TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN
University of Kaiserslautern

Fachbereich Informatik
Department of Computer Science

Bachelor Thesis

Effects and Factors of Network Instability:

Daniel Berger

d_berger09@cs.uni-kl.de

January 30, 2012

Supervisors: Prof. Dr.-Ing. Jens B. Schmitt (Technische Universität Kaiserslautern)
Prof. Dr.-Ing. Martin Karsten (University of Waterloo)

Abstract

Adversarial queueing models have been well studied over the last fifteen years. While many results characterize network stability, few address the effects and factors determining network instability in real networks. However, several results show that it might be hard to guarantee a network's stability and hence the prediction of underload instability - a network's queues can grow without bound - can pose a threat to networks.

We seek to analyze deterministic network instability effects as predicted by the theoretic adversarial queueing models. This is enabled by the following three contributions (1) a fundamental characterization of instability and its factors, (2) the introduction of a practical class hierarchy for instability incidents, which allows grading of all previously published instability examples, (3) the proposal of a finite adversarial model, where instability effects can be studied and some preliminary results are discussed.

Important findings of this work comprise topology and packet-flow requirements which determine scaling properties of instability for large networks, the effect of instability in a finite adversarial model: the translation of infinite packet-accumulation into buffer-flooding and loss, and that the growth of burst sizes with worst-case adversarial effects stays linear with the buffer size for two major classes (which comprise all previously published instability examples).

Zusammenfassung

Adversarielle Warteschlangen Modelle wurden in den letzten 15 Jahren intensiv untersucht. Viele Ergebnisse beschreiben Stabilitätseigenschaften, nur sehr wenige untersuchen allerdings die Auswirkungen und Einflussgrößen von Netzwerkinstabilität. Verschiedene Ergebnisse zeigen, dass es schwer ist Stabilität für Netzwerke zu garantieren und entsprechend könnte die Vorhersage von Instabilitätsfällen - Warteschlangen im Netzwerk wachsen ins Unendliche - eine Gefahr für Netzwerke darstellen.

Diese Arbeit versucht deterministische Instabilitätseffekte in Netzwerken zu analysieren wie sie in theoretischen adversariellen Warteschlangen Modellen vorhergesagt werden. Dazu leisten wir die folgenden drei Beiträge: (1) eine grundsätzliche Beschreibung von Instabilität mit entsprechenden Einflussgrößen, (2) die Einführung einer Klassenhierarchie um Instabilitätsfälle beschreiben zu können. In diese können alle bisherigen Beispiele für Instabilität eingeordnet werden. (3) Ein Vorschlag um Adversarielle Effekte im endlichen Maßstab untersuchen zu können und die Diskussion einiger vorläufigen Ergebnisse in diesem Modell.

Wichtige Ergebnisse dieser Arbeit sind Vorausetzungen der Topologie und Paketströme, welche die Instabilitätseigenschaften für große Netzwerke festlegen, der Effekt von Instabilität bei der Betrachtung in einem endlichen Modell (im Netzwerk sammeln sich nicht mehr unendlich viele Pakete, stattdessen werden Puffer überschwemmt bis weitere ankommende Anfragen als Verlust auftreten) und dass Bursts für die zwei wichtigsten Klassen (welche alle bisher veröffentlichten Instabilitätsbeispiele umfassen) in der Größe weiterhin linear von der Puffergröße abhängen.

Contents

Abstract iii

Lists ix

1. Introduction 1
1.1. Contribution and Structure of this Thesis 2

2. Background and Related Work 3
2.1. The Adversarial Queueing Model 4
2.2. Positive Stability Results 5
2.3. Applicability of Stability Results 5
2.4. Previous Work on Instability 6

3. Characterization and Classification 9
3.1. Preliminaries 9
3.2. Queue Build-Up 13
3.3. Assumptions and Efficient Injection Schemes 15
3.4. Classification 22

4. Finite Effects 27
4.1. The Finite Adversarial Model 27
4.2. Finite Instability Effects 28
4.3. Bounding Bursts 29
4.4. Bounded Confinement 30
4.5. Bounds on Injection Classes 35

5. Conclusion 41
5.1. Summary 41
5.2. Open Questions and Future Work 42

Bibliography 43

A. Topology Examples for Instability Classes 46

Erklärung 51

Lists

Figures

2.1. The characterization differentiates between network circumferences (the maximal size of a simple circle) and if we focus on realistic ones, circumferences $k \geq 4$, then we obtain that only decorated directed cycles are left as stable topologies.
The figure shows a decorated cycle as discussed in [40, Lemma 7, page 89]. The thick edges are to represent a collection of at least one upward and one downward edge. This figure is courtesy of [40] . . . 6

3.1. Multiple Round Simultaneous Impact System 10
3.2. Path-Length Example 11
3.3. Example topology to show the use of confinement and initial sets. . 12
3.4. A network called the baseball graph, a popular example to show instability in adversarial models 13
3.5. To visualize the possible relations between L_m and L_n 18

4.1. The buffer of E_t for adversarial injections into the baseball graph starting with an initial set of size 10 (Spline interpolation of predicted buffer states over time). This buffer faces periodic bursts which are growing until they reach buffer size (100 packets) and further growth will be translated into loss. Loss is suggested by the light-grey lines above 100 packets. 29
4.2. The confinement path without additional input edge where the traversal of the first edge in the proof of Lemma 9 is shown. Time step $t+1$ is split into two parts. 33
4.3. Extract Confinement Path 39

A.1. Weinard's simple-path unstable graph $\mathcal{A}^+$. An interrupted line indicates a simple path of an arbitrary number of edges. 46
A.2. Example of an efficient topology for an injection scheme $\in \mathcal{C}_1$. Topologies of this class have been used quite often for instability examples. The dashed edges in the middle and the dots on the right should indicate further edges and paths. 47
A.3. Weinard's simple-path unstable graph $\mathcal{B}^+$. An interrupted line indicates a simple path of an arbitrary number of edges. 47

A.4. An example for a topological structure sufficient for an injection schemes $\in \mathcal{C}_2$. Furthermore, this topology is a good example to visualize the bound in Theorem 5. 48
A.5. Extended topology to allow injection schemes which are not considered in any of $\mathcal{C}_1$,$\mathcal{C}_2$ 49

Tables

3.1. A preliminary classification approach of published FIFO instability examples. 23
3.2. Classification of published FIFO instability examples. The last two are constructions with the parameters M and α, k respectively. . . . 23

1. Introduction

The Internet evolves into an integrated part of daily life where information retrieval, processing, and communication are becoming crucial for society. While this has various consequences it most of all emphasizes how important the availability and efficiency of this resource has become. To describe the desired state of a network the term *stability* emerged. A network is considered stable if it is capable to process the requests given to it, be it that they are benign or malicious traffic.

This aspect seems like a central characteristic that today's networks must comply with. However, achieving stability over long time-scales is non-trivial and has been in the scope of research in varying forms ever since telecommunication networks have been created. Although it had long been conjectured that it is sufficient to require the overall request rates to be lower than the actual processing capacity of any individual part in the networking system, this intuition proved wrong.

Since this discovery the goal of many researchers has been a characterization of stability conditions to find scenarios where stability can be ensured. Nevertheless, several results show that it might be hard to enforce stability in today's network infrastructure (Compare for example [40, Lemma 1], [40, Lemma 7], and [14, Contribution])[1]. Thus, networks might need to cope with instability incidents which might be similar to what has been modelled for the so-called baseball graph (Figure 3.4 on page 13) for which a deterministic underload packet-flow pattern exists such that packets accumulate infinitely in the network. The possibility of such incidents occurring in large real-world networks has drawn much attention to the model[2] it is based on.

On the other hand, in practice, it is long known that networks experience self-similar and bursty traffic, and persistently full buffers begin to constitute a major problem at present [20]. Even the old notion of *congestion collapse* is brought into discussion again [20, page 65] and "there are some reports of congestion in the core" [20, page 62]. Network instability might be one of many factors in this challenge. However, instability itself is assessed rarely and research focuses on characterizing stability, in spite of issues in practice and the fact that networks cannot be easily spared non-stable conditions.

Besides instability which is caused by unfortunate circumstances, this issue has been considered as a possible security threat as malicious attackers might impose instability on networks. In particular, the Internet as a heterogeneous multimedia network [27, page 1695] and even wireless networks [41, 4.2.4 Threats to Availability, page 5] have been called at risk. Another work claims that "if a network is proven to

[1] We also give a more detailed overview on page 5 in section 2.3

[2] The adversarial queueing model which is introduced in the next chapter, section 2.1, page 4

be stable its users are ensured that this network is secure against malicious attacks" [26, page 445] .

Strictly speaking, these statements remain claims. The original model has been developed to provide robust results on stability [9, page 14] and hence makes several assumptions which make instability incidents appear more likely. Other assumptions such as infinite queue sizes may be sensible to study stability, as bounds on queue sizes can be derived under this condition, but they disagree with a direct application of instability results from this model. Hence, it remains unclear whether instable conditions are an effective threat to real-world networks and even the actual effects of instability are not well-defined.

A definite answer can only be given by experimental evaluation. However, experimenters need to know what to look for and several underlying questions need to be studied first. Firstly, we need to ascertain the situations and necessary conditions in which instability might occur. As a second step one might need to classify these conditions, if there are several possible ways to induce instable behavior. Third of all, infeasible model assumptions need to be approached and questioned whether results still hold under real-world conditions. Finally, the actual effects of instable behavior are to be determined.

1.1. Contribution and Structure of this Thesis

This work is intended as a preliminary step in understanding instability in order to fill the gap between theoretic results in this domain and the challenges in practice. The contribution of this thesis is three-fold.

Firstly, we characterize general aspects of instability beginning with simple queue build-up in underload situations and consequently describe requirements on how instability can be induced in our model. This characterization provides topological insights into the factors and requirements of instability and enables us to analyze previously published instability examples.

Secondly, we derive a class hierarchy for instability cases based on an analysis of previous instability examples. The classification includes all known FIFO instability examples and a class of strategies for which no instability examples have been published yet.

Thirdly, we introduce a finite buffer model and study instability to determine its behavior under these conditions. We discuss corresponding effects and present bounds on burst sizes of two of our classes.

The structure of this thesis is as follows: In the second chapter we consider related work on stability, introduce the adversarial queueing model and review previous work in this framework. The third chapter comprises the characterization of instability and the classification hierarchy. The fourth chapter introduces the finite model and describes the finite effects of instability. In the last chapter we conclude with a summary and state several open questions to outline possible future research directions.

2. Background and Related Work

The research on network stability dates back to Jackson in 1957 when he based his consideration for the steady-state of a Network of Waiting Lines [22] on early work by Erlang [18]. The state of stability of the considered systems was then called steady-state or approached as a search for equilibria. In 1975 Kelly expanded this result from the assumption of homogeneous customers to the equilibrium of networks of queues in which customers may be of different types [23]. Together with Kleinrock's pioneer work on packet switched networks [24],[19], also see [25] these results formed queueing theory and shaped the initial belief that instability might only happen in overloaded networks.

A network is commonly considered to be in an underload condition if the mean arrival rate of requests for any particular queue in the network does not exceed its processing rate. Though, the exact definition varies depending on the underlying model. In spite of Jackson's and Kelly's work, early examples in the 1990s Lu and Kumar [34], Kumar [30], and Seidman [38] showed several examples of non-existent equilibria in networks with underload conditions. Bramson[10] then showed the applicability to FIFO queueing networks, introduced the term instability and gave an overview and discussion of previously found instability examples. From his work we deduce that instability in these examples relies on the relaxation of the strict path-independence assumption[1]. Similar assumptions[2] were further called into question as measurements showed self-similar and bursty characteristics of network traffic [32], [36] which cannot be easily modelled with queueing theory.

First, stability was studied still under a queueing theory approach or with the use of fluid models. As a good introduction and overview we refer to the recent survey [11] where Bramson addresses the general question: when is a network stable? In [15] Cruz developed what was later called the Cruz Permanent Session Model, a non-probabilistic approach to model communication networks under the new finding of bursty traffic. While basic properties and stability have been studied in this model and Cruz' consideration played a major role in the development of network calculus [31] it was not until Borodin et al. introduced Adversarial Queueing Theory in 1996 [9] that deterministic analysis of stability became the focus of attention.

[1] That is, previous work assumed that an underload condition is sufficient, but [10] shows for FIFO networks: if the mean service time for a packet at a queue is dependent on its previous path ($\mu_{i,j}$ depends on j, page 415ff) then, even with constant arrival times (deterministic model, page 417), unstable networks can be constructed

[2] E.g: the path-dependence of inter-packet space or the Poisson-modelling of arrivals at all.

2.1. The Adversarial Queueing Model

As usual networks are represented as directed graphs $\mathcal{G}$ with the vertices meaning network nodes and the edges network links. Graphs are assumed to only allow simple-paths and edges comprise a queue at their origin where packets wait if they arrive at an edge which is already processing requests.

We consider discrete time steps in which each edge may process one packet each time step. Processing means that the packet disappears from the queue and appears in the queue of the next edge in the packet's path if it has not yet reached its final destination. If it arrives there, it is assumed to be absorbed, leaving the system.

In full queues the next packet to be processed is chosen by a scheduling policy $\mathcal{P}$. Example of such policies are NTG (Nearest To Go), LIS (Longest In System) or the well-known FIFO (First In First Out)-Policy. If two packets arrive at the same time a tie-breaking rule is used.

Besides the network representation, the model comprises an adversary $\mathcal{A}$, giving name to the model. The adversary injects packets which follow predefined routes, which are chosen by the adversary such that the stress on the network maximizes. All injections are subject to a load condition which ensures that the network is not trivially overloaded by the requests upon injection. In the original definition [9] we define $w = t_2 - t_1$ as the duration in time steps of the interval $[t_1, t_2]$ and $I_p(t_1, t_2)$ as the total number of packets injected during this interval using path p. Additionally, we fix an injection rate $r \leq 1$ which the adversary is not allowed to exceed under the following constraint:

$$\forall\, edges\ e \sum_{p:e\in p} I_p(t_1, t_2) \leq \lceil w * r \rceil$$

Later, Andrews et al. [3] expressed this constraint differently to express their leaky bucket approach. As it proved to be of the same power [37], we adhere to the original model.

