Simple Routing Strategies for Adversarial Systems

Baruch Awerbuch*
Department of Computer Science
Johns Hopkins University
3400 N. Charles Street
Baltimore, MD 21218, USA
baruch@cs.jhu.edu

Petra Berenbrink†
Department of Computer Science
University of Warwick
Coventry,CV4 7AL, UK
petra@dcs.warwick.ac.uk

André Brinkmann‡
Heinz Nixdorf Institute and
Department of Electrical Engineering
University of Paderborn
33102 Paderborn, Germany
brinkman@hni.upb.de

Christian Scheideler§
Department of Computer Science
Johns Hopkins University
3400 N. Charles Street
Baltimore, MD 21218, USA
scheideler@cs.jhu.edu

May 11, 2002

Abstract

In this paper we consider the problem of delivering dynamically changing input streams in dynamically changing networks where both the topology and the input streams can change in an unpredictable way. In particular, we present two simple distributed balancing algorithms (one for packet injections and one for flow injections) and show that for the case of a single receiver these algorithms will always ensure that the number of packets in the system is bounded at any time step, even for an injection process that completely saturates the capacities of the available edges, and even if the network topology changes in a completely unpredictable way. We also show that the maximum number of packets or flow that can be in the system at any time is best possible by providing an (essentially) matching lower bound that holds for any online algorithm that does not duplicate packets. Interestingly, our balancing algorithms do not only behave well in a completely adversarial setting. We show that also in the other extreme of a static network and a static injection pattern the algorithms will converge to a point in which they achieve an average routing time that is close to the best possible average routing time that can be achieved by any strategy. This demonstrates that there are simple algorithms that can be efficient at the same time for very different communication scenarios. To have such algorithms will be of particular importance for communication in wireless mobile ad hoc networks (or short MANETs), in which at some time the connections between mobile nodes and/or the rates of input streams may change quickly and unpredictably and at some other time may be quasi static.

*Supported by DARPA grant F306020020550 "A Cost Benefit Approach to Fault Tolerant Communication" and DARPA grant F30602000-2-0526 "High Performance, Robust and Secure Group Communication for Dynamic Coalitions".

†Supported by EPSRC grant GR/M96940 "Sharper Analysis of Randomised Algorithms: a Computational Approach" and the IST Programme of the EU under contract numbers IST-1999-14186 (ALCOM-FT) and IST-1999-14036 (RAND-APX).

‡Supported in part by the DFG-Sonderforschungsbereich 376 "Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen".

§Contact author

1 Introduction

This paper considers the problem of designing distributed algorithms for the delivery of dynamically changing input streams of packets in a dynamically changing network, where both the topology and the input streams change unpredictably and are under adversarial control. Such a network model makes sense, for example, in the context of a wireless mobile ad hoc network (or short MANET) where links (connections) between mobile nodes change quickly and unpredictably.

This paper focuses on the special case of a single receiver. If the topology and the input streams were fixed, this would have translated into a distributed max-flow problem. The dynamic nature of the problem poses new and quite formidable challenges.

First and foremost, the topological changes prevent us from using the augmenting path based methods that were originally introduced by Ford and Fulkerson [11] and subsequently generalized for the distributed setting by Goldberg [15] and Goldberg and Tarjan [16]. That is, if one attempts to run these algorithms, they will fail, in the worst case, to deliver any packets to the destination.

Only "truly" distributed algorithms that work in an augmenting-paths-free way have a chance to succeed in such manner. This fact motivated Awerbuch, Mansour and Shavit [8] to introduce local load balancing as the major building block for dynamic network flow algorithms, and all the subsequent work on dynamic networks is built around this concept. It was further studied and improved in [1, 2, 3] in the context of a single sender-receiver pair, and then generalized by Awerbuch and Leighton [6, 7] to multiple senders and receivers. However, the results [6, 7] are only claimed to hold for known, fixed input rates that are below the maximum possible injection rate. The most recent work in this area has been done by Aiello et al. [4] and Gamarnik [13]. Aiello et al. show that as long as the adversary injects packets into the system so that on average every edge has to carry a load of at most $1-\epsilon$ per time step for some $\epsilon > 0$, a simple, distributed routing strategy based on balancing manages to keep the number of packets that are in transit bounded at any time. Gamarnik only presents a centralized algorithm, but for a more general injection model.

