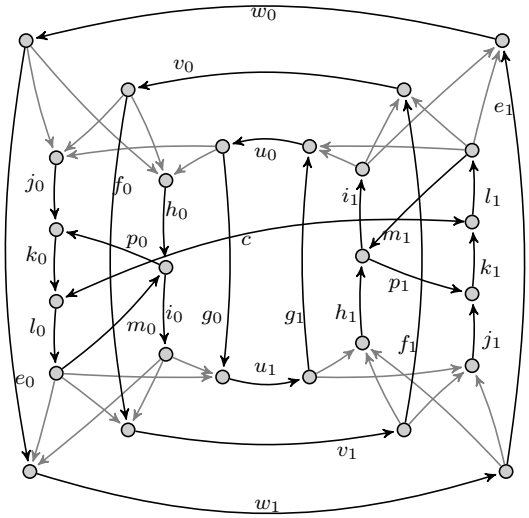

Figure 8: An extension to the Interlocked Baseballs topology with maximum in-degree three which allows for more than three sets to be injected per adversary. The edges m_0, p_0, m_1, p_1, and c are exploited by the Reactive Adversary to create additional confinement by the sets I_n^1 and Z_n.

Previously published adversaries fix an injection pattern which remains the same for each phase; the number of injected packets depends linearly on the cardinality of an initial set. We introduce the notion of a *reactive adversary* which is allowed to adjust more parameters of its strategy based on the network state at the start of a phase. This creates more powerful adversaries, but nevertheless still complies with all previous restrictions on the adversary's power.

We present a simple reactive adversary based on the topology in Figure 8. The adversarial strategy in Adversary 3 utilizes edge c to synchronize packet flows across phases. In particular, if the set Z_n is injected, it affects phase $n+1$ where it creates additional confinement in the path of M_{n+1}^1 and M_{n+1}^2. The essential idea is to conditionally inject Z_n, only if needed, in order to hold the denominator in its loss equation small (cf. the insight from Section 5.2). An efficient condition is to check whether the RA is *not* in steady state, i.e., the threshold in the definition of Adversary 3 is set to $T := b - 1$.

To exploit the graph's maximum in-degree of three, the adversary interlocks three sub-adversaries denoted by *Inner*, *Middle*, and *Outer Adversary* and each adversary injects four (instead of three for the IB) sets (e.g., $O_n^1 - O_n^4$). The first three sets travel along the three paths that connect each two bottleneck links (w_0/w_1, v_0/v_1, u_0/u_1), and the fourth set is a direct injection into the bottleneck link. This extends the duration of each phase n from $4\,o_n$ (for the IB) to $6\,o_n$ time steps (for the RA).

The mutual interdependence of various packet flows makes analytical treatment of this case hard. However, we argue that this construction is very interesting; firstly to study a generalization to the class of interlocked scenarios, and secondly because the network loss of this scenario exceeds all others, as shown by simulations in the next section.

ADVERSARY 3. *Reactive Adversary (RA)*

time interval	*set*	*at*	*with path*	*size*
at t_0 *(ih)*	O_n	w_0	w_0	o_n
$(t_0, t_1 = t_0 + o_n]$	O_n^1	w_0	w_0, j_0, k_0, l_0, w_1	$o_n\, r$
at t_1 *(ih)*	M_n	v_0	v_0	m_n
at t_1 *(ih)*	X_n	v_0	v_0, j_0	x_n
$(t_1, t_2 = t_1 + o_n]$	O_n^2	w_0	w_0, h_0, i_0, w_1	$r\, o_n$
$(t_1, t_2]$	M_n^1	v_0	$v_0, j_0, k_0, l_0, v_1, j_1$	$r\, o_n$
at t_2 *(ih)*	I_n	u_0	u_0	i_n
at t_2 *(ih)*	Y_n	u_0	u_0, h_0, p_0, k_0	y_n
$(t_2, t_3 = t_2 + o_n]$	O_n^3	w_0	w_0, e_0, w_1	$r\, o_n$
$(t_2, t_3]$	M_n^2	v_0	v_0, h_0, i_0, v_1	$r\, o_n$
$(t_2, t_3]$	I_n^1	u_0	$u_0, j_0, k_0, l_0, u_1, h_1, p_1, k_1$	$r\, o_n$
$(t_3, t_4 = t_3 + o_n]$	O_n^4	w_1	w_1	$r\, o_n$
$(t_3, t_4]$	M_n^3	v_0	v_0, f_0, v_1	$r\, o_n$
$(t_3, t_4]$	I_n^2	u_0	u_0, h_0, i_0, u_1	$r\, o_n$
$(t_4, t_5 = t_4 + o_n]$	M_n^4	v_1	v_1	$r\, o_n$
$(t_4, t_5]$	I_n^3	u_0	u_0, g_0, u_1	$r\, o_n$
$(t_4, t_5]$ *(cond.)*	Z_n	j_0	$j_0, k_0, c, l_1, m_1, i_1$	$r\, o_n$
$(t_5, t_6 = t_5 + o_n]$	I_n^4	u_1	u_1	$r\, o_n$