Adversarial queueing theory now considers the time-evolution of the triple $(\mathcal{G}, \mathcal{A}, \mathcal{P})$ where $\mathcal{G}$ is the network topology, $\mathcal{P}$ the scheduling policy and $\mathcal{A}$ the adversary. Although the model was motivated by dynamic packet routing it was the first to study "queuing models where we do not essentially assume independent input streams" [8, page 14]. For this reason stability results in this model were considered particularly robust and stability was studied extensively in this model. The original model definition also coined the term universal stability which means the existence of a time-invariant bound on the number of packets in the system $(\mathcal{G}, \mathcal{A}, \mathcal{P})$. It can be considered from the view of a scheduling policy i.e. universal stability for a fixed $\mathcal{P}$ and any adversary and any topology. Alternatively, the topology can be fixed and stability is studied for any adversary under any scheduling protocol.

Besides the positive results presented next, this model also yielded various examples of instable networks confirming previous findings of the instability of FIFO even at arbitrary low injection rates.

2.2. Positive Stability Results

Besides instability examples some scheduling policies[3] and some simple topologies[4] proved to be universally stable for $r < 1$. Dependent on the longest path in a particular network Lotker [33] gives a lower bound on the injection rate needed for instability. Another approach [12], considers the number of interfering sessions[5] to give another bound on the injection rate ensuring stability. There are further approaches with similar directions such as pathwise constant injection rates. For an overview see [13, section 6.3, page 4469].

Furthermore, transformations seem like an obvious approach. For example, spanning tree algorithms could transform a network structure into a virtual tree. More sophisticated approaches propose implementing virtual universally stable topology on top of a given network by using turn-prohibition algorithms. See [39] as an example or an overview in [13, section 6.5, page 4470].

Finally, graph minors leading to instability have been studied [3], [21], [2], [40] and the polynomial decidability of universal stability of a network has been shown. Hence, forbidding unstable graph minors for networks might be an option to guarantee stability.

A survey paper from 2007 [13] concludes with the expectation "that in the next few years many more new results will appear, especially in the form of identifying situations where FIFO is stable [...] with work carried out on devising techniques for guaranteeing stability".

2.3. Applicability of Stability Results

As FIFO is (in variations) the standard scheduling policy in the Internet we consequently constrict ourselves to this policy.

Let us first consider stable topologies. Without doubt the previously named universally stable topologies are insufficient to build networks. Thus, ensuring not to use forbidden graph minors seems like the only solution to ensure stability from a topological viewpoint. In [40] the author presents a FIFO-specific decision algorithm for stability which yields a characterization of the topologies left to build FIFO-stable networks. However, this characterization leaves only decorated directed cycles (see Figure 2.1) as stable topologies. Due to the fact that a network is FIFO stable if and only if all strongly connected components are stable [40, Lemma 1], all subnetworks - in particular each autonomous system in the Internet - must comply to a decorated cycle structure to make the Internet FIFO-stable. This is infeasible and hence essentially renders this approach unuseful.

[3] SIS (Shortest in System), LIS (Longest in System), FTG (Farthest to Go), NTS (Nearest to Go) all in [3]

[4] rings[3], DAGs[9], trees

[5] the route interference number of any particular path is roughly the number of other routes in the system that share a common sub-path with the first-mentioned path

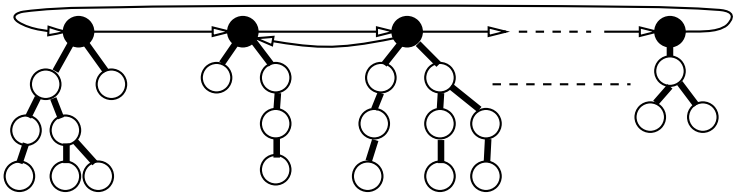

Figure 2.1.: The characterization differentiates between network circumferences (the maximal size of a simple circle) and if we focus on realistic ones, circumferences $k \geq 4$, then we obtain that only decorated directed cycles are left as stable topologies.
The figure shows a decorated cycle as discussed in [40, Lemma 7, page 89]. The thick edges are to represent a collection of at least one upward and one downward edge. This figure is courtesy of [40]

The last topological approach considers building virtual topologies on top of real world networks. However, this does not look like an efficient and practical solution [6] either, as performance would severely degrade due to this measure.

The most common approach to universal stability are the proven universally stable scheduling policies LIS, SIS, NTS and FTG. Though, besides practical implementation issues[7], a transformation to new scheduling policies is problematic due to the following reasons. Firstly, transforming the Internet en-block is infeasible. Secondly, the fact we state above prevents solutions for only strongly-connected components (such as ASs) of this network. Furthermore, "the composition of FIFO with any of the universally-stable protocols [..] is not universally-stable"[14]. Thus, also this third proposals does not provide a working solution.

Finally, also the route interference number as well as bounds on the request rates would severely degrade performance and usefulness of networks.

Assuming the completeness of this review, we claim that none of these previous approaches yields an applicable solution to enforce stability on real-world networks. Moreover, we conjecture for the reasons stated above and due to the heterogeneity of the Internet that stability cannot be given as a strong guarantee at such great scale. With that said, it seems remarkable how few work has been done to study instability itself.

2.4. Previous Work on Instability

Usually instability occurs only as a counterexample to stability and is not studied any further. Exceptions are rare. In [29] Koukopoulos et al. distinguish between two types of injections: investing flows and short intermediate flows. Weinard, while completing omitted proofs in the appendix of the full version of [40], defined "basic

[6] This is our own conjecture, no experimental evaluation has considered this yet
[7] for example for LIS: time synchronization, cheating

maneuvers that an adversary may perform, as we intend to use them several times in our instability proofs". Although both did not actually assess instability but used these statements to address typical stability properties and ease the specification of instability examples, these are first steps towards finding a form of building blocks for instability.

Another interesting work has been done [5]. Blesa investigates topological (n,n)-th extensions of 2-dimensional networks and shows that the advantage which the adversary can gain is limited. Although these were preliminary results on instability, this work is not directly applicable to FIFO networks as only the LIFO protocol is considered and only limited injection schemes are allowed. Besides, the measure Blesa used to assess the success of the adversary is the possible minimal injection rate. Newer results later showed that instability may happen at arbitrarily low rates.

In [33, page 49] Lotker et al. prove an upper bound for the stability for FIFO[8]. Even though the authors address a typical stability aspect, the methods used in this proof are very intuitive and constitute the first approach to generalize basic instability aspects.

Our work picks up some of these ideas and puts them in a focused context to study instability.

[8]The existence of an $r_{\mathcal{N}}$ for any network $\mathcal{N}$ such that FIFO is stable for any adversary with injection rate below $r_{\mathcal{N}}$

3. Characterization and Classification

In this section we will present a basic characterization of packet flow patterns leading to queue build-up, and how these patterns can become more efficient i.e. how a severe queue build-up can happen.

The first ideas to find "basic maneuvers" of adversaries go back to [40] and gave initial motivation to the approach presented here. However, our work follows different goals as we study instability and eventually want to assess what worst case adversaries can make possible.

3.1. Preliminaries

As it was pointed out in [3, 1.3. Preliminaries] it is easy to formalize the adversarial queueing model. However, it has been consent in the last decade to "keep definitions slightly informal for the sake of readability"[3]. Following this philosophy we state results and proofs in a less formal language than what might be expected and use the help of "the adversary" as though the system fights an attacker. Moreover, to understand the possibilities of an adversary we will often impersonate this opponent to consider on how to best attack the network.

Although it might happen that an actual attacker takes the role of the adversary, this consideration is merely meant to ease the description and readability and no actual aggressor is needed for the model's results. The model originally deals with possible effects caused by accidental packet flows causing the deterministic worst-case patterns considered in this context.

3.1.1. Definitions

Definition 1 (queue build-up strategy)**.** *An injection pattern of an adversary in the proposed adversarial model where at least one queue faces a queue build-up.*

Out first goal is to characterize queue build-up strategies. As the consideration can be applied to any queue one at a time we focus on one particular queue and call it's corresponding edge E_t. First, we tackle the question how a queue build-up can happen for E_t in spite of the bound the model poses on the adversary's injections. We begin with an inherent and apparent quality of the FIFO policy.

Lemma 1 (FIFO)**.** *In a network which uses a FIFO scheduling policy a packet that is arriving at a specific queue can only be delayed by packets that arrived earlier at this queue.*

Proof. Besides the fact that this is common wisdom: a packet cannot pass other packets while it is in a queue and due to the First-Come First-Served manner of FIFO queues it is evident that all packets which arrived earlier than any particular one will be processed first. □

This means that on a shared path packets cannot proceed faster than others that are in front of them. Furthermore, we can deduce that no packet can be delayed forever and will eventually arrive at its destination[1] .

The load condition constrains us to injections whose overall sum at the time of injection still can be processed by any edge in the network. In particular, edge E_t will always be able to cope with the requests introduced at injection time. Hence, overloading any edge is, at least, nontrivial. Before we follow that line further we define what we call a "set". It might slightly differ from the common sense of sets in different contexts. One might call it a packet flow but we treat it in mathematical set sense and so give this definition.

Definition 2 (set)**.** *Packets sharing the same route (path) are called a set if they belong to the same injection by the adversary.*

As all packets in a set share the same path and injection time, we now speak of the path and the injection time of a set.

3.1.2. Intuitive Description of Queue Build-Up

Due to the load condition in the model no simultaneous injections can overload any edge. Still, at E_t the adversary needs to accomplish a situation where arrivals with a rate greater than one are possible. An unexpected analogy can be found in modern artillery systems. As shown in Figure 3.1 when firing over a hill at a target, different firing angles are possible so that higher trajectories may take longer as lower ones. Then the target can be hit simultaneously[16, page 190f] despite an analogous load condition: the cannon only shoots one projectile after another.

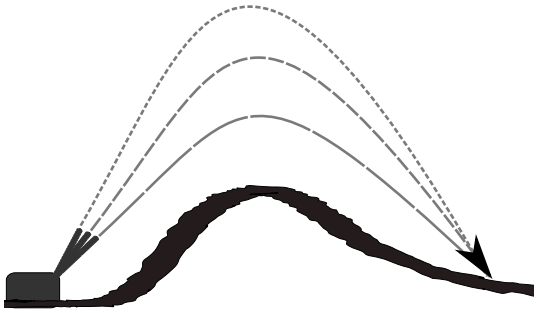

Figure 3.1.: Multiple Round Simultaneous Impact System

[1] We will detail and use this in the next chapter. For now it is sufficient to intuitively grasp this property of FIFO networks.

For our network model we have neither projectiles nor different trajectories. The adversary can use, however, different paths in the network. As a set cannot be divided up and merged due to its characteristics and Lemma 1 we need to merge packets from more than one set. In fact, various possibilities are given to the adversary to accomplish this in a network. In order to merge different sets they must have existed in the network at the same time while, due to the load condition, they must have been injected at different time ranges. How is this possible? Clearly, some sets need to reach E_t faster than others. This can be caused intuitively, by only two cases, passive or active delay. Either can one set have a longer path to travel. Or it can be delayed by being hold back in some form or another. We will now discuss these two possibilities.

First of all, let us consider the case where only the length of a path is used to build up a queue and where this approach works. Let us assume $r = \frac{3}{4}$ and the topology shown in Figure 3.2. The adversary injects a set S_1 into edge c_1 destined for X. It is injected four time steps long and thus contains three packets. Next, while those packets are traveling towards E_t the adversary starts another injection into edge b_1, a set S_2 with the same destination and again uses four time steps. Now all injections must pass E_t. The first packet of the first injection needs between six and seven time steps to reach E_t and in the fifth step the injection of S_2 starts so that in any case at least two packets reach E_t simultaneously. Eventually, a queue which contains one to two packets emerges.

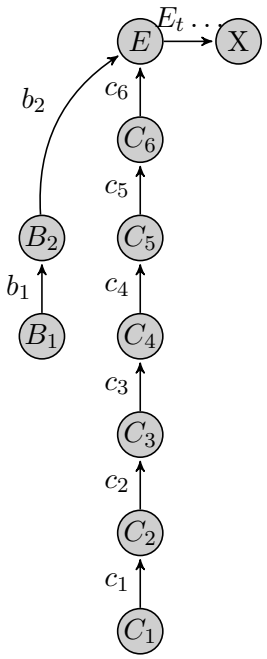

Figure 3.2.: Path-Length Example

It is important to see that more congestion can be accomplished by assuming that some initial sets already exist in the system before the adversary starts to inject packets. This might seem inaccurate, but firstly an equivalence of adversaries with and without initial sets is proven in [7, Fact 2]. Secondly, an inductive argumentation is used to show instability. Generally, examples assume that we start with one to many initial sets queued in a buffer which the adversary uses as what we will call a block later. When injecting packets behind a block they will not start to travel along their path right away. Instead, they will queue up and may work as a block for another injection. Sometimes topology allows the adversary to re-establish the initial situation with some sets in a similar configuration as the initial sets before injections started. If these new sets, then, are of larger size than the original sets it follows per induction that the adversary may go one forever. Topologies that can be exploited by adversaries in this way are the ones considered as instability examples. However, they are usually more difficult to understand than the previous injection scheme. This difficulty is another argument for a structured study of the factors of instability which we will begin after this next example.

Assume that the topology shown in Figure 3.3 has a set I queued in e_0 and $|I| = i$. This set has a remaining route of only e_0 and will be used as a block for the first injection. After this first injection took place in a first "phase" it is in turn used to block the injection of a second set in a second "phase".

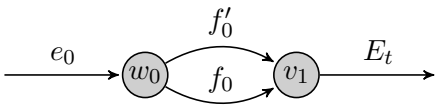

Figure 3.3.: Example topology to show the use of confinement and initial sets.