The results by Awerbuch and Leighton [7] and Aiello et al. [4] do not only hold for multiple receivers in static networks but also for dynamic networks under the restriction that for any sufficiently large interval I, the number of time steps an edge e must be working must be above the number of packets injected in I that have paths (selected by the adversary) that contain e. In this paper, we will analyze strategies in a dynamic network model that does not need this restriction and therefore seems to be more general.

It is important to distinguish the adversarial routing setting above from a much more restrictive setting of *adversarial queueing* in which an adversary injects packets into the system, and it has to reveal their paths to the system. There is no need to do either routing or admission (input stream) control. Thus, it only remains to find the right switching strategy to send the packets to their destinations. This model was introduced by Borodin et al. in [9] and has subsequently been studied in several papers [5, 12, 13, 14, 17, 18].

All previous results mentioned above (except for a result by Gamarnik for the special case that the adversary reveals the paths to the system [13]) only manage to accomplish their task if the injection process is strictly below what can be handled by the network. Or more precisely, it has to be ensured that every edge of the network on average only has to carry a load of at most $1-\epsilon$ per time step (using a best possible strategy) for some fixed $\epsilon > 0$. The bounds shown on the maximum number of packets that are in the system at any time go to infinity as ϵ approaches 0.

In contrast, this paper shows that even if the paths for the packets are not given to the system and even if the network changes are completely unpredictable, it is possible to guarantee an upper bound on the number of packets that can be in the system at any time even if the injection process is exactly at the limit of what the network can handle. Interestingly, our online algorithms that achieve this result are not only optimal in the completely adversarial setting but can also be used for almost optimal routing in the situation where the network is static and the input streams have a fixed rate. Furthermore, they have the important property that they do not "oscillate" in this case but monotonically reach the maximum throughput.

1.1 Models

In this section we describe our network models and injection models in more detail. We distinguish between two network models (adversarial and static networks) and four injection models (adversarial injections and static injection patterns, both for the packet and the flow injection model). We assume that every node has an arbitrarily large buffer to store packets or flow, and that the edge capacity is one. Furthermore, we will only consider directed edges. This does not exclude the undirected edge case, since an undirected edge can be viewed as consisting of two directed edges, one in each direction.

The adversarial network model. Motivated by the unpredictable behavior of mobile ad-hoc networks, we introduce the following adversarial network model. Suppose we have a set of n nodes. We assume that the way these nodes are connected can change over time and is completely under the control of an adversary. That is, only the connections specified by the adversary are working and all others are not working.

We distinguish between two adversarial injection models, one for packet injections and one for flow injections. In the case of the *adversarial packet injection model*, the adversary also controls the injection of packets into the system. Of course, the adversary has to be restricted in order to ensure that the number of packets in transit can be kept finite. We assume that for each injected packet the adversary has to specify a schedule. A *schedule* $S = ((e_0, t_0), (e_1, t_1), \ldots, (e_\ell, t_\ell))$ consists of a sequence of movements by which the injected packet P can be sent from its source node to its destination node. For this the adversary must ensure that

- P is injected at time step t_0, which is represented by an *injection edge* e_0 leading from some imaginary node $\times$ to the source of P,
- the edges $e_1, \ldots, e_\ell$ form a connected path, with the starting point of e_1 being the source of P and endpoint of e_ℓ being the destination of P,
- the time steps have the ordering $t_0 < t_1 < \ldots < t_\ell$, and
- edge e_i is active at time t_i for all $1 \leq i \leq \ell$.