The abbreviation (denoted as 'ih') indicates the inductive assumption of an initial set. The injection denoted by (*cond.*) takes place as long as $o_n = |O_n|$ is less than some threshold T depending on the buffer size b.
Note that X_n is a subset of M_n – both are initial sets that reside in the same queue, but differ in that the packets in M_n require to traverse the path (v_0, j_0). Analogously, $Y_n \subset I_n$ has a remaining path (u_0, h_0, p_0, k_0).

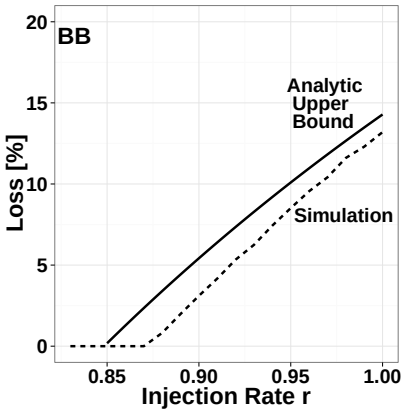

Figure 9: Simulation of Baseball scenario (BB)

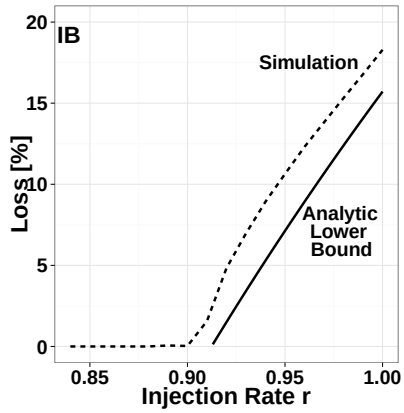

Figure 10: Simulation of Interlocked Baseballs scenario (IB)

6.4 Simulation Results

We implemented the deterministic simulation framework for AQT in OMNeT++ [19]. While previous work in adversarial queueing models relies on pencil and paper evaluation, our new tool allows efficient exploration of the parameter space of various graphs and adversaries. In particular, the simulations are used to obtain quantitative results on the loss in the reactive adversary scenario. Besides analyzing the reactive adversary, the analytical results and the correctness of the simulations are mutually validated by comparing the results in Figures 9 and 10.

For the Baseball scenario, it can be observed that the upper bound is correct (with some overestimation) and the simulation is reasonably close to the upper bound. For the Interlocked Baseball scenario, the lower bound also appears to be correct. In fact, loss has been underestimated by 2-3% using the bound from Proposition 2.

In Figure 11, the loss of several scenarios is compared against each other. In particular, the results for the novel proposals (Interlocked Baseball (IB) and Reactive Adversary (RA)) are shown, but also for the Baseball (BB) from [3] and another scenario termed Gadget Chain [17] (see Figure 12), which also exhibits an interesting behavior. It introduces a very distinct idea from other previous work and utilized the first example of a parametric topology. However, the Gadget Chain scenario oscillates under finite queue conditions, because the timing of adversarial packet flows is interrupted due to lost packets. With a minimal change, however, this adversary can cause some loss, as shown in Figure 11.

The results demonstrate that the new scenarios incur significantly higher loss than the classical AQT scenarios. In particular, the Reactive Adversary rises relatively steep up to 27% loss in the deterministic simulations starting from $r = 0.85$. Also the difference between the Interlocked Baseballs and the Baseball scenarios becomes more pronounced. Further, the behavior of the Gadget Chain scenario [17] is quite different from the others: while loss occurs at low injection rates, the loss curve flattens out and never exceeds 6%. Nevertheless, this gives rise to speculation about the feasibility of loss-maximizing adversaries to threaten a network at low injection rates.