1. A set S_0 is injected in the first i time steps behind the block I into e_0. This is the time the initial set I needs to completely vanish as it traverses its last edge. At the end of this phase $r * i$ packets have been injected forming the set S_0. They were injected together with the path $e_0 f_0' E_t$ which they start to follow in phase 2.

2. Now, S_0 starts to follow its path. All packets of it will completely traverse e_0 in this phase making this phase $r * i$ time steps long. But its further progress is partly delayed by single-edge injections into f_0'. These mix with S_0. This is what we later call confinement, a mechanism to delay a set.
   Furthermore, S_0 is used as a block for another injection of a set S_1 wanting to traverse $e_0 f_0 E_t$. As the length of this phase is $r * i$ time steps the set S_1 has size $r * (r * i)$.

3. No further confinements are needed. For another $r*r*i$ time steps an injection directly into E_t takes place creating another set S_2 of size $r * (r * r * i)$.

Note, that in this example the path length of the merged sets S_0 and S_1 are equal and that we used single-edge injections as a delay mechanism. However, due to Lemma 1 some packets of S_0 will slip through the confinement. Thus, S_0 and the single-edge packets mix, a portion of each is hold back. Furthermore, observe that the load condition has not been violated and that when we put this topology into a larger context such as the baseball graph in Figure 3.4 a symmetric situation evolves so that further injections become possible. In fact, this example is part of the instability proof for the original baseball graph [3].

To compute the final queue size in E_t we need to compute the set sizes of sets S_0,S_1,S_2 and the number of packets that traversed E_t in phase 3. This is only stated briefly for completeness, the reader might skip this paragraph at first read.

S_0 initially had the size $r * i$. However, while arriving with rate 1 it mixed with the single-edge injection into f_0' at rate r. Hence, in $r * i$ time steps $r * i$ packets of S_0 arrive, while $r^2 * i$ confinement packets got injected into f_0'. This gives us

$\frac{ri}{r^2i+ri} = \frac{1}{r+1}$ as the portion of S_0 which traverses f'_0. So, the remainder of S_0 is:

$$|S_0| - \frac{ri}{r+1} = ri - \frac{ri}{r+1} = \frac{(ri)(r+1) - ri}{r+1} = \frac{r^2i + ri - ri}{r+1} = \frac{r^2i}{r+1}$$

Finally, S_1 and S_2 stay the same and those packets arrive and traverse E_t for $r^2 * i$ time steps. This yields an overall queue size for E_t as:

$$\frac{r^2i}{r+1} + |S_1| + |S_2| - r^2 * i = \frac{r^2i}{r+1} + r^2i + r^3i - r^2 * i = r^3i + \frac{r^2i}{r+1} > i \, (for \, r \geq 0.85)$$

As the result is larger than i for the specified injection rate we build up a queue which is larger than the initial one.

Delaying packets in an active way is more versatile than passive delay mechanisms. The adversary can delay packets longer in an easy way and does not need long paths. In fact, if we consider injections at rate 0, active confinement formally subsumes the longer-path model as consecutive single-edge injections would be able to simulate the same behavior. The question of the exact implementation of confinement is considered in Theorem 3.

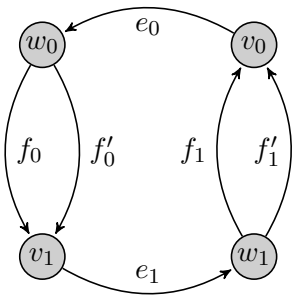

Figure 3.4.: A network called the baseball graph, a popular example to show instability in adversarial models

3.2. Queue Build-Up

The intuitive notion of instability and underload queue build-up will be collected in the following straightforward theorem.

Theorem 1 (Minimal Injection Scheme)**.** *Let E_t be the edge where the adversary want to cause a queue. A queue build-up strategy needs the following premises to overload edge E_t:*

Injection *Injecting of several sets that include E_t compliant to the load condition. At least one needs to have a different path from the others.*

Confinement *Delaying of subsets of some sets to give other sets time to catch up. All while preserving the load condition.*

Merge *Merging several sets necessarily from different injection time ranges so that the actual rate of arriving packets at E_t is strictly greater than one for at least one time step.*

Topology *The in-degree of edge E_t is greater than one: $indeg(E_t) > 1$*

Before we begin the proof, we admit that this first characterizations stays rather vague. This is due to the fact that the model grants the adversary elusively much freedom so that only limited abstractions can be made. Note, for instance, that ***Confinement*** is not unambiguous and may include various different implementations. So, after we stated the proof we will restrict our scope to more specific cases.

Proof. As we do not assume the case of sets at maximal injection rate, we have to deal with any number of sets sharing E_t in their path. This obliges us to argue on an atomic level i.e. consider single time steps. Given that all actions must be valid also on an atomic level, it follows that our consideration here is also valid on a larger scale.

By abuse of the previous notation, let $I_p(l)$ denote the rate of the injections using path p in time step l as well as fractions of packets injected. As we previously expressed the load conditions in integers and now will need to deal with fractions, we need to adapt interpretations to rates. This means that the load condition needs to be expressed as $\forall$ *edges* $e : \sum_{p:e\in p} I_p(l) \leq r$. Hence, even if we bundled all injections containing edge E_t and if they gave full packets, due to $r < 1$ it is impossible for simultaneous injections to overload any edge.

Furthermore, due to Lemma 1 even if we got several packets behind each other on the same route towards E_t they will not be able to catch up or even overtake each other.

Joining these previous observations makes it clear, that we need to merge sets that travelled paths that were different in at least one edge and were injected at different time ranges. So, to consider the most basic requirement for this, let us fix some time steps l,l'. Assume WLOG that l<l' and that E_t is included in paths of both $I_p(l)$ and $I_p(l')$. In order to get a higher rate, we want to merge these two injection sets i.e. they need to reach E_t at exactly the same time step[2]. Following the previous argumentation, it is not sufficient to just have different paths. Instead, $I_p(l)$ needs to be delayed so that at some point some packets of both sets reach E_t concurrently.

Finally, it is clear that for a queue to build up we need an arrival rate strictly greater than 1 as E_t processes packets at rate 1. This implies that the node cor-

[2]This might be relaxed for non-atomic considerations i.e. if we consider sets injected over longer time periods. Then, in order to merge sets their arrival time ranges need to overlap. However, in order to actually merge them in at least one atomic time step the described event must take place.

responding to the beginning of E_t needs to have at least two input edges serving packet streams whose rate sum is strictly greater than 1. □

3.3. Assumptions and Efficient Injection Schemes

It became clear that adversarial models give the adversary a high degree of freedom. For these reasons, absolute statements do not allow deep insights. Thus, we will restrict our scope on specific adversary strategies and topologies. In this work we will focus on how worst case injection schemes and topologies may look like. So, we will do several assumptions to make investigations simpler and easier to understand.

3.3.1. Assumptions

Assumption 1 (`MSS`)**.** *Each set with a path that includes E_t is injected at rate r. No parallel injections into E_t take place.*

Overall this means that we assume a maximum set size for all injections traversing E_t. It might seem clear that this does not affect the eventual load the adversary imposes on E_t. However, the confinement method used in the instability example in [4] shows that also very efficient injection schemes sometimes rely on injections which do not comply with this assumption. That we will nevertheless assume `MSS` lies in the great simplification this assumption allows. Furthermore, the insights from our following characterization is still applicable without `MSS` which can be seen when examining the same example in [4] after we finished our characterization.

Lemma 2 (Unique Injection Order)**.** *Under the assumption `MSS` injections are uniquely ordered as sets $S_0, S_1, \ldots, S_{n-1}$ and $n \geq 2$.*

Proof. As we know from Theorem 1 sets need to be merged from injections from different time ranges. Due to the assumption `MSS` we can consider exclusively those injections at full rate r. Hence, at any time step only one injection whose path includes E_t takes place [3] and so sets are uniquely defined by their position in the injection order. Suppose n sets sharing edge E_t in their path. Without loss in generality we can define the unique order S_0,S_1,S_2,...,$S_n - 1$ such that $I_b(S_0) < I_e(S_0) < I_b(S_1) < I_e(S_1) < ... < I_b(S_{n-1}) < I_e(S_{n-1})$. Here, $I_b(S_i)$ provides the time step at which the injection of set S_i began and $I_e(S_i)$ the corresponding time step when it ended.

Finally, as a single set cannot be divided and merged to get higher injection rates due to Lemma 1 it already became clear that we need to merge at least two sets. Thus, $n > 1$. □

To study how severe instability cases arise it is sensible to not consider possible small incidences of instability that might occur in a network. Hence, we will focus

[3] Formally, injection at rate 0 may be possible. We explicitly do not consider them here as they obviously do not have any impacts on the system.

on efficient injection schemes where the adversary's injections play well together and hence the least possible of packets should be wasted.

Assumption 2 (EFF)**.** *Suppose the adversary does injections in the period of time* $[i_b, i_e]$*. We assume the following goals of the adversary*

- *achieve a worst-case burst at* E_t *that builds up the longest queue as possible in* $[i_b, i_e]$
- *keep the needed topology as small as possible*

Hence, we assume efficient strategies.

In the next chapter we will find bounds on i_b and i_e in several cases. For the time being, let us assume that the adversary can choose these parameters according to the given topology. Due to the constraint to minimal topologies the adversary is forced to use active confinement for obvious reasons. This is justified by real-world network where we do not want to assume unnecessarily large structures.

Before we continue we would like to clarify the notion efficiency.

Lemma 3 (Interpretation Efficiency)**.** *Efficiency (EFF) means that the time span needed for merging*[4] *is to be minimized while the number of packets arriving in this period should stay the same (or even grow).*

The minimal topology property is kept intact by this interpretation.

Proof. As defined in the model, any edge, in particular E_t, serves at rate 1. Hence, after the first packet of the burst arrives the queue constantly looses packets. This means that decreasing the number of time steps needed for merging, leads to reduced runoff by traversals via E_t. If now the number of packets arriving in this period stayed the same (or even grew) then the queue contains more packets. Thus, this increases the efficiency of the injection scheme. □

Lemma 4 (Injection Paths)**.** *Under the assumption EFF all injections need to share distinct paths. Hence, all injected sets are distinct.*

Proof. Due to EFF the adversary must not waste any injections. If the path of one set S_i would be exactly the same as of any injected set S_j, S_i could not be merged with S_j due to Lemma 1. Even if it was so, then S_i could not be confined by any active methods while S_j is injected. To still confine S_i passive methods, i.e. longer paths, would be needed which contradicts EFF's "minimal topology" assumption. □

So, to achieve efficiency this means we need to have distinct paths for all sets. Besides, injections whose path include E_t must not have any gaps. For example pauses between injections of the adversary should not occur. This would trivially contradict EFF.

[4] The time when the merge of sets as described in Theorem 1 is done.

Finally, we state one additional restriction which simplifies things but is also motivated from practical considerations. A large burst of packets that arrives at one specific node in the network is considered worse than a distribution of this load over several nodes. Thus, we will deal with one node in the network where the adversary will cause the queue build-up to happen.

Assumption 3 (MERGE)**.** *We assume that sets S_i are merged exactly at E_t.*

These preparations now allow us to describe a specific class which we call "efficient injection schemes". The assumptions MSS, EFF, and MERGE allow us to describe the adversary's strategies in more detail than the universal but vague description in Theorem 1. Next, we list and proof some partial results leading to the final characterization Theorem 2.

3.3.2. Properties of Efficient Injection Schemes

Lemma 5 (Ideal Efficient Merge)**.** *To achieve EFF under the assumption MERGE the following statements are necessary properties :*

1. *at merge time all S_i have the same size*
2. *sets S_i arrive at E_t as bursts i.e. their arrival rate is* 1
3. *sets S_i arrive at E_t at the same time*

This Lemma does not state that these properties are sufficient to ensure EFF under MERGE. However, is says that they are necessary and applying them as rules to any injection scheme will make it more or equally efficient.

Proof. Suppose the adversary's injections follow a queue build-up strategy as formulated in Theorem 1. After any set traversed E_t its packets will never come back to this edge, due to the simple-path model. Hence, and due to MERGE and EFF's it is clear that several merges do not yield maximum efficiency. So, we consider just one merging period.

To achieve a merge of the maximal number of packets in the shortest time period, merging any two sets should take exactly the same number of time steps. Consider the case $n = 2$. Let L_i,L_j be the set of time steps the two sets S_i, S_j need to fully traverse the edge that is adjacent to E_t and is just before E_t in each set's path. Define $W = (L_i \bigcup L_j) \setminus (L_i \bigcap L_j)$ If $W \neq \emptyset$ the packets arriving in these time steps are wasted as packets from only one of the two sets arrive. Hence, we may interpret EFF as: the adversary's aim is to minimize W.

Now, consider the general case for n sets where a similar behavior can be observed. Basically, all previous considerations are also valid for this case. We define L_i, $i \in 0, \ldots, n-1$ to be the set of time steps of each set S_i when it traverses the edge on its path which is adjacent to E_t. Again, we define $W = (\bigcup_{i=0}^{i=n-1} L_i) \setminus (\bigcap_{i=0}^{i=n-1} L_i)$ in a similar way. Clearly, all time steps in the set W are not used efficiently. We argue that minimizing W is equivalent to becoming more efficient.

To clarify this point let us assume that $W \neq \emptyset$. This means that at least some set S_m does not yet or anymore provide any packets for the merging while at least one other set S_n already or still has packets that want to traverse E_t in the corresponding time steps. Or, it could be that some set S_m has intermediate time steps where no packets want to traverse E_t while another set S_n has a packet in this time step. We will deal with three different cases separately. Recall that the time steps of S_n, S_m where packets arrives are named L_n, L_m respectively. Now consider the possible classes of relations between L_m and L_n as shown in Figure 3.5.