Another necessary requirement is that two schedules are not allowed to *intersect*, that is, they are not allowed to have a pair (e, t) in common. This ensures that a schedule represents a valid routing strategy for a packet.

There are no further restrictions. Thus, the adversary is free to activate any set of non-schedule edges at any time in addition to the schedule edges. Also, we do not require any restriction on the time it takes to complete a schedule. Hence, a schedule could also take forever without changing any of the results below. Certainly, there should only be a finite number of schedules of this kind to ensure that the number of packets in transit can be kept finite at any time.

A schedule $S = ((e_0, t_0), (e_1, t_1), \ldots, (e_\ell, t_\ell))$ is called *active* at time t for every t with $t_0 \leq t \leq t_\ell$. Besides the model it is important to specify parameters that allow to prove reasonable results within the model. First we demonstrate that it is not possible to bound the number of packets in the system in terms of the number of currently active schedules and any collection of network parameters.

Proposition 1.1 *Even if the adversary is oblivious (i.e. does not see the distribution of packets in the network), the number of packets in the system at some time step cannot be bounded by the number of currently active schedules or any network parameter.*

Proof. Suppose that we inject s schedules into node 1. Then we offer s edges to node 2. No matter what kind of routing algorithm we use, either node 1 or node 2 will have (in the expected case or with certainty, depending on the algorithm) at least $s/2$ packets afterwards. Suppose that this is node 1. (The arguments for the case of node 2 are similar.) Then we simply choose the adversarial strategy of using s schedules of the form $((\times, 1), t_0), ((1, 2), t_1), ((2, 3), t_2)$, where 3 is the destination node. This causes the (expected) number of packets that are still in the system after all schedules have been completed to be at least $s/2$, which certainly cannot be bounded by any network parameter or the number of currently active schedules at that time. $\square$

Hence, we will use a parameter $S_{\max}$ that gives an upper bound on the maximum number of schedules that can be active at any given time. Obviously, $S_{\max}$ is a lower bound for the number of packets that are in the system at any given time using any algorithm. For online algorithms, a much higher lower bound can be given:

Theorem 1.2 *For any online algorithm that does not duplicate packets there is an adversary, using only one node as destination for all packets, that causes the number of packets in the system to be at least* $S_{\max} \cdot \binom{n-1}{2} + S_{\max}$ *at some time.*

Proof. W.l.o.g. we restrict our attention to deterministic algorithms. This allows us to use adaptive adversaries instead of adversaries that have to specify schedules at injection times, which simplifies the proof. (The proof can be extended to randomized algorithms, since any set of choices that can be taken with positive probability within a finite time interval will eventually be taken by the algorithm.)

For all $i \in \{0, \ldots, n-1\}$ let the *height* h_i of node i be defined as the number of packets stored in it. Initially, $h_i = 0$ for all i. Let the *potential* of the system at any given time be defined as $\Phi = \sum_{i=0}^{n-1} h_i^2$. Node 0 will represent the destination.

The adversarial strategy works in rounds that are performed one after the other. Each round takes $\tau = 2S_{\max}+1$ time steps. Suppose that at the beginning of a round at some time step t_0 there are two non-destination nodes v and w with $h_{v,t_0} \geq h_{w,t_0}$ and $h_{v,t_0} < h_{w,t_0} + S_{\max}$. If no such pair exists, then the amount of packets in the system must be at least $S_{\max}\binom{n-1}{2}$. Injecting $S_{\max}$ more packets at some node then results in the lower bound. Otherwise, the adversary first injects $S_{\max}$ packets one after the other at node v. Then it activates $S_{\max}$ times the edge (v, w). Suppose that this causes $i \in \{0, \ldots, S_{\max}\}$ packets to move from v to w. We distinguish between two cases for i.