7. EVENT RANDOMIZATION

Adversarial effects reported in the literature are closely coupled to the tight synchronization of packet-level events. As such, considering them as blueprints for problematic configurations or malicious attacks appears somewhat paranoid. Therefore, the impact of randomization effects on loss is

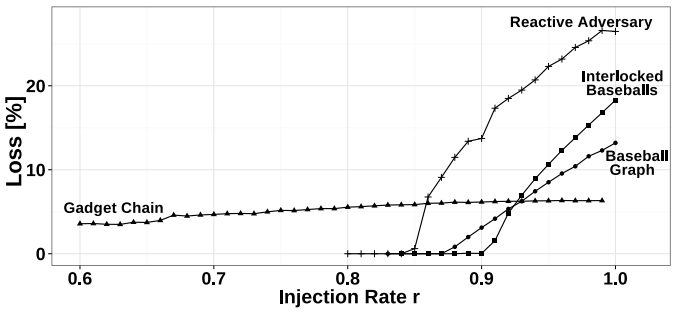

Figure 11: Loss results from deterministic simulation for different scenarios. The new adversarial scenarios yield significantly higher loss than classical adversaries. Note also the low-rate behavior of the Gadget Chain (due to [17]): although flat in shape, the scenario results in about four to six percent loss even for low injection rates.

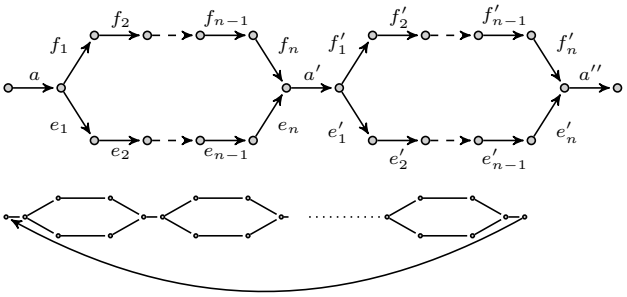

Figure 12: Topology of another adversarial scenario: the Gadget Chain from [17] is constructed by connecting gadgets one after another (top). Each "gadget" $F(i)$ (bottom) composes two lines which join at either end into a common input and output edge.

studied here to better assess the real-world implications of AQT from that perspective as well.

7.1 Imperfect Adversary Synchronization

We model differences in clock time or processing variability that affect the timing of adversarial injections. The time at which the adversary schedules any specific injection is now a random variable. In the classical model, the injection time is deterministic and controlled by the adversary. In the randomized model, realistic differences in clock time or processing variability are expressed as a variance affecting the time at which each injected packet arrives at its target edge. We have chosen a normal distribution because of the distribution's simplicity and symmetry properties. The random variable is truncated at time zero and the impact of varying standard deviations is studied. The empirical CDFs for such randomized injections are shown in Figure 13.

Figure 14 shows that for an example of the classical adversarial scenarios (the Baseball scenario [3]) large values of standard deviation indeed break the inner synchronization of the adversary resulting in smaller network loss. We have found similar behavior for other classical scenarios, but omit the detailed results for brevity. On the other hand, the newly proposed adversaries prove more robust as the effect

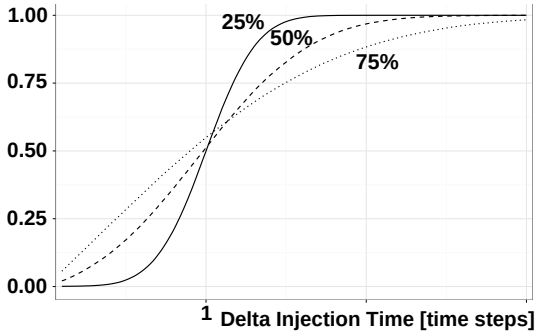

Figure 13: Empirical CDF of the inter-injection time. The inter-injection time follows a truncated normal distribution.