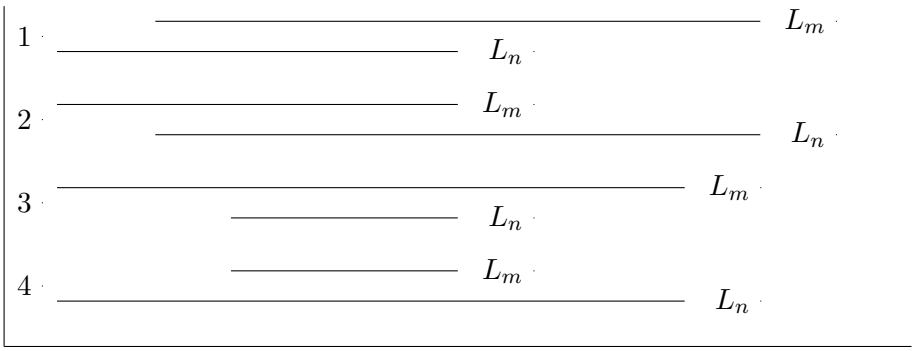

Figure 3.5.: To visualize the possible relations between L_m and L_n

Assume that the adversary had injected sets S_n and S_m and used confinement methods in a way such that L_n and L_m would begin at the same time step and applied this idea to all other L_is in similar relations. This method could resolve the case of shifted sets as the cases 1 and 2 show. Hereafter, the total number of time steps needed for the merge is equal or less to the case before. At the same time, the number of packets being merged has not been changed. Hence, the efficiency improved.

The adversary can do a similar step to equalize the set sizes. However, before we do this we will deal with the gaps in the flow of sets S_i. Assume the adversary confined packets in a way such that they arrived as a burst of consecutive packets, i.e. at rate 1. This leaves us with arrival time which may be shifted against each other and the idea from above may applied again. While we gain the same situation as above, in each S_i there are no more gaps and hence the overall time needed to merge these set again is equal or less to the case before while the number of packets stayed the same.

Finally, we want to equalize set sizes to deal with the classes 3 and 4. By shortening the time of some injection S_m and injecting more packets to form set S_n for instance the case 3 may be resolved. For case 4 exactly the same method may be used, the other way round. Repeatedly applying this algorithmic idea leads to equal sets which arrive without gaps at the same time step. Again, the total number of time steps needed for the merge is equal or less to the case before. And at the same time, the number of packets being merged has not been changed.

□

Further optimizations can be made to gain more efficient injection schemes. The most important part of this is choosing n which may be called the split-factor of injections.

Lemma 6 (Splitting Injections)**.** *Under* `EFF` *the adversary chooses a maximal* n*.*

Proof. According to Lemma 4 the adversary splits up injections into n sets which follow distinct paths. The efficiency as required by `EFF` is clearly related to this number.

To clarify this, suppose we fix a number of time steps l. Hence, the adversary may only inject an overall number of $l * r$ packets with a path including E_t. Further assume two integers f, g where $f >> g$. If we split these into small sets $\frac{l*r}{f}$ and they are merged at the same time they will achieve a fast queue build up in a really short time as the ideal merge time is exactly $\frac{l*r}{f}$ time steps long (Lemma 5). On the other hand, consider we had only g sets and each set is quite large, of size $\frac{l*r}{g}$. Then they would eventually yield the same number of packet arriving at E_t, but, the merge would need $\frac{l*r}{g}$ time steps.

As $f >> g$ it holds that $\frac{l*r}{f} << \frac{l*r}{g}$. Thus, the queue build up with the split factor g will be much slower. Furthermore, E_t will already process more packets in this time, so the queue build-up would also be less than in the case before. Hence, generally a maximal n should be chosen. □

For best efficiency the adversary chooses a maximal n i.e. the injections should be split up into small sets. In contrast to previous results we will add here that obviously n is determined by the network topology. This follows from Lemma 4 as we need at least n distinct paths to accomplish what is stated there. Hence, the maximum which the adversary should choose, according to this Lemma 6, is defined and existent. Actually choosing n in reality has to be approached as an optimization problem with various parameters. That it is very simple in some cases can been later, we will leave a corresponding algorithm as an open question.

3.3.3. Rounds

Adversarial instability examples are always described in an inductive form, as they want to prove infinite effects i.e. packets accumulate in the system without bounds. We already touched on this topic in section 3.1.2 and gave some preliminary introduction in the second queue build-up example. The time range $[i_b, i_e]$ which is used to describe Assumption 2 (`EFF`) is often referred to as a "round". This notion is intuitive to catch and clear from inductive considerations. The inductive steps for the proofs are indeed these "rounds": the time steps for which an injection strategy is described and after which is may be reapplied in either the same or a symmetric way. To get a better understanding and if not already done so, the reader might consider to read exemplary instability proofs such as [3], [17], [28], [33], [4], [21], [2],

[40]. As we will need the formal notion of initial sets and refer to this later, we now roughly define what we understand as one round.

Definition 3 (Round)**.** *A round is the period of time $[i_b, i_e]$ when the injections of some adversary strategy take place. If at i_b, m sets are in the network we call these sets initial sets I_i, $i \in 0, \ldots, m-1$.*

It may only hold for some particular injection schemes that this period of time is indeed well-defined. We will turn towards this topic in the next chapter in more detail when we will give bounds on the length of this interval in several cases. For now, we argue that this definition is sensible due to Lemma 1 and the implications described below it: no packet can be delayed forever.

3.3.4. Characterization of Efficient Injection Schemes

Previous lemmata gave us an characterization of how injections should be made (Lemma 2, Lemma 4) and how the prerequisites for final merging should ideally look (Lemma 5). This leaves us with the question of implementing those and which specific actions the adversary may use to achieve the goal to create worst-case bursts. We still assume hypothetical topologies which only affect us via the parameter n and allow these worst-case considerations to take place.

Theorem 2 (Efficient Injection Scheme)**.** *Suppose the adversary wants to inject $n > 1$ sets (see Lemma 6) to congest queue E_t under the assumptions `MSS`, `EFF`, and `MERGE`.*

Then the corresponding injection scheme is characterized as follows:

Injection *The injected sets S_1,S_2,S_3,...,$S_n - 1$ follow n paths which are pairwise distinct in at least one edge. Sets are of equal or decreasing size for $i \to n$*

Blocks *Each set S_i is injected behind another set B_i which works as a block and is of size $\mathcal{O}(\frac{|S_i|}{r})$.*

Confinement *The confinement methods assure that a set's packets are hold back sufficiently long such that all sets in the order behind S_i get injected (and possibly reach E_t) i.e. S_i is confined for (at least) $\sum_{j=i+1}^{n} \frac{|S_j|}{r}$ time steps.*

For the first set, S_0, ***Blocks*** leaves only one possible case: S_0 is injected behind some set B_i which is an initial set.

Proof. From `MSS` we draw the unique order on injection sets (Lemma 2). The adversary will inject them according to Lemma 4 which already proves the first statement of ***Injection***.

Next, we prove that sets will not grow once the injection of the next set has started. At first sight, this might be done very well although it might ruin the injection order of distinct sets. So, suppose the adversary injects some set S_j together with the

path p. Then the adversary starts the injection of some other sets. Because sets are defined by "having the same route (path)" in Definition 2 the only possibility to let set S_j grow would then be to inject a set S_k together with the same path p. Obviously, due to `MSS` no active confinement actions could take place in this period. Since S_j still needs to be confined for at least the same number of time steps (Lemma 5) and only passive confinement methods are available this would require a longer path, contradicting the minimal path assumption from `EFF`. Hence, we established that injections do not grow after their injection is paused or ended . Owing to this and to 1. in Lemma 5 sets need to be injected already with equal size. Besides, confinement may be lossy[5]. This means that, to actually achieve equal set sizes at merging time, sets of lower index may need to be larger.

Packets start traveling through the network once injected. Because of `MSS` it is not possible to confine sets while they are injected. But we need to confine sets as efficiently as possible to have time for further injections while not needing large topological structures (`EFF`).

Assume we do not use blocks and we inject the set S_j of size $|S_j| = x$. As described no active confinement can take place. Hence, while being injected the first packet of the set would traverse x edges of its path and it would be followed by the rest of the set. In particular for large[6] x the set would just run off. The adversary would need tremendous topological structures to passively confine it until the other sets are injected or lose parts of it. Either case contradicts `EFF` and shows that blocks are required.

A block basically consists of a set which is already queued up in the edge where some set is injected before this injection starts. It is clear that for efficiency reasons the adversary needs to block as many packets as possible while they are injected. Assuming the adversary injects some set S_j and wants to block a constant fraction $f \in \mathbb{N}$ of the injected packets i.e. $\frac{S_j}{f}$ should remain.

Let us call the corresponding block B_j. By `MSS` the injection rate is r and S_j needs $\frac{S_j}{r}$ step to be injected. So, to hold back $\frac{S_j}{f}$ packets $\forall f \in \mathbb{N}$: $|B_j| = \frac{S_j}{f*r} \in \mathcal{O}(\frac{S_j}{r})$ as B_j needs $|B_j|$ time steps to traverse the injection edge and $|B_j| * r = \frac{S_j}{f}$.

Finally, suppose the adversary injects $n \geq 2$ sets with the order from Lemma 2 and under the characterization we already proved for this theorem. Packets travel along theirs paths as characterized in Lemma 4 and injections are split up according to Lemma 6. Following Theorem 1 sets need to be confined before they reach E_t. Finally, due to Lemma 5 all sets S_i need to arrive at E_t simultaneously.

So, consider some set S_j. It needs to be confined until all sets, which were injected

[5] Lossy in the sense that a subset of some set S_i is insufficiently confined. This subset may travel towards E_t to fast reaching it before merging is planned, creating a larger W as defined in the proof of Lemma 5. This will be discussed in detail in the next chapter, in 4.4.2 on page 31

[6] Large set sizes do certainly occur due to unbounded growth or just because of large network structures.

after S_j (all sets S_k $k > j$), are injected and reached E_t. Because of previous considerations we know that it takes $\frac{S_k}{r}$ time steps to inject this set S_k. This means S_j must need at least $\sum_{k=j+1}^{n} \frac{S_k}{r}$ time steps to arrive at E_t after injection.

□

3.4. Classification

Having given a characterization of efficient injection schemes we will now introduce our classification of injection strategies. Classifying injection schemes, for instance of previously published instability examples, is important for two reasons. First, it allows us to consider bounds on the possible burst sizes in each section separately. This is necessary as some classes do not allow bounds of closed form while tighter bounds on other classes can be derived due to unique properties. Second, classification is an important tool to prepare for an algorithm which may examine real-world networks to asses instability risks.

A first approach could have been the well-discussed graph minors [21], [6], [40]. These allow use to decide the stability of FIFO network in polynomial time. Why shouldn't they yield a good start for a classification of injection schemes? This is due to the following reason: for large topologies which may be composed of various and mixed graph minors conditions change. Large structures provide the adversary with far more possibilities to create large-scale bursts of incomparable size and with completely different methods than what is possible in graph minors. While we cannot yet assess all the additional possibilities our classification provides a starting point for this direction.

As a first step we will use the findings of Theorem 2 for a classification of previously published examples. Although we needed three assumptions to prove this theorem, the basic properties of efficient injection schemes are useful for an even broader class of adversary strategies. For instance, [33] and [4] do not adhere to `MERGE` and `MSS` but can still be described under the found characterization. So, we will use the three components ***block***, ***injection***, ***confinement*** as a start to partition injection schemes although the proof of a similar theorem without the assumption would be quite more tedious and various case differentiations would be needed. The following list is exhaustive to catch how components were implemented in previously published injection schemes but does not satisfy a general case. However, Table 3.1 already shows that various different methods are already in use for instability proofs.

As a next step we graded some well-known instability examples derived from this list. In Table 3.2 we incorporated the factor of which the size of the initial sets is multiplied after one round, the classification according to Table 3.1 and the examples are put in an ascending order compliant to our class structure. Interestingly, this orders them according to a performance measure which may be defined as the "1-Round Factor" for a fixed r. Another possible, valid, and commonly-used measure would be the minimal injection rate needed for instability.

As an overview, the table can be divided into injection schemes from three de-

block
1. $B_0 \in \bigcup I_i$, $B_i = S_{i-1}$ for $i > 0$
2. $\forall i \geq 0 : B_i \in \bigcup I_i$

injection
1. injections into one single edge
2. injections into several edges: jumping around, none to weak coherence
3. injections into several edges: wandering along some path in the network, strong coherence

confinement
1. single-edge
2. single-edge + merge with I_i
3. single-edge + merge with I_i + merge with S_i (e.g. S_j with S_k, $j \neq k$)

Table 3.1.: A preliminary classification approach of published FIFO instability examples.

Network	1-Round Factor	Block	Conf.	Inject.	Cit.
Graph Minors H_1, H_2	$2r^4$	Class 1	Class 1	Class 1	[21]
Baseball	$\frac{r^2}{r+1} + r^3$	Class 1	Class 1	Class 1	[3]
Graph Minor $\mathcal{A}, \mathcal{A}^+$	$r^6 * \frac{2+r}{1+r}$	Class 1	Class 1	Class 1	[40]
Graph Minor $\mathcal{B}, \mathcal{B}^+$	$r^6 * \frac{2+r}{1+r}$	Class 1	Class 1	Class 2	[40]
Graph Minor $\mathcal{C}, \mathcal{C}^+$	$r^8 * \frac{2+r}{1+r}$	Class 1	Class 2	Class 2	[40]
Extended Baseball	$\frac{(r^2+r)(r^3+2r-2)-2}{r(r+2)+2}$	Class 1	Class 2	Class 2	[17]
Extended Baseball	$\frac{r+2r^2+3r^3+r^4+r^5)}{(1+r)(2+r^2)}$	Class 1	Class 2	Class 2	[28]
Gadget Chain	$\frac{r^3(1+\epsilon)^M}{4}$	Class 1	Class 3	Class 3	[33]
Multi-Dim Gadget Chains	$\frac{\alpha r}{4k}$	Class 2	Class 4	Class 3	[4]

Table 3.2.: Classification of published FIFO instability examples. The last two are constructions with the parameters M and α, k respectively.

velopment phases. The first phase was coined by the first example of adversarial instability by Andrews et al. in 2001 [3]. Similar techniques were used to prove instability for several graph minors. The used topologies were rather small, around four nodes, and single-edge injections are the prevalent confinement method.