If $i \leq S_{\max} - 1$ then the adversary actives $S_{\max}$ times the edge $(w, 0)$. This results in $S_{\max}$ valid schedules S_i of the form $(((\times, v), t_0), ((v, w), t_0 + i), ((w, 0), t_0 + S_{\max} + i))$. Afterwards we have that $h_{v,t_0+\tau} \geq h_{v,t_0} + \delta$ for some $\delta \geq 1$ and $h_{v,t_0+\tau} + h_{w,t_0+\tau} \geq h_{v,t} + h_{w,t}$. Hence,

$$\begin{aligned} h_{v,t_0+\tau}^2 + h_{w,t_0+\tau}^2 &\geq (h_{v,t_0} + \delta)^2 + (h_{w,t_0} - \delta)^2 \\ &= h_{v,t_0}^2 + h_{w,t_0}^2 + 2\delta(h_{v,t_0} - h_{w,t_0} + 1) > h_{v,t_0}^2 + h_{w,t_0}^2 \ . \end{aligned}$$

Thus, the adversary will manage to increase the potential in this case.

For the remaining case $i = S_{\max}$ the adversary activates $S_{\max}$ times the edge $(v, 0)$. This results in $S_{\max}$ valid schedules S_i of the form $(((\times, v), t_0), ((v, 0), t_0 + S_{\max} + i))$. Afterwards we have that $h_{w,t_0+\tau} \geq h_{v,t_0} + \delta$ for some $\delta \geq 1$. Hence, we can reduce this case to case 1 with $i = S_{\max} - \delta \leq S_{\max} - 1$ by exchanging the roles of v and w. Thus, also here the adversary manages to increase the potential of the system.

Therefore, the only way to prevent the potential (and therefore the number of packets) from increasing to infinity is to arrive at the situation where there are no two non-destination nodes v and w with $h_{v,t_0} \geq h_{w,t_0}$ and $h_{v,t_0} < h_{w,t_0} + S_{\max}$. $\square$

Our second adversarial injection model, the *adversarial flow injection model*, is closely related to the above described model. We assume that flows of arbitrary positive values are injected into the network. Each flow comes with a schedule describing a path system that can be used to route the flow to its destination. Of course, the flow may be partitioned into several streams that are routed along different paths and that use different time steps to cross edges. Hence, a schedule for a flow f with value c may decompose into k schedules along simple paths for some number k:

$$((c^1, e_0^1, t_0^1), (c^1, e_1^1, t_1^1), \ldots, (c^1, e_{\ell_1}^1, t_{\ell_1}^1)), \cdots,$$

$$((c^k, e_0^k, t_0^k), (c^k, e_1^k, t_1^k), \ldots, (c^k, e_{\ell_k}^k, t_{\ell_k}^k)).$$

c^i is the *weight* (i.e. the amount of flow) of the ith schedule and $c = c^1 + c^2 + \cdots, +c^k$. Again, we have to demand from the adversary that all edges used in a schedule have to be active for that point of time. We assume that each edge can route a total flow of one per edge and that every node can store an arbitrarily large amount of flow. Hence, the schedules have to fulfill the property that the sum of flow values routed over the same edge at the same time has to be at most one. In the flow setting, we define $S_{\max}$ as the sum of the weights of all flow schedules along simple paths that are active at any time.

Additionally, we consider two non-adversarial injection models, again in a flow injection and a packet injection version. In our *static pattern packet injection* model we assume that a fixed set of source-destination pairs is injected in every time step. Of course, the adversary has to provide schedules that enables every generated packet to be sent to its destination. In the *static pattern flow injection* model, we have a fixed set of source-destination pairs with fixed flow values that is injected in every step. Also here, valid schedules have to be provided by the adversary.

The static network model. Here, we simply assume that the network is static, i.e, all specified links are working all the time.