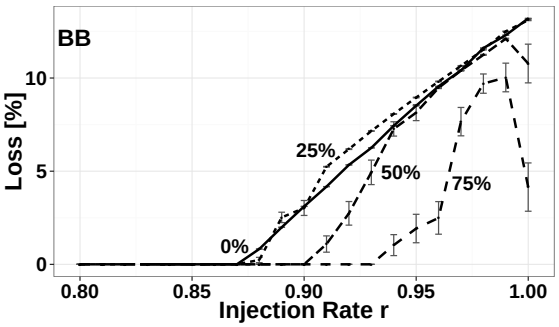

Figure 14: Randomized Injections (BB)

of this form of desynchronization remains small. In fact, the measured loss remains almost unchanged, as depicted in Figures 15 and 16.

7.2 Randomized Network Model

Instead of assuming a deterministic network model with uniform and synchronized network links, each channel's delay (the corresponding server delay for processing a single packet) is now modeled as a Weibull distributed random variable with mean one. Again, the effect of different standard deviations is investigated; we chose up to 300% of the mean as the largest deviation. For comparison, 200% roughly corresponds to the real-world measurements reported in [18]. This delay variability may be due to various effects such as processing variability or volatile cross-flows. Clearly, this only approximates reality since the impact of volatile cross-flows is only represented implicitly because actual cross-flows do not exist in our model. A Weibull model has been shown to be realistic [18] and allows using the standard deviation as parameter in this case (compared to a truncated normal distribution). The empirical CDF for a randomized channel delay is shown in Figure 17.

The results are shown in Figures 18-20. The Baseball scenario (Figure 18) is a typical representative of classical adversaries and its loss diminishes to irrelevant values unless the network utilization closely approaches 1. The Interlocked Baseballs scenario (Figure 19) is similarly affected but appears slightly more robust. Finally, the Reactive Adversary scenario (Figure 20) can yield loss rates of about 10% even for large values of the standard deviation. This adversary's

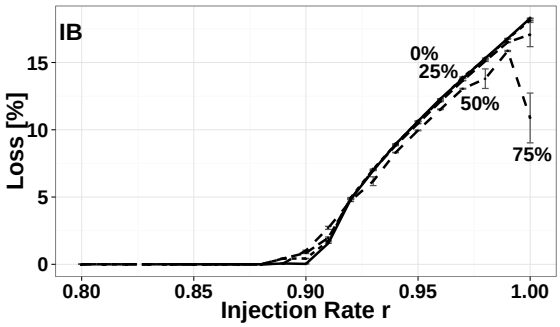

Figure 15: Randomized Injections (IB)

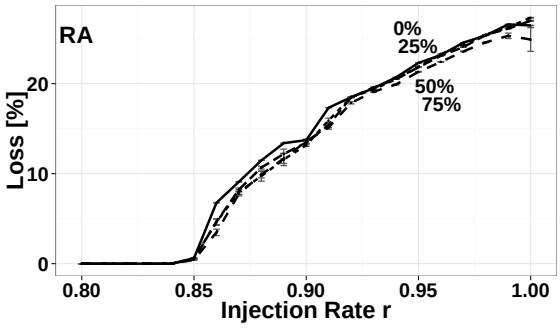

Figure 16: Randomized Injections (RA)

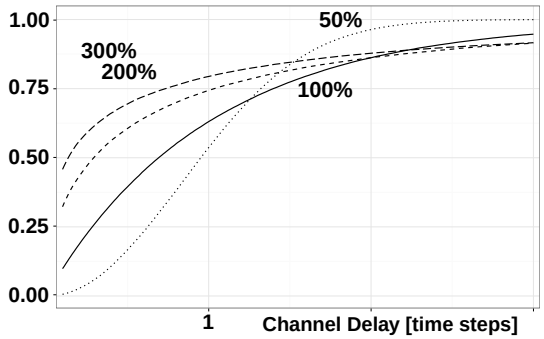

Figure 17: Empirical CDF of the channel delay. The channel delay follows a Weibull distribution.

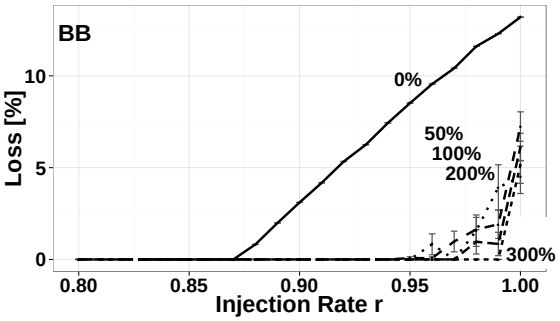

Figure 18: Channel Delay Variation (BB)

self-stabilizing property seems to retain its general behavior and thus also the loss curve's shape.