While searching for a lower bound on the instability injection rate, topologies grew yielding new injection and confinement methods. Then, initial sets were not only used as a block for the injection of S_0 but also for confinement. For instance, this phase includes the FIFO graph minors $\mathcal{C}, \mathcal{C}^+$ in [40] with more than six nodes.

So far, the final phase in the design of instable topologies was the introduction of parameterized large structures. The proposals underlying the last two entries in the table, gave rise to further adversarial strategies and minimal injection rates dropped first to $0.5 + \epsilon$ and finally to $0 + \epsilon$ for infinitesimal $\epsilon > 0$.

Although from an historical viewpoint these three phases are distinct, basing a classification on this approach would overlook the most important factor: the blocking technique. In fact, classifying injection schemes by ***block*** class 1 versus ***block*** class 2 is sufficient to give us two unique classes which share strong similarities. More justification to this can be drawn from the next chapter when we approach bounds on the burst size of these classes. Besides this clean classification we must not forget that various further techniques are possible in all domains, possibly yielding several additional classes.

Definition 4 (Injection Schemes Classes)**.** *We define the following classification for injection schemes:*

Class $\mathcal{C}_1$ *:= {injection scheme* $\mid B_0 \in \bigcup I_i, B_i = S_{i-1}$ *for* $i > 0$ *}*
"traditional injection schemes"

Class $\mathcal{C}_2$ *:= {injection scheme* $\mid \exists k \in \mathbb{N}_0 : \forall S_j$ *with* $j \leq k : B_j \in (\bigcup_{i \leq k} I_i \cup B)$ *and* $\forall S_j, j > k : B_j \in \bigcup_{i>k} S_i$ *}*
"sophisticated initial sets"

Class $\mathcal{C}_3$ *:= {injection scheme | under the assumption* `EFF` *}*
"efficient injection schemes"

Class $\mathcal{C}_\infty$ *:= {injection scheme}* $\setminus \mathcal{C}_3$
"non-efficient injection schemes" (remainder-class)

Where for the second class $B := \{injectedsets\} \setminus \bigcup_{i>k} S_i$ *i.e. all sets injected in the current round but not one of the queue build-up sets* S_i*.*

First note that $\mathcal{C}_1$ and $\mathcal{C}_2$ are not disjoint and the same holds also for $\mathcal{C}_3$. Furthermore, if we set $k := 0$ then $\mathcal{C}_1 \subset \mathcal{C}_2$ and if we use `EFF` to label all efficient injection schemes, it holds that

$$(\mathcal{C}_1 \cup \texttt{EFF}) \subset (\mathcal{C}_2 \cup \texttt{EFF}) \subset \mathcal{C}_3$$

Besides, none of these classes is empty. That $\mathcal{C}_\infty$ is non-empty has been shown in the introduction of this chapter. Examples for $\mathcal{C}_1$ are the first eight entries in Table

3.2, and the last entry is an example for $\mathcal{C}_2$. The third class $\mathcal{C}_3$ is non-empty and of different nature as the previous classes. This is shown with an example in Appendix A and with an example in the next chapter. For the first three classes examples and general topology forms are also found in the Appendix A.

The characterization above and this classification completes this chapter. Together they gave an overview on which factors make adversarial network instability possible. Besides we learned how efficient schemes can be classified into a class structure. In the next chapter we will focus on the classes $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ and consider possible burst sizes in a finite adversarial model. This final step then shows that this structure puts injection schemes in an ascending order according to their worst-case behavior.

4. Finite Effects

In the previous chapter we characterized injection schemes, in particular efficient injection schemes, and introduced a classification approach. Now, we assess the actual effects arising in instable networks if instability occurs over long time ranges. Previous work considers instability as an unbounded queue build-up over time under the assumption of infinite buffers. As pointed out in the introduction we argue that this assumption directly affects the infiniteness of the results and so, to study instability effects, we propose an extension to the model: a finite buffer size. Hence, the first step in this chapter is to present our finite adversarial model. Consequently, we briefly consider what happens in this model if instability occurs over a long time. The rest of the chapter is dedicated to studying finite instability effects and to prepare bounds on the effects of instability in the classes from the previous chapter.

4.1. The Finite Adversarial Model

As usual, networks are represented as directed graphs with the vertices meaning network nodes and the edges network links. Edges comprise a buffer at their origin where packets wait if they arrive at an edge which is already processing requests. This structure is extended with a uniform buffer size b. As we assume uniform packet sizes we may express b as a number of packets. As expected, if any particular buffer is full while new packets arrive, these packets are dropped and occur as loss to the system.

Again, an adversary is used to model worst-case behavior under the known load condition: $\forall\, edges\, e \sum_{p:e\in p} I_p(t_1, t_2) \leq \lceil w * r \rceil$ where $w = t_2 - t_1$ and $I_p(t_1, t_2)$ is the total number of packets injected during $[t_1, t_2]$ using path p.

For the possible scheduling policy we will focus on FIFO, the First In, First Out policy. This algorithm is commonly used in practice and thus represents best the networks we want to analyze.

As before, we consider discrete time steps in which each edge may process one packet each time step. Particularly, each time step of the system will consist of the injection of packets, of packets which traverse edges according to their path and position in a buffer, dropping of packets if they arrive at a buffer which is already full, and finally the absorption of packets if they reach their destination.

4.2. Finite Instability Effects

To understand the use and some consequences of our extension to the original model we will briefly recall an instability incident. As we learn from previous work, instability is determined by inductive effects. Assume several sets of packets (initial sets) which are queued in some queues in the networking system. With the use of some injections and over the time span one "round", the adversary builds up queue sizes which are strictly greater than the initial ones were at the beginning of the current round. However, the adversary ensures that these new sets are still in the same or symmetrically the same positions and hence fulfill inductive assumptions for further rounds where the adversary ensure the same course of events. Consequently, packets accumulate in the system.

Due to the finiteness of buffers we introduced into the model, the temporal evolution can be divided into two phases. In the first phase, buffers, in particular bottleneck buffers on the path of the injections, face periodic bursts which grow larger with each round, see Figure 4.1 for time steps smaller than ca. 1500. Hence, even though the injection rates might be arbitrarily small and are not bursty in themselves, bursts are created by the network itself i.e. by the packet flows modelled by the adversarial injections.

Once several buffers[1] are full, the system arrives in a second phase where packet loss occurs. In previous adversarial models the number of packets in the system can grow without bound over long time-spans. But with the introduction of finiteness, buffers can never be greater than b. Thus, if a packet burst arrives at full buffers it translates into loss as Figure 4.1 shows for time steps greater than ca. 1500.

Lemma 7 (Loss)**.** *In a queue build-up strategy, once, set sizes reach buffer size, inductive growth stops and further rounds look exactly the same as increases in set size translate into loss.*

Proof. When the initial sets reach buffer size they cannot grow further as there is now space left in the respective buffers and packets are dropped. Hence, each round, the largest initial set size is exactly the same, as larger initial sets cannot be stored in buffers.

All packets which accumulate each round due to additional injections will be thrown away as they do not fit into their buffers. □

Apparently this is different to the infinite effects that previous adversarial models predicted. However, this result is far from unexpected, and in practice or different models this has been well-known before as we modelled highly bursty traffic and it "is well understood [...] that bursty traffic produces higher queueing delays, more packet losses, and lower throughput"[1, I. Introduction, page 1157] and it is known that "in the case of bursty traffic where a burst of packets arriv[es] at a full router it is likely to be partially or entirely lost"[35, 1. Introduction, page 308]. Nevertheless, adversarial instability has not been studied in this context.

[1] The buffers which contain the initial sets at each beginning of a round

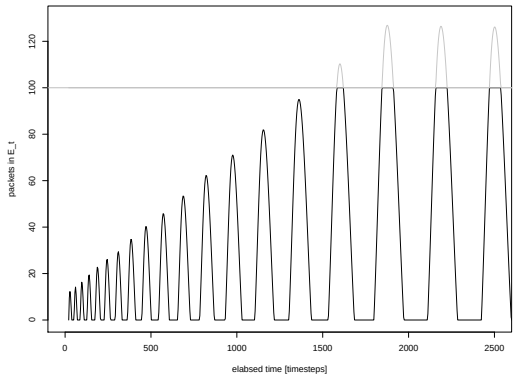

Figure 4.1.: The buffer of E_t for adversarial injections into the baseball graph starting with an initial set of size 10 (Spline interpolation of predicted buffer states over time). This buffer faces periodic bursts which are growing until they reach buffer size (100 packets) and further growth will be translated into loss. Loss is suggested by the light-grey lines above 100 packets.

In the first phase, queue sizes grow according to how many packets the injections accumulate each round, as in the original model. The amount of loss that occurs in the second phase is determined by the same number. Thus, it is an interesting question to quantify possible burst sizes in our finite adversarial model.

4.3. Bounding Bursts

Before we finally assess bounds of possible bursts in the different classes let us first detail what we actually want to bound. Without further specification, finding any bound on bursts is rather easy.

Imagine, we want to bound the arrivals over time in the most general case. That is, we do not pose any constraints on neither adversary or topology apart from the load condition. Then the time efficiency of the adversary's injection is bounded. By time efficiency we mean the comparison of how long it takes for the adversary to build a queue of a specific size: $\frac{\#\ packets\ in\ burst}{\#\ time\ steps\ needed}$. This efficiency is always bounded by r the maximum injection rate due to the model definition and under arbitrarily long time-scales.

A second example on a simple bound is also a bound on arrivals over time from the limited viewpoint of one node on short time intervals. The arrival rate at any node is always bounded by the number of input edges into that node, i.e. its in-degree. Although this might be a trivial observation it is valuable for two reasons. First, this viewpoint gives important intuition for later, when we consider a bound on injection schemes in class $\mathcal{C}_2$. Second, together with the example above, both "bounds" give

rise to the following question: What is the total number of packets that can arrive at any node in the network as a burst and may build up huge queues?

What, at the very heart of adversarial instability examples, really bounds the burst size is the time available for the injection. If we got only x time steps available for injections, then only $x * r$ packets will be injected whose path includes E_t. Thus, the "available time" needs to be determined. If new injections happen only slower than the rate older injections disappear, they are of small use to the adversary. This is directly related to confinement and blocking actions as these are the only ones holding back old packets from arriving at their destination. Hence, their efficiency determines how much time it takes until packets will be lost for the final queue build up[2].

4.4. Bounded Confinement

We have shown in Theorem 2 that a block should be of size $O(\frac{|S_i|}{r})$ to successfully hold back at least a fraction of S_i. To achieve maximum efficiency we first focus on the case where the whole set is hold back. This is what we call "full confinement".

We abstract from any specific confinement method. In fact, there are quite many different ways to confine a set, for examples compare the published instability examples. Finding the actual implementation of a worst-case confinement scheme is not in the scope of this work and is left as an open question. First, we need two basic definitions.

Definition 5 (Confinement Path)**.** *Suppose the adversary wants to confine a set S_j on a specific segment of its path. This path is then called confinement path and its length is called l_c.*

As each edge has its own queue and as the overall confinement time is clearly dependent on l_c we approach the confinement bound in a per queue consideration. Hence, the longest path in a network seems an important property of a network. Besides, for further applications we also need to consider the in-degree of nodes. Summing up, we need the following parameters to express our bounds.

Definition 6 (Bound Parameters)**.** *The bound parameters are b (the uniform buffer size), l (the longest path in the network system), and indeg (the maximum in-degree on any node in the network).*

4.4.1. Full Confinement

Full confinement yields a straightforward bound on the case where we want to block a set completely without loosing any of its packets.

[2]Reconsider the set W in the proof of Lemma 5

Lemma 8 (Bounded Full Confinement)**.** *In an efficient injection scheme (Theorem 2) the time a set S_j can be held on the confinement path of length l_c as a whole is bounded.*

The maximum time any set can be fully hold back is $(b-1) * l_c$.

Proof. Suppose the want to confine the set S_j which is arriving at the first queue of the confinement path of length l_c. In the worst case any packet can be delayed for at most $b-1$ time steps at one queue. If the queue was full when it arrived the packet would be dropped which is obviously not efficient (`EFF`).

It is sufficient to consider the first packet of S_j as due to Lemma 1 no other packet of S_j can travel faster. This packet will leave the first queue after $b-1$ time steps. If it already traversed k edges and needed $(b-1) * k$ steps it would in the worst case also need another $b-1$ steps to traverse the next edge on its path. By induction follows the statement. □

As we only focused on bounding confinement, we did not describe how the adversary actually implements a confinement method which may be comparable efficiency as in the lemma. In particular, we purposely[3] left out, how filled queues can be created all along the path with the right timing. This might have lead to the notion that injections into edges of the confinement path are not needed for this form of confinement. However, this is wrong. To prevent the need for large topological structures (`EFF!`) we need active confinement as shown before.

From the previous chapter one may guess that only full confinement may be used due to `EFF`. In contrast to this believe, lossy confinement is an equally important topic as not only sets S_i may need to be hold back. This is considered in section 4.5.2.

4.4.2. Lossy Confinement

To consider lossy confinement in an abstract form we focus on the leak of the confinement path after some time t which is longer than what is achievable by Lemma 8. Hence $t > (b-1) * l$. Deriving a formula on the leak is done by finding a bound on the slowest rate the packets of a set can be forced to. Then, multiplying this by the time we need to confine the set give us the leak.

To find the slowest rate, we first look at how far two packets can be spread out and start with the following definition.

Definition 7 (Distance)**.** *Assume two packets p_v,p_w of S_j which were injected one directly after the other (still compliant to the adversary bound!). The distance between p_v and p_w on S_j's confinement path is measured in the number of packets which are queued between them.*

After the second packet, p_w, traversed k edges of the confinement path define $dist(k)$ as the distance between p_v and p_w.

[3] An efficient injection schemes of general form which achieves that and scales even with large set sizes is not yet known.

It is not yet clear if this distance is actually well-defined as it might be different for any two pairs of packets in the set S_j. As a matter of fact, it is a definition which may be only valuable in the context of the next lemmata as we consider it under further assumptions.