1.2 New results

In order to analyze the performance of our algorithms, we will use the parameters $S_{\max}$ and B, where B denotes the maximum number of non-injection edges leaving (resp. leading to) a single node that can be active at any time. Since our results allow multiple edges connecting the same pair of nodes, they also hold for networks that use non-uniform edge capacities. However, in the case of non-uniform capacities, B has to be defined as the maximum sum of the capacities of the edges leaving (resp. leading to) a single node at any time.

A protocol for our injection models is called *bounded* for some $S_{\max}$ and B if it ensures that for any adversary respecting $S_{\max}$ and B there is an upper bound on the number of packets (depending only on $S_{\max}$, B, and the network size) that can be in the system at any time. Note that this notion is more strict than the stability notion used by some other papers in the adversarial setting, since it requires to have a strict upper bound on the number of packets in the system (which has usually not been demanded for randomized protocols). As our main result we show that there are simple balancing algorithms that are bounded for both packets and flows, even when both the network and injection process is adversarial and the injection process is at the limit of what the network can handle. Even more, our upper bounds on the number of packets and flows (see Theorem 3.2) that are in the system at any time essentially match the lower bound given by Theorem 1.2 .

Our algorithms are also bounded in the setting of (w, λ)-bounded adversaries introduced by Borodin et al. in [9]. For any w and $\lambda > 0$, we call an adversary a *discrete* (w, λ)*-bounded adversary* if it selects paths for the injected packets so that for all edges e and all time intervals I of length w, e is contained in no more than $\lambda \cdot w$ paths of packets injected during I. Analogously, we call an adversary a *fractional* (w, λ)*-bounded adversary* if it selects fractional path collections for its injected flows so that for all edges e and all time intervals I of length w, the total flow of the fractional paths injected in I that traverse e does not exceed $\lambda \cdot w$. Note that transferring stability to these models is not straight forward, since our model requires the existence of a bounded number of active schedules at any point of time. However, we show that our algorithms will guarantee that the number of packets (resp. the amount of flow) will always be polynomial in the network size and w even if $\lambda = 1$.

Next, we show that our algorithms also behave well if we have a static network and a static injection pattern. Our algorithm for the flow case converges against a fixpoint in which the amount of flow stored by any node does not change any more (Theorem 5.1). In Theorem 5.3 we show that in this fixpoint, when using LIFO (last in first out), the average delay of the flow achieved by our algorithm is very close to a best possible average delay, and hence almost as good as it can possibly be. Furthermore, our algorithm does not “oscillate”, because we can show that the flows in the nodes will monotonically increase and that the throughput monotonically reaches the rate of flow injected into the system. We also note how to transfer these results to the situation that packets have to be sent.

Our results demonstrate that simple balancing strategies are not only efficient in a completely adversarial scenario but also efficient in the case of static networks and injection patterns. Thus, there is hope that efficient and flexible communication strategies can be constructed that can even handle such "adversarial" networks like MANETs.

2 The Balancing Algorithms

In this section we will present simple, distributed balancing algorithms that are related to balancing algorithms studied by Awerbuch and Leighton [6] and Aiello et al. [4]. Both algorithms use parameters σ and $\Delta \geq 1$ that determine how aggressively the algorithms try to balance the load in the network. In the following, let $h_{v,t}$ denote the amount of packets or flow in node v at the beginning of time step t. The balancing algorithms will always assume that for the destination d, $h_{d,t} = -\infty$.

2.1 The discrete (σ, Δ)-balancing algorithm

This algorithm will be used for the packet injection models, where packets cannot be split and have to be sent through the network in one piece. The algorithm assumes that each node has a sufficiently large space for storing packets. Let $\bar{h}_{v,t} = h_{v,t}/\Delta$. In every time step $t \geq 1$ the discrete balancing algorithm performs the following operations.

1. For every edge $e = (v, w)$, check whether $\bar{h}_{v,t} - \bar{h}_{w,t} > \sigma$. If so, send a packet from v to w (otherwise do nothing).
2. Receive incoming packets and newly injected packets, and absorb those that reached the destination.