We conjecture that for both the Baseball and the Interlocked Baseballs scenario the adversarial effect does not play a major role any more under channel delay randomization exhibiting no significant deviation, but the losses are rather due to the typical hyperbolic growth known from classical queueing theory when the average arrival rate approaches the server capacity.

7.3 Discussion of Results

It is interesting to see that all three scenarios appear more robust against changes in the timing of injections than to variability in link delay. We have observed that this is due to packet flows being repeatedly aggregated in queues so that the exact timing of their arrivals in the network has only limited impact. For example, all bottleneck packets are first stored in a queue before they start traversing their first edge. Then, the variability of injection timings are averaged to close to the mean (smoothing effect). Their subsequent travels through the network then depends mostly on the server delay experienced at each edge. This is also why changes in the channel delay have a more pronounced effect on the loss behavior (e.g., compare the results for 50% deviation of Figure 15 to the results in Figure 19).

Both novel scenarios, the Interlocked Baseballs and the Reactive Adversary, prove to be more efficient and more stable than previous examples. Analyzing packet trajectories by simulation shows that indeed the two concepts, interlocking and reaction, are instrumental to achieve this robustness. Firstly, the Interlocked Baseballs adversary is the first scenario not requiring active confinement, i.e., packets which are only injected to slow down another packet flow. Instead, all packet injections are targeted at some bottleneck queue. This strategy improves robustness against randomization due to the queue-aggregation effect described above. Secondly, the new concept of reactive injections, which is used for the set Z_n in the Reactive Adversary, protects the adversary's injections against randomization effects. For previously published adversaries, we have observed that under randomization the initial queue lengths often drop below a certain threshold not allowing further adversarial activity. The Z_n injections prevent these situations by "boosting" the cardinality of initial sets in such situations.

8. CONCLUSION

Adversarial queueing theory has introduced the notion of network instability. However, it is also important to understand and quantify the robustness of adversarial scenarios when assessing them from a practical network engineering perspective. In this paper, we report an investigation of the threat potential of adversarial effects under the realistic assumptions of finite buffers and imperfect synchrony. The hypothesis that classical instability scenarios do not translate into excessive loss under these assumptions is corroborated. However, a novel class of adversarial scenarios is introduced; these exhibit considerably more loss and prove to be more robust under realistic conditions. The results are derived analytically and, untypical for AQT work, through simulations.

Further work is needed to conclusively determine whether there are conditions under which adversarial queueing effects constitute a real-world threat. For example, we conjecture that more efficient loss-maximizing adversarial scenarios can

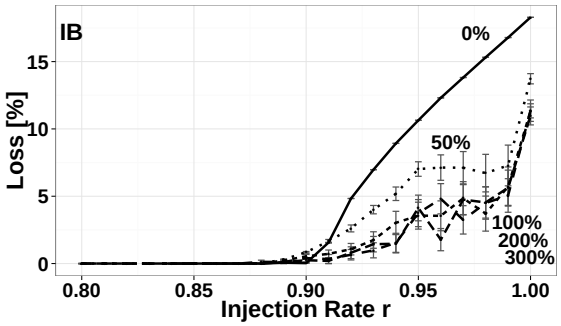

Figure 19: Channel Delay Variation (IB)

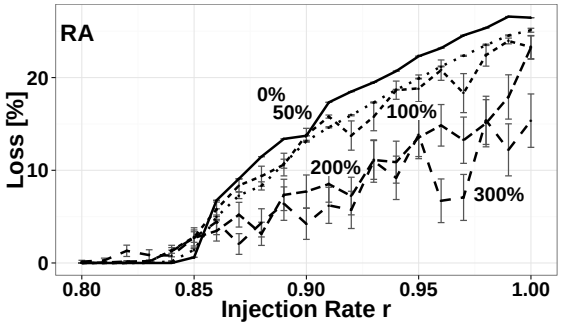

Figure 20: Channel Delay Variation (RA)

be conceived, possibly at lower injection rates. On the other hand, the complexity of such adversarial scenarios probably makes them an unlikely blueprint for actual attacks. As well, the chance of such patterns occurring naturally as a result of misconfiguration or unfortunate circumstances is probably small. This paper is the first to systematically address these fundamental questions and presents first steps towards their answering.