Lemma 9 (Worst Case Distance)**.** *A set S_j arrives at rate 1 ,i.e. without gaps, at a confinement path of length l_c. The worst case distance after traversing the confinement path is:*

$$dist(l) \leq (indeg + r - 1)^{l_c+1} - 1$$

For the subsequent proof we assume a specific fixed processing order for the traversal phase of each time step[4]. This is without loss of generality as the exactly same proof works for any other processing order also, as long as it is stable.

However, instable processing orders are not explicitly forbidden in the model and may be considered. Then, this lemma only holds for long periods of time where we consider averaged data. For subsequent bounds, even this would be enough but we do not consider it explicitly here.

Proof. This lemma is proven by induction over the number of edge some packet has traversed. Let $(E_1E_2 \dots E_{k-1}E_kE_{k+1} \dots E_{l_c-1}E_{l_c})$ be the current confinement path of S_j and call the last queue before this path E_0. Although we pictured seven edges it is not assumed that $l_c \geq 7$. It is just done to improve readability.

Finally, we fix a processing order for traversals: edge E_0 is always the first to be processed for its traversals.

For the induction basis suppose the current time step is t and some arbitrary packet $p_v \in S_j$ is in the first place at the queue at E_0. It will arrive at E_1 in the next step $t + 1$ as shown in Figure 4.2. Due to the assumption "no gaps", the next packet $p_w \in S_j$ was queued behind p_v in t and will arrive at E_1 in the time step $t + 2$.

Imagine that between the arrival of p_v and p_w at E_1 i.e. the time span from the beginning of the processing of traversals in time step $t+1$ (1.) to this phase in time step $t + 2$ (1.), packets can arrive at E_1 queueing up in between the two packets. The following packets can queue in E_k before p_w arrives in $t + 2$:

Still in time step $t+1$, but just after p_v arrived: arrivals from each input edge into E_1 which are not shown in Figure 4.2. Then, in time step $t + 2$, the model allows for an injection which is done before traversal are processed: a single-injection[5] into E_1.

Overall, this will be $(indeg - 1) + r - 1$ packets in the worst case as each input edge except E_0 serves confinement packets at rate 1, injections take place at rate r, and one packet traverses E_1 in $t + 1$.

[4] compare the model definition in 4.1 on page 27

[5] Formally, one could consider several different injection schemes for this. The only important aspect here is that the rate will always be bounded by r and that the adversary is not able to inject packets into each edge which want to traverse the whole path. Hence, we model this by single-edge injections.

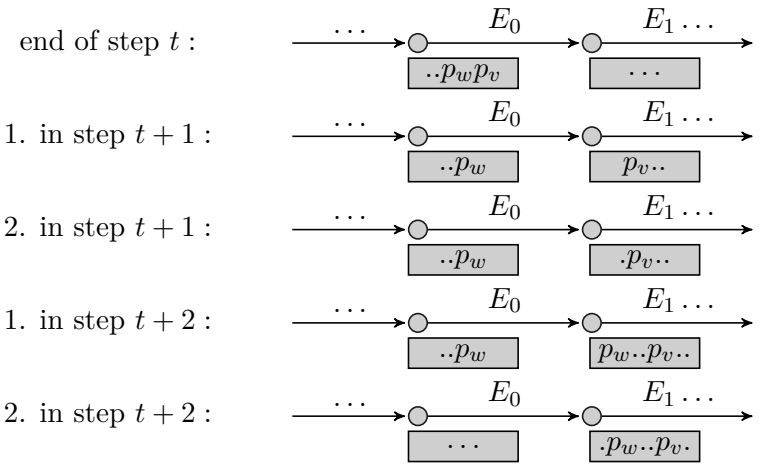

Figure 4.2.: The confinement path without additional input edge where the traversal of the first edge in the proof of Lemma 9 is shown. Time step $t+1$ is split into two parts.

Now, p_v and p_w do not travel on the confinement path consecutively any more. They traversed 0 edges of the confinement path, and hence the basis is proven as $(indeg + r - 1) - 1 = (indeg + r - 1)^{0+1} - 1 \geq dist(0)$.

For the inductive step now assume that p_w already traversed n edges and that $dist(n) \leq (indeg + r - 1)^{n+1} - 1$. So, we now consider what happens if p_w traverses another edge.

Suppose we have a similar situation as in the induction basis, only that there are already $dist(n)$ packets between p_w and p_v. Furthermore, we now consider edge E_k and E_{k+1}, and we are in a later time step which we call t_n. The packet p_v will traverse E_k in step $t_n + 1$ and the packet p_w will do so not later than in time step $t_n + 1 + dist(n)$.

In the interim period, there are again rate-1-arrivals from each input edge into E_{k+1} except E_k itself. Besides, there are also single-edge injections into E_{k+1}. Thus, to compute the overall number of packets queued between p_v and p_w, we need to sum the old packets, the rate-1-arrivals for each input edge per time step, the injections per time step, and finally we need to subtract the number of traversals. As usual, each time step one packet traverses E_{k+1}. This leaves us with the following equation:

new dist:	old pcks	(# time steps)∗[arrivals, injections]	traversals of E_{k+1}
$dist(n+1) \leq dist(n)$		$+ (dist(n)+1) * [(indeg - 1) + r]$	$-(dist(n)+1)$

After substitution according to the inductive assumption:

$$\begin{aligned} dist(n+1) &\leq ([(indeg + r - 1)^{n+1} - 1] + 1) * (indeg - 1 + r) - 1 \\ &= (indeg + r - 1)^{n+1} * (indeg + r - 1) - 1 \\ &= (indeg + r - 1)^{(n+1)+1} \end{aligned}$$

Any two packets of S_j are indistinguishable except by their order on arrival at E_1 and they arrive at the same rate. Thus, $dist(n)$ is the distance between any two packets of S_j after the latter traversed n edges. As the confinement path was assumed of length l_c, $dist(l_c) = dist(n)$ for $n = l_c$ which means that the packet traversed l_c edges.

□

This lemma now enables us to state a theorem which gives us the size of any set after it was confined lossily for a specific number of time steps.

Theorem 3 (Bounded Lossy Confinement)**.** *For lossy confinement of a set S_j on a confinement path of length l_c where packets of S_j arrive without gaps, the remaining set size after $t > (b-1) * l_c$ time steps is at most $|S_j| - leak(t)$ with the leak over time:*

$$leak(t) \geq: \quad \lceil \frac{t - l_c * (b-1)}{(indeg + r - 1)^{l_c+1}} \rceil$$

Proof. Let $l_c \in \mathbb{N}$ be fixed, the length of the confinement path. Besides, fix the buffer size b, let $t > (b-1) * l_c$ be the time we want to confine the set S_j. As shown in Lemma 8, t is strictly greater than the maximum time available for full confinement. Hence, some packets will already leave the last edge and continue their path through the network. This means our set size is reduced by some leak, so the remainder looks like $|S_j| - leak(t)$ where $leak(t)$ still needs to be defined.

As we know from Lemma 9 the distance between two packets after traversing the confinement path under such conditions is $dist(l_c) \leq (indeg + r - 1)^{l_c+1} - 1$. After the first packet of S_j arrived at the first place of the queue of the last edge of the confinement path at time step $(b-1) * l_c$ we have $t' = t - (b-1) * l_c$ time steps left where S_j should be confined. Then, another packet of S_j arrives at most every $(dist(l_c) + 1) \leq (indeg + r - 1)^{l_c+1}$ time steps. Hence, we are looking for a minimal x such that $t' \leq (x+1) * ((indeg + r - 1)^{l_c+1})$ and x is the leak we looked for.

This can be seen more easily, if we consider the arrival rate of packets for time steps $\geq t'$. The rate is the reciprocal of the time between to packets i.e. $\frac{1}{(indeg+r-1)^{l_c+1}}$. So, the number x of packets leaking the confinement path would be

$$x \geq \lceil t' * \frac{1}{(indeg + r - 1)^{l_c+1}} \rceil$$

After substituting t' back to $t - (b-1) * l_c$ this gives us the leak with

$$leak(t) = x \geq \lceil \frac{t - l_c * (b-1)}{(indeg + r - 1)^{l_c+1}} \rceil$$

.

□

Note that this characterization of lossy confinement somewhat overrates the capabilities of the adversary. Besides, we pointed out it is already hard to implement an injection scheme which achieves full confinement. Lossy confinement with the capabilities of the theorem is even less feasible.

4.5. Bounds on Injection Classes

4.5.1. A Bound on Class $\mathcal{C}_1$

As shown before most instability examples share that B_0 is some initial set and hereafter $B_i = S_{i-1}$. This was defined as the class $\mathcal{C}_2$ in Definition 4. The general worst case of any such instability form can be modeled by the topology found in Figure A.2 in the appendix. So, let us assume some arbitrary number n of possible parallel injection paths. Usually, these paths are of finite length, lets call this length l.

Given this, and in the light of Lemma 8 we can compute a bound on the queue build-up of a series of injections into E_{inject} which accumulate in the queue of edge E_t. Fully confining each set S_i on its path towards E_t gives the adversary a maximum time of $(b-1) * l$ time steps. After this time is over at least the first set will begin arriving at E_t and hence the size of some sets will decrease. If the adversary wants to prevent this loss the maximum full confinement time gives us the maximum injection time and without further examination the overall bound will be $[(b-1) * l] * r$.

As one might have guessed already, this is not the full truth. To be exact, we need to consider what happens after the first sets begin arriving at E_t and when the adversary will continue to inject packets into E_{inject}. This brings in lossy confinement for a growing number of sets S_i. As long as their arrival rate at E_t is below r, the rate of new injections, the adversary will be able to inject further and further packets[6]. At first sight this needs to be approached as an optimization problem: decide when the adversary needs to stop and release his injections. Even so, a bound for this case can be derived much easier.

Theorem 4 (Bound on $\mathcal{C}_1$)**.** *The burst size of injection schemes in $\mathcal{C}_1$ is bounded by $|I| * \frac{1}{1-r}$ where I is the largest of all initial sets.*

We may also consider the overall largest possible set size for I i.e. $|I| = b$. Then the burst size at any time is $\leq b * \frac{1}{1-r}$.

Proof. Let $l \in \mathbb{N}$ arbitrary. Besides, suppose also $n \in \mathbb{N}$ arbitrary.

Now, each set can be fully confined as long as needed to inject as many paths as available. For this case, we need to focus on the blocking technique as defined in the

[6] In fact, if we had assumed `EFF` for this class this would not be possible: W as defined in the proof of Lemma 5 would grow as packets from only some sets will already arrive.

class definition. Suppose we start with an initial set I queued in $E_{injection}$ whose remaining path is only to traverse this edge[7]. Then the first set S_0 may be injected behind I (i.e. $B_0 = I$) and will be of size $|S_0| = I * r$. The next set will use the set injected before as a block e.g. S_1 uses S_0 as a block and will be of size $|S_1| = S_0 * r$. Hence, in the worst case for $l, n \Rightarrow \infty$ we will have the following series:

$$I * r + (I * r) * r + (I * r * r) * r + \ldots = \sum_{i=1}^{\infty} I * r^i = I * \sum_{i=1}^{\infty} r^i$$

This is the geometric series which is bounded due to $r < 1$ with the sum $I * \sum_{i=1}^{\infty} r^i = I * \frac{1}{1-r}$. □

Another interesting aspect of this class is the relation of instability and the injection rate r. As described before, adversarial instability for FIFO networks does occur for injection rates $r = 0 + \epsilon$ even with infinitesimal small $\epsilon > 0$. However this class does not show this behavior which can be easily derived from the bound above. In order to yield instability the queue build-up needs to be strictly greater than I itself. But to consider this case we do not only need to consider a bound on the burst size but also the traversals over E_t while the burst arrives at this edge.

For the worst case assume E_t has sufficient input edges as in Figure A.2. So, all sets arrive perfectly aligned at the same time fulfilling Lemma 5. This means that still at least $r * I$ time steps are needed for the first set to arrive. In order to yield instability this means that $I * \sum_{i=1}^{\infty} r^i - r * I > I \Rightarrow \frac{1}{1-r} - r > 1 \Rightarrow 1 > (1+r)(1-r) \Rightarrow r^2 > 2$ Finally, this means that not only the burst size is bound but also a lower bound on the injection rate exists where all injection schemes of this class do not overload networks.

The class comprises many previously published instability examples. However, some use topologies which are different from Figure A.2. For instance, [33] introduced the idea of gadgets and a sophisticated injection and confinement scheme which is quite near to the bound on the injection rate as they require $r = \frac{1}{2} + \epsilon$ with $\epsilon > 0$. Still, also for these examples this bound is valid as they belong to the class presented above; the figure in the appendix was only to ease perception.

Besides, this simple example already shows the general approach taken to find bounds on the worst case burst size of injection schemes.

4.5.2. A Bound on Class $\mathcal{C}_2$

To derive the bound on the burst size of injection schemes in the second class we first estimate how long old packets can stay in the system and then how long the remainder can be used as blocks for injections. In particular this gives more meaning to the parameter k in the definition of $\mathcal{C}_2$.

[7] It may have a longer path but must not reach E_t due to the simple-path model and thus there is no loss of generality by this assumption.

Definition 8 (M)**.** *M is defined as the number of time steps until all initial packets present at the begin of one round (Definition 3) have left the network.*

The way to think of initial packets is that we fix the time step i_b as the beginning of one round. The packets which are in the system exactly in this time step are then called initial packets.

As seen before blocks rely on already enqueued packets. Hence, for the worst case we assume an initial set which is queued in the first edge on the longest path in the network. Its packets are in the worst case destined for the last edge on the longest path and the set size might be the largest of all initial sets. We call this initial set I_0 for that reason.

To find the time when I_0 completely left the system, we assume that the adversary uses the whole longest path as a confinement path to delay this as long as possible. The previous characterization of confinement in Theorem 3 proves that the time until any set may leave the network is linearly dependent on the length of the confinement path. Because of this, by the time I_0 vanished, all other initial packets will have left the network, too, as I_0 traversed the longest path in the network. Thus, M is exactly this time.