2.2 The fractional (σ, Δ)-balancing algorithm

Again, let $\bar{h}_{v,t} = h_{v,t}/\Delta$. In every time step $t \geq 1$ the fractional balancing algorithm performs the following operations.

1. For every edge $e = (v, w)$, check whether $\bar{h}_{v,t} - \bar{h}_{w,t} > \sigma$. If so, send a flow of $\min[\bar{h}_{v,t} - \bar{h}_{w,t} - \sigma, 1]$ from v to w (otherwise do nothing).
2. Receive incoming flow and newly injected flow, and absorb the flow that reached the destination.

We will demonstrate in the following that the two balancing algorithms perform well in a wide range of injection models, ranging from adversarial networks and injections to static networks and injection patterns.

3 Adversarial Networks and Injections

In this section we present upper and lower bounds on the number of packets or flow in the system when using the (σ, Δ)-balancing algorithm in the scenario that we have an adversarial network and adversarial packet or flow injections.

3.1 Packet injections

First we present a lower bound for the discrete algorithm, and then we present a matching upper bound (that also matches the universal lower bound in Theorem 1.2 for $T = \sigma \cdot \Delta = 1$).

Theorem 3.1 *For any $S_{\max}, B \in \mathbb{N}$ and any $T \in \mathbb{N}_0$, there exists an adversary such that the number of packets in the system when using the (σ, Δ)-balancing algorithm with $T = \sigma \cdot \Delta$ is at least $\max[S_{\max}, S_{\max} + T - 1] \cdot \binom{n-1}{2} + S_{\max}$.*

Proof. We will only consider the case $T > 0$, since the bound for $T = 0$ follows from Theorem 1.2. Let be nodes be numbered from 0 to $n-1$ and let node 0 represent the destination. Initially, $h_{0,0} = -\infty$ and $h_{i,0} = 0$ for all $i > 0$. Let the *potential* of the system at any given time be defined as $\Phi_t = \sum_{i=1}^{n-1} h_{i,t}^2$.

The adversarial strategy works in rounds that are performed one after the other. Each round takes $\tau = 2S_{\max} + 1$ time steps. Suppose that at the beginning of a round at some time step t_0 there are two non-destination nodes v and w with $h_{v,t_0} \geq h_{w,t_0}$ and $h_{v,t_0} < h_{w,t_0} + (S_{\max} + T - 1)$. If no such pair exists, then the amount of packets in the system must be at least $(S_{\max} + T - 1)\binom{n-1}{2}$. Injecting $S_{\max}$ more packets at some node then results in the lower bound. Otherwise, the adversary injects $S_{\max}$ schedules $S_1, \ldots, S_{S_{\max}}$, where schedule S_i is of the form $(((\times, v), t_0), ((v, w), t_0 + i), ((w, 0), t_0 + S_{\max} + i))$. As soon as the difference in height between node v and w is at most T, no packet will be moved any more from v to w. Since $h_{v,t_0} < h_{w,t_0} + (S_{\max} + T - 1)$, this will happen before the edge activation $((v, w), t_0 + S_{\max})$ because $(h_{v,t_0} - h_{w,t_0}) + 1 - (S_{\max} - 1) \leq T$. Hence, at least one injected packet will remain at v. Thus, at the end of the round, $h_{v,t_0+\tau} \geq h_{v,t_0} + \delta$ for some $\delta > 0$ and $h_{v,t_0+\tau} + h_{w,t_0+\tau} \geq h_{v,t} + h_{w,t}$. Hence,

$$\begin{aligned} h_{v,t_0+\tau}^2 + h_{w,t_0+\tau}^2 &\geq (h_{v,t_0} + \delta)^2 + (h_{w,t_0} - \delta)^2 \\ &= h_{v,t_0}^2 + h_{w,t_0}^2 + 2\delta(h_{v,t_0} - h_{w,t_0} + 1) > h_{v,t_0}^2 + h_{w,t_0}^2 \ . \end{aligned}$$