9. ACKNOWLEDGEMENTS

This research was supported by the German Research Foundation (DFG) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

10. REFERENCES

[1] W. Aiello, R. Ostrovsky, E. Kushilevitz, and A. Rosén. Dynamic routing on networks with fixed-size buffers. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 771–780. Society for Industrial and Applied Mathematics, 2003.

[2] C. Alvarez, M. Blesa, and M. Serna. A characterization of universal stability in the adversarial queuing model. *SIAM Journal on Computing*, 34(1):41, 2004.

[3] M. Andrews, B. Awerbuch, A. Fernández, T. Leighton, Z. Liu, and J. Kleinberg. Universal-stability results and performance bounds for greedy contention-resolution protocols. *Journal of the ACM*, 48(1):39–69, Jan. 2001.

[4] D. Berger, M. Karsten, and J. Schmitt. Simulation of adversarial scenarios in OMNeT++: Putting adversarial queueing theory from its head to feet. In *Proceedings of the 6th ICST Conference on Simulation Tools and Techniques*, pages 291–298, 2013.

[5] R. Bhattacharjee and A. Goel. Instability of FIFO at Arbitrarily Low Rates in the Adversarial Queueing Model. *SIAM Journal on Computing*, 34(2):318, 2005.

[6] M. J. Blesa. *Stability in communication networks under adversarial models*. PhD thesis, Universitat Politècnica de Catalunya, 2005.

[7] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queuing theory. *Journal of the ACM*, 48(1):13–38, Jan. 2001.

[8] V. Cholvi and J. Echague. Stability of FIFO networks under adversarial models: State of the art. *Computer Networks*, 51(15):4460–4474, Oct. 2007.

[9] M. Chroni, D. Koukopoulos, and S. D. Nikolopoulos. An experimental study of stability in heterogeneous networks. In *Experimental Algorithms*, volume 4525 of *Lecture Notes in Computer Science*, pages 189–202. Springer-Verlag, 2007.

[10] J. Diaz, D. Koukopoulos, S. Nikoletseas, M. Serna, P. Spirakis, and D. M. Thilikos. Stability and non-stability of the FIFO protocol. In *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 48–52, 2001.

[11] A. Fernandez Anta, J. L. Lopez-Presa, M. A. Lorenzo, P. Manzano, J. Martinez-Romo, A. Mozo, and C. Thraves. Performance of scheduling policies in adversarial networks with non synchronized clocks. *Theory of Computing Systems*, 48(1):1–22, Jan. 2011.

[12] A. Goel. Stability of networks and protocols in the adversarial queueing model for packet routing. *Networks*, 37(4):219–224, July 2001.

[13] D. Koukopoulos. The impact of dynamic adversarial attacks on the stability of heterogeneous multimedia networks. *Computer Communications*, 33(14):1695–1706, Sept. 2010.

[14] D. Koukopoulos, M. Mavronicolas, S. E. Nikoletseas, and P. G. Spirakis. On the stability of compositions of universally stable, greedy contention-resolution protocols. In *Proceedings of the 16th International Conference on Distributed Computing*, pages 88–102, 2002.

[15] A. Kuzmanovic and E. W. Knightly. Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. In *Proceedings of ACM SIGCOMM*, pages 75–86, 2003.

[16] Y. Lorion and M. Weinard. The effects of local randomness in the adversarial queueing model. In *Algorithms - ESA 2008*, volume 5193 of *Lecture Notes in Computer Science*, pages 672 – 683. Springer-Verlag, 2008.

[17] Z. Lotker, B. Patt-Shamir, and A. Rosen. New stability results for adversarial queuing. *SIAM Journal on Computing*, 33(2):286, 2004.

[18] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, and C. Diot. Measurement and analysis of single-hop delay on an IP backbone network. *IEEE Journal on Selected Areas in Communications*, 21(6):908–921, Aug. 2003.

[19] A. Varga. The OMNeT++ discrete event simulation system. In *Proceedings of the 15th European Simulation Multiconference*, 2001.

[20] M. Weinard. Deciding the FIFO stability of networks in polynomial time. *Algorithms and Complexity*, (3):81–92, 2006.