For M time steps initial sets can be used as blocks for injections. After this period, the newly injected sets may be used as blocks for subsequent injections i.e. $\exists k$ such that $\forall j > k$ S_j now works as a block for the injection of S_{j+1}.

So, as a first step we quantify M.

Lemma 10 (Bound on M)**.**

$$M \leq |I_0| * indeg^{l+1} + l * b$$

Where I_0 is the size of the largest of all initial sets.

Proof. Suppose that I_0 is the largest of all initial sets, queued in the first edge on the longest path in the network[8]. To estimate the total time until I_0 vanished two phases pass. The first phase is the time until the full confinement of I_0 is over: $l * (b - 1)$, so the first packet of I_0 reaches the first place in the queue of the last edge on the longest path. The second phase is the time until even lossy confinement is over i.e. the time until the last packet of I_0 left the system.

As already said, as we assumed I_0 to be the largest of all initial set, left with the largest possible path a bound on the time I_0 can spend in the networks yields a bound on M. Hence, we directly apply Theorem 3 to M.

The leak at time M should be the whole set I_0 except the first packet which already arrived after the time of full confinement was over. So, $leak(M) = |I_0| - 1$. According to Theorem 3 it follows:

[8] or one of them if there exist more than one

$$
\begin{aligned}
leak(M) \geq \quad & \lceil \frac{M - l * (b-1)}{(indeg + r - 1)^{l+1}} \rceil \\
|I_0| - 1 \geq \quad & \frac{M - l * (b-1)}{(indeg + r - 1)^{l+1}} \\
M &\leq (|I_0| - 1) * (indeg + r - 1)^{l+1} + l * (b-1) \\
&\leq |I_0| * indeg^{l+1} + l * b
\end{aligned}
$$

□

Again we can assume the case of a completely filled buffer at the first edge of the longest path i.e. $|I_0| = b$. This simplifies this form to $M \leq b * (1 + indeg^{l+1})$

Theorem 5 (Bound on $\mathcal{C}_2$)**.** *The burst size of injection schemes in $\mathcal{C}_2$ is*

$$\leq (|I_0| * indeg^{l+1} + l * b) * \frac{r}{1-r}$$

if $|I_0|$ is the set size of the largest of all initial sets
or the burst size is

$$\leq b * (1 + indeg^{l+1}) * \frac{r}{1-r}$$

if $|I_0| = b$
(If a set of buffer size is queued in the first edge of the longest path in the network at i_b.)

Proof. Suppose that I_0 is the largest of all initial sets, queued in the first edge of the longest path in the network. As an example topology in Figure A.4 in Appendix A shows, there might be paths from each edge of the longest path to E_t and each time I_0 is queued in some edge this situation can be used for injections along these paths. Lets assume the longest path of length l and so the adversary can inject l sets in the time steps $[0, M]$. Then, after I_0 vanished, define $k := l$ and the subsequent sets $S_j, j > k$ may be injected behind each other satisfying the definition of class $\mathcal{C}_2$. For the worst case let us assume that injections in the period (M, ∞) take place as a geometric series of the form $(M * r) * r + (M * r * r) * r + \ldots = \sum_{i=2}^{\infty} M * r^i$ as the adversary already injected $M * r$ packets in $[0, M]$.

In Lemma 10 we derived a bound on M and the geometric series is bound due to $r < 1$. So the overall burst size is less than:

$$M * \sum_{i=1}^{\infty} r^i = (|I_0| * indeg^{l+1} + l * b) * \frac{1}{1-r}$$

For the second statement set $|I_0| = b$ and use the comment below Lemma 10.

□

In contrast to the class $\mathcal{C}_1$, injection schemes of the second class can cause queue build-up at any rate. The topology and injection schemes presented by Bhattacharjee et al. in [4] proves that and is an example for the bound presented here.

4.5.3. Approaching Other Classes

The bound on class $\mathcal{C}_2$ is quite a general one and under further assumptions might be complete. Without further assumptions it is not easy to show that after all initial sets have left the network system the adversary must resort to a $\mathcal{C}_1$ injection scheme finally bounding the burst size. This is why we present two examples from $\mathcal{C}_3$ which violate the previous consideration. However, the bound is estimated very roughly and may be still valid[9] even for cases where its proof is already wrong. We do neither describe nor prove the instability examples, these are only meant to contradict the bound's idea and show that the class structure is justified introducing this new class $\mathcal{C}_3$.

The first example uses a separate queue build-up gadget so that after the initial sets left the system a new blocking set can be produced. It is depicted in Figure A.5 in Appendix A. A second example is presented here. This one is based on a different idea. What is the reason that the first class is so easily bound as a geometric series? Obviously, this is related to the fact that the next set which is used as a block is much smaller than the one before. That it is just strictly smaller is not enough, as it can be seen in the bound of $\mathcal{C}_2$: while the initial set traverses along its path it is spread out and for each injection S_i the block size is strictly smaller than for S_{i-1}.

With that thought in mind the idea is to use any of the already injected sets S_i as a block. But not in the form of a class $\mathcal{C}_1$-scheme, but in a moving form similar to that of the initial set in $\mathcal{C}_2$-schemes.

Assume the overall network topology as shown in Figure A.4 in Appendix A. This one was used for $\mathcal{C}_2$ schemes. After M time steps the initial packets vanished from the system. Now the sets S_i are further confined and travel along their confinement paths, the dashed edges from top to down.

Imagine a sector of such a path: $(c_j c_{j+1} c_{j+2} c_{j+3} c_{j+4})$. Figure 4.3 show a sketch where c_i are the previously dashed edges, the confinement path. We now assume a topology where each of the nodes of this path has additional edges creating additional paths to E_t. These additional paths are denoted with p_x in Figure 4.3. While some set S_i travels along their confinement path they are confined and some packets queue in the edges c_x. Each of these queues can be used as a block for a new injection along the newly introduced paths. After the last packet of S_i arrived, for instance, at edge c_j and is queued together with other packets, no further injections into c_j are sensible to confine S_i further. Instead, the adversary uses this filled queue as a

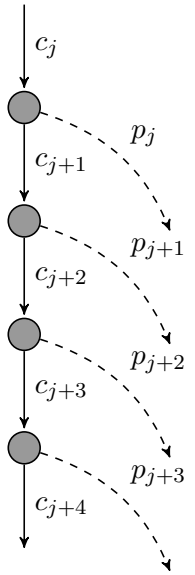

Figure 4.3.: Extract Confinement Path

[9] Or the bound is only conquered by tremendous topological structures

block for the injection of some set S_{k+1}[10] together with the path $(c_j p_j E_t)$. Next, after the last packet of S_i left the queue of edge c_j the adversary may perform the same on c_{j+1} and p_{j+1}.

This scheme is bound as easily as the ones before. The series of injections after the initial packets vanished does not follow a geometric series. Instead, we again need to fall back to the optimization treatment already mentioned above. While some sets S_i already arrive at E_t new injections can take place as long as the combined leak rate is less than the injection rate of new packets.

According to Theorem 3 the gap between packets is at most $(indeg + r - 1)^{l+1}$. Therefore we would need the case that h sets already arrive at E_t such that their rate $h * \frac{1}{(indeg+r-1)^{l+1}} \geq r$. After this time step further injections are counterproductive and a bound may be found. Although, for any specific case the solution is not hard to find, the general form does not yield any insights.

Note that this also holds under the assumptions `MSS`, `MERGE`, and `EFF` from the previous chapter. Especially without the first two assumptions various other schemes are thinkable leading to this class $\mathcal{C}_3$. The other remaining class $\mathcal{C}_\infty$ i.e. injection schemes that are not efficient comprises another plethora of injection schemes. Nevertheless, these are by definition not efficient and shall not be of concern from a worst case perspective.

[10] We use $k + 1$ here as we already defined it to be the set number after M time steps (all initial packet vanished). This was defined in the proofs for the bound of class $\mathcal{C}_2$

5. Conclusion

5.1. Summary

We started this work with a discussion of previous stability results in the adversarial queueing framework. We found that although much work has been done on stability and its favorable properties have been shown, stable conditions are an implausible assumption for real-world networks on the long term. For this reason we concluded the importance of characterizing instability and to research possible effects.

From simple queue build-up examples we first draw explanation on why queue build-up happens in spite of load conditions. Consequently, the introduction of three assumptions to decrease the freedom of adversaries lead to the notion of efficient injection schemes which comprised a description of the three components: blocks, injections, and confinement.

Although our proofs of characteristic instability properties were based on all three assumptions, we were able to classify a broader class of previously published examples of network instability. We successfully reduced this preliminary classification to the class hierarchy of $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, and the remainder class $\mathcal{C}_\infty$.

To study the effect of instability on real-world networks, several assumptions of the original adversarial queueing model need to be removed. As infinite buffers apparently allow for infinite queue lengths, we focused on this assumption and examined instability effects in a finite model. We found that even with non-bursty requests below network capacity in an instable network buffers can fill up fast. However, once some particular buffers are full, instability effects do not worsen and the periodic growth of the bursts comes to halt.

As instability effects are determined by the burst size they may provoke, we seek to quantify these effects. To enable some preliminary results in this domain we first studied how one central aspect of instability incidents, confinement, behaves in the finite model. Under the assumption of a worst case confinement efficiency we subsequently proved bounds on the burst sizes possible in $\mathcal{C}_1$ and $\mathcal{C}_2$. In the first class the burst size proved to be dependent on the initial set size (in the worst case the buffer size) only. The second class needed more sophisticated methods, and so, the burst size was expressed dependent on the maximum in-degree, the longest path, and again the uniform buffer size in the network. Finally, we introduced two examples from the third class $\mathcal{C}_3$ which contradicted the $\mathcal{C}_2$-bound assumptions showing that a different approach shall be needed here and a closed form bound on $\mathcal{C}_3$ is left as an open question.

Overall, our contribution is three-fold. Firstly, to our knowledge, we give the first characterization of instability and some of its factors. Secondly, we introduce a prac-

tical class hierarchy in which each class has unique topological requirements. The proposal allows the classification of all known FIFO injection schemes up to today. Thirdly, we assessed the infinity problem and brought adversarial instability effects into a domain where they may be discussed with numerical values. Nevertheless, we must admit that the bounds presented for this are rough and only a preliminary step.

5.2. Open Questions and Future Work

Further open questions occurred throughout this work. We distinguish two possible directions for future work for which we gave starting points.

The first is further theoretic study of adversarial instability. Our characterization was focused on instability in a general topology case. That is, our efficient injection schemes, the class hierarchy, and finally all the bound considerations all treat topology as if the adversary can chose its actual implementation only restricted by few parameters. This helped to learn about instability and to describe some basic topological requirements of each class. Nevertheless, this is infeasible for a practical purpose: in real-life we are confronted with a given topology which is to be analyzed. So the missing part is an algorithm which gives the worst case adversary strategy together with a bound on its burst size for a given topology.

Another important contribution would be to review component based instability. Although used already earlier, Weinard was the first to prove that a network is FIFO stable if and only if all strongly connected components are stable [40, Lemma 1]. However, it is an infeasible assumption that the Internet as a interconnected cluster of various autonomous systems would draw conclusions and take action against any risk as a whole. Thus, it is inevitable to investigate a per-component based risk if no or weak assumptions are made on arrival patterns of the component's ingresses. Several approaches such as extended ingress scheduling policies or topological burst collectors are conceivable.

The second future research direction must be experimental. Although eventually targetting "real-world" networks, we never conducted any experiments proving the practicability of our results. The same holds for results from the adversarial queueing community as a whole. The only known simulation has been [14] when Chroni et al. implemented a simulation of "network constructions, the adversarial strategies and the properties of contention-resolution protocols" to study "the behavior of the number of packets of all the network queues in successive phases for various compositions of protocols". Nonetheless, their approach does not allow practical conclusions as they only copied the adversarial framework. Exact timing or routing issues as well as cross talk with end-to-end congestion control may arise in real-world networks possibly distorting adversarial effects. Furthermore, in the Internet congestion control algorithms as well as routing protocols will interfere with the adversarial model. Thus only simulations or experiments that integrate the multi-layer and multi-protocol nature of modern networks can yield meaningful results.

Bibliography

[1] Aggarwal, A., Savage, S., & Anderson, T. (2000). Understanding the performance of TCP pacing. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3 (pp. 1157–1165).: IEEE.

[2] Alvarez, C., Blesa, M., & Serna, M. (2004). A Characterization of Universal Stability in the Adversarial Queuing Model. *SIAM Journal on Computing*, 34(1), 41.

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

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

[5] Blesa, M. (2004). On the use of topology extensions for provoking instability in communication networks. In *Information Society , subconference on Theoretical Conputer Science (IS04-TCS)*, number October 11-15 (pp. 197–199).

[6] Blesa, M. (2005a). Deciding stability in packet-switched FIFO networks under the adversarial queuing model in polynomial time. *Distributed Computing*, 33116, 429–441.

[7] Blesa, M. J. (2005b). *Stability in communication networks under adversarial models*. PhD thesis, Universitat Politcnica de Catalunya.

[8] Borodin, A., Kleinberg, J., Raghavan, P., Sudan, M., & Williamson, D. P. (1996). Adversarial queueing theory. STOC '96 (pp. 376–385). Philadelphia, Pennsylvania, United States: ACM.

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

[10] Bramson, M. (1994). Instability of FIFO queueing networks. *The Annals of Applied Probability*, 4(2), 414–431.

[11] Bramson, M. (2008). *Stability of queueing networks*, volume 5.

[12] Chlamtac, I., Farago, a., & Fumagalli, a. (1998). A deterministic approach to the end-to-end analysis of packet flows in connection-oriented networks. *IEEE/ACM Transactions on Networking*, 6(4), 422–431.

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

[14] Chroni, M., Koukopoulos, D., & Nikolopoulos, S. D. (2007). An experimental study of stability in heterogeneous networks. WEA'07 (pp. 189–202). Rome, Italy: Springer-Verlag.

[15] Cruz, R. (1991). A calculus for network delay. I. Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1), 114–131.

[16] Denny, M. (2011). *Their Arrows Will Darken the Sun: The Evolution and Science of Ballistics*. JHU Press, illustrate edition.

[17] D'ıaz, J., Koukopoulos, D., Nikoletseas, S., Serna, M., Spirakis, P., & Thilikos, D. M. (2001). Stability and non-stability of the FIFO protocol. SPAA '01 (pp. 48–52). Crete Island, Greece: ACM.

[18] Erlang, A. (1909). The theory of probabilities and telephone conversations. *Nyt Tidsskrift for Matematik B*.

[19] Frank, H., Kahn, R. E., & Kleinrock, L. (1972). Computer communication network design: Experience with theory and practice. *Networks*, 2(2), 135–166.

[20] Gettys, J. & Nichols, K. (2012). Bufferbloat: Dark buffers in the internet. *Communications of the ACM*, Volume 55(Issue 1), 57–65.

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

[22] Jackson, J. (1957). Networks of waiting lines. *Operations Research*, 5(4), 518–521.

[23] Kelly, F. (1975). Networks of queues with customers of different types. *Journal of Applied Probability*, 12(3), 542–554.

[24] Kleinrock, L. (1964). *Message delay in communication nets with storage*. McGraw Hill, New York.

[25] Kleinrock, L. (2002). Creating a Mathematical Theory of Computer Networks. *Operations Research*, 50(1), 125–131.

[26] Koukopoulos, D. (2009). Stability in heterogeneous multimedia networks under adversarial attacks. *Journal of Universal Computer Science*, 14(2), 444–464.

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

[28] Koukopoulos, D., Mavronicolas, M., & Nikoletseas, S. (2002). On the stability of compositions of universally stable, greedy contention-resolution protocols. *Distributed*, 14186, 88–102.

[29] Koukopoulos, D., Mavronicolas, M., Nikoletseas, S., & Spirakis, P. (2005). The Impact of Network Structure on the Stability of Greedy Protocols. *Theory of Computing Systems*, 38(4), 425–460.

[30] Kumar, P. R. (1993). Re-entrant lines. *Queueing Systems*, 13(1-3), 87–110.

[31] Le Boudec, J. & Thiran, P. (2001). *Network calculus: a theory of deterministic queuing systems for the internet.* springer-Verlag.

[32] Leland, W. E., Taqqu, M. S., Willinger, W., & Wilson, D. V. (1993). On the self-similar nature of Ethernet traffic. SIGCOMM '93 (pp. 183–193). San Francisco, California, United States: ACM.

[33] Lotker, Z., Patt-Shamir, B., & Rosen, A. (2004). New Stability Results for Adversarial Queuing. *SIAM Journal on Computing*, 33(2), 286.

[34] Lu, S. & Kumar, P. P. (1991). Distributed scheduling based on due dates and buffer priorities. *Automatic Control, IEEE Transactions on*, 36(12), 1406–1416.

[35] Nain, P. (2002). Impact of bursty traffic on queues. *Statistical inference for stochastic processes*, 5(3), 307–320.

[36] Paxson, V. & Floyd, S. (1995). Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Trans. Netw.*, 3(3), 226–244.

[37] Rosen, A. (2002). A note on models for non-probabilistic analysis of packet switching networks. *Information Processing Letters*, (3), 0–5.

[38] Seidman, T. (1994). ' First Come , First Served ' can be unstable ! *Automatic Control, IEEE Transactions on*, 39(10), 2166–2171.

[39] Starobinski, D., Karpovsky, M., & Zakrevski, L. (2003). Application of network calculus to general topologies using turn-prohibition. *IEEE/ACM Transactions on Networking (TON)*, 11(3), 411–421.

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

[41] Zdarsky, F. a., Robitzsch, S., & Banchs, A. (2011). Security analysis of wireless mesh backhauls for mobile networks. *Journal of Network and Computer Applications*, 34(2), 432–442.

A. Topology Examples for Instability Classes

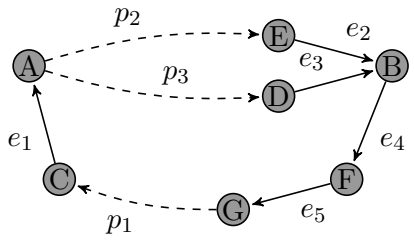

Figure A.1.: Weinard's simple-path unstable graph $\mathcal{A}^+$. An interrupted line indicates a simple path of an arbitrary number of edges.

In this appendix we will discuss examples for all three classes of the classification in the second chapter of this work. For minimal examples for the classes $\mathcal{C}_1$ and $\mathcal{C}_2$ we will use the graph minors by Weinard. Additionally we will give typical, parameterized structures for all three classes. The graph minors introduced by Weinard in [40] come in two groups: a family of unstable graphs $\mathcal{A}, \mathcal{B}, \mathcal{C}$, and a family of simple-path unstable graphs: $\mathcal{A}^+, \mathcal{B}^+, \mathcal{C}^+$. For real-world networks it is sensible to assume simple-path injection schemes, as we already did in the introduction of the adversarial model. Hence, for the following considerations we will consider the second family.

A.0.1. Examples for Class $\mathcal{C}_1$

A good example of minimal topological properties of the first class is the first graph minor $A^+$of Weinard. It is shown in Figure A.1. Two paths (of arbitrary length) connect one corner-edge (A) to the opposite edge (B) giving the adversary the possibility to inject to sets after each other which, due to confinement, reach B at the same time.

The graph minor form is expandable in one dimension. This allows for more efficient confinement but not for the injection of more parallel sets. An example of a parameterized and more efficient topology for this class is shown in Figure A.2. Sets S_i are injection into E_{inject} one after another and then travel along distinct paths towards E_t where confinement can take place.

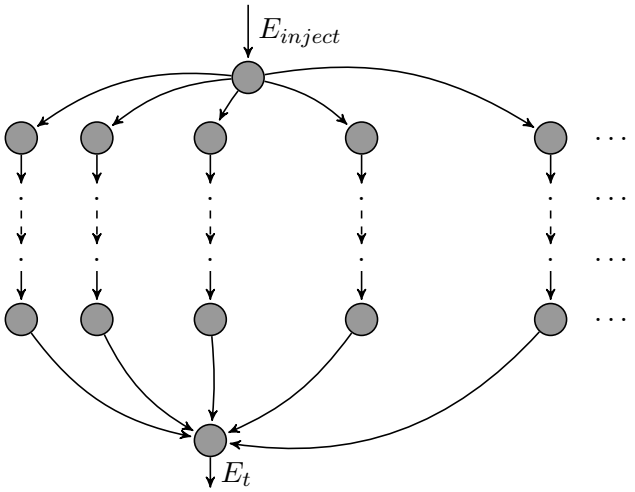

Figure A.2.: Example of an efficient topology for an injection scheme $\in \mathcal{C}_1$. Topologies of this class have been used quite often for instability examples. The dashed edges in the middle and the dots on the right should indicate further edges and paths.

A.0.2. Examples for Class $\mathcal{C}_2$

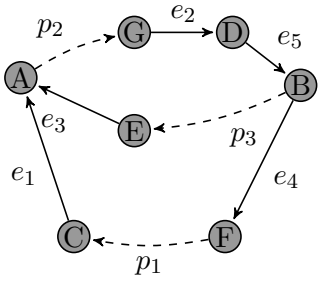

Figure A.3.: Weinard's simple-path unstable graph $\mathcal{B}^+$. An interrupted line indicates a simple path of an arbitrary number of edges.

Although the original instability proof does not use in the way to be an element of $\mathcal{C}_2$ the graph minor $\mathcal{B}^+$ is a minimal example to show what topological requirements are needed for injection schemes of this class. To describe this we will use an informal description of a matching injection scheme.

The Figure A.3 shows that the minor B^+ has two incoming edges $e3$,e_1. In contrast to the first minor, the two paths leading to e_3,e_1 do not start from the same node. Instead, a set S_0 might be injected into e_5 targeted at least at p_2, while S_1 is injected into e_4 which are not the same. This can be accomplished if a

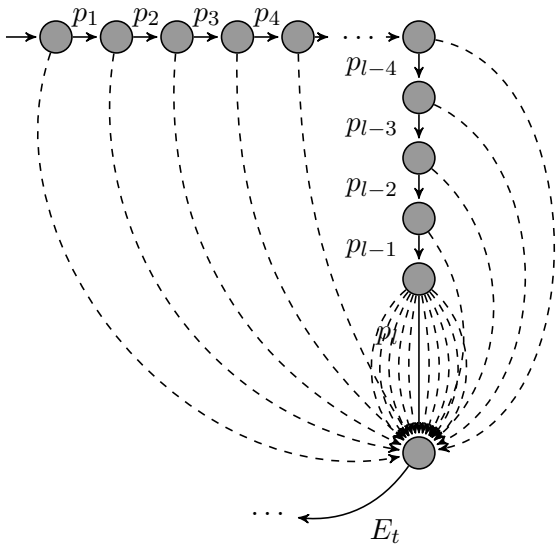

Figure A.4.: An example for a topological structure sufficient for an injection schemes $\in \mathcal{C}_2$. Furthermore, this topology is a good example to visualize the bound in Theorem 5.

temporary set would be injected behind an initial set queued in p_2 together with the path (e_2, e_5, e_4). Then S_0, S_1 could be injected behind this set and together with single-edge confinement a queue build-up can be derived. As this examples show it is possible to implement minimal $\mathcal{C}_1$ examples, although they only occurred for larger parameterized topologies up to now.

It is not sensible to consider injection schemes of this class in networks of this size and it never occurred on any small ones. For large structures schemes in $\mathcal{C}_2$ are very efficient as the example in Figure A.4 shows. In this topology an initial set I_0 is assumed to travel from p_1 to p_l while active confinement actions take place to slow it down. Every time it is queued in some edge p_i a set S_i may be injected behind it. S_i then follows its own path towards E_t where it can be confined until the other sets are injected. Finally, when the last packets of I_0 left the queue of p_{l-1} and some set S_{l-1} is queued up there, an injection scheme similar to class $\mathcal{C}_1$-examples takes place: further injections use the last set as a block. Overall this leads to the series $\sum_i r^i * M$ where M is the number of time steps until I_0 completely left the system, as depicted in Theorem 5.

A.0.3. Examples for Class $\mathcal{C}_3$

Examples for this class are necessarily more complicated than the others before. One reason is that no previously published instability examples of this class are known as to the time writing. Another is that, although this class allows for many implementations as it is a super class of both previous classes, we want to show it

is actually different.

For this purpose remember the parameterized injection scheme from the class before. At some edge p_i we adjoin a baseball graph such that one of it queue build-up edges (e_1) is part of the original initial-set path i.e. $p_i = e_1$. This can be seen in Figure A.5. Now, M time steps passed and all initial packets just left the system. Usually, as in the example of $\mathcal{C}_2$ before, now only a $\mathcal{C}_1$-type injection scheme can take place which is naturally bounded as shown in section 4.5.1. To avoid this, let us assume the adversary timed injections in the baseball graph in a way so that edge e_1/p_i is filled with a new blocking set. This set is targeted at p_l (for example), so that it travels along the old confinement path of the initial set I_0. Consequently, further injections can take place as it this new set works as a block for further injections into the paths which have not yet been used for injections.

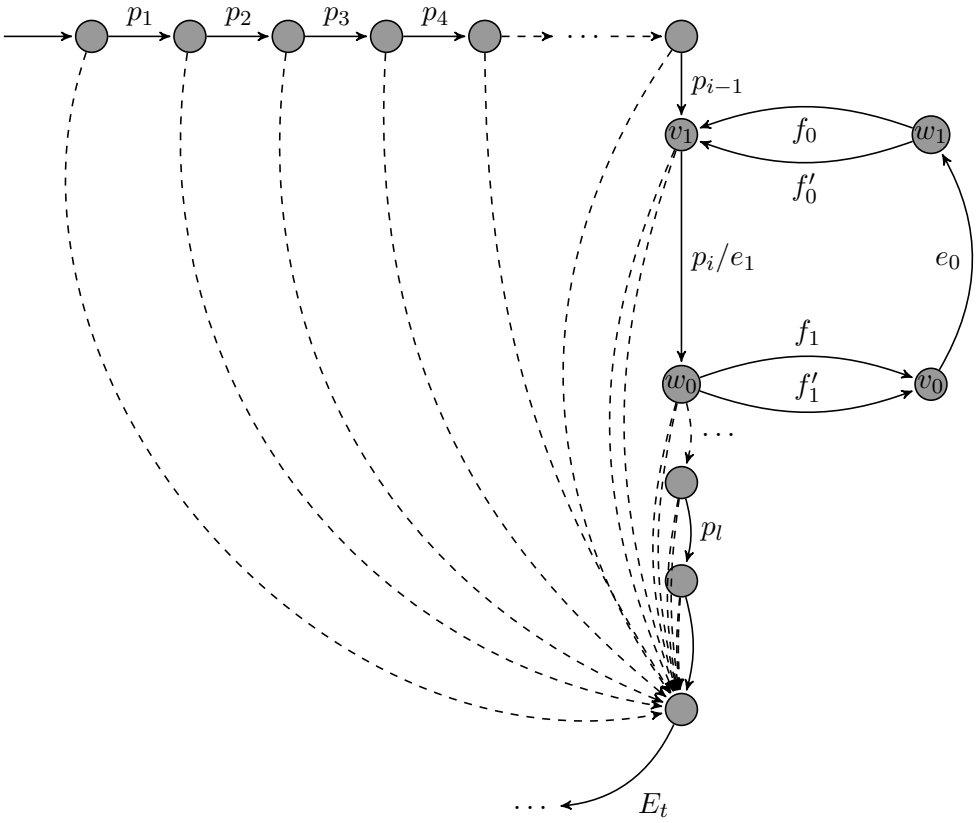

Figure A.5.: Extended topology to allow injection schemes which are not considered in any of $\mathcal{C}_1$,$\mathcal{C}_2$

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich durch die Angabe der Quelle, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht.

Kaiserslautern, den 30. Januar 2012 _____

Daniel Berger