Thus, the adversary will manage to increase the potential. Therefore, if the balancing algorithm can achieve stability at all, there must be a time when there exists no v and w with $h_{v,t} \geq h_{w,t}$ and $h_{v,t} < h_{w,t} + (S_{\max} + T - 1)$, which concludes the proof. $\square$

Theorem 3.2 *For any $S_{\max}, B \in \mathbb{N}$ and any σ and Δ with $T = \sigma \cdot \Delta \geq 2(B-1)$, the discrete (σ, Δ)-balancing algorithm ensures that there are at most $\max[S_{\max}, S_{\max} + T - 1] \cdot \binom{n-1}{2} + S_{\max}$ packets in the system at any time. Furthermore, the maximum number of packets in a node at any time is at most $\max[S_{\max}, S_{\max} + T - 1](n-2) + S_{\max}$.*

Proof. For our proof we will always view the nodes as being sorted in the order of decreasing height, i.e. for every time step t we assign positions to the nodes so that $h_{n-1,t} \geq h_{n-2,t} \geq \ldots \geq h_{0,t}$. We assume that the height of the destination is always $-T$ (which has the same effect as $-\infty$). The position of a node v is *left* (resp. *right*) from a node w if the positions i of v and j of w fulfill $i > j$ (resp. $j < i$). We will assume that every node has sufficiently many slots to store packets. The slots are numbered in a consecutive way starting with 1. Every slot can store at most one packet. After every step of the balancing algorithm we assume that if a buffer holds h packets, then its first h slots are occupied. The *height* of a packet is defined as the number of the slot in which it is currently stored. If the balancing algorithm allows k packets to be moved out of some node, we will assume for the proof that always the k packets of highest height are taken. If a new packet is injected, it will obtain the lowest slot that is available after all packets that are moved to that node from another node have been placed.

A slot i in position j of the sorted node sequence is called a *slope slot* if $i \leq \max[S_{\max}, S_{\max} + T - 1] \cdot (j-1)$. The set of all slope slots is defined as the *slope* (see also Figure 1). The definition of the slope ensures that all non-slope slots (except for those in the destination, which we will not need to consider) have positive numbers. That is, if we know for some node that it has a height of beyond the slope, then there must be packets stored in it in non-slope positions.

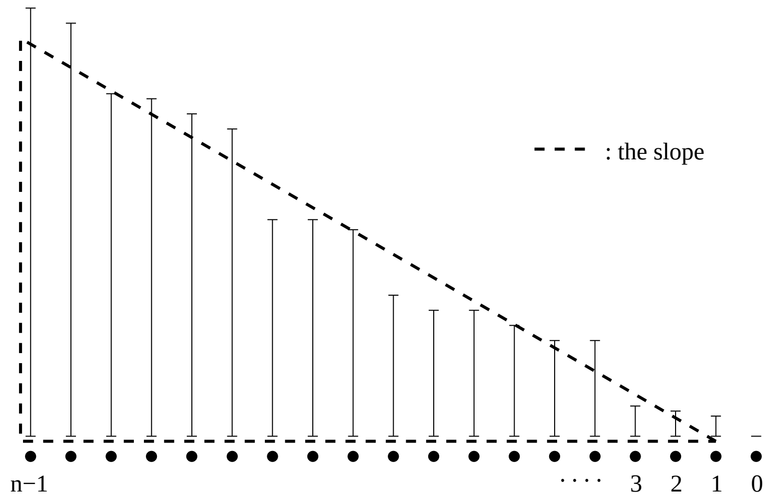

Figure 1: Illustration of the node ordering and the slope.

In order to show that the number of packets in the system is bounded, we will compare the actions of an optimal strategy (given by the schedules of the adversary) with the actions of our balancing algorithm. To do so we will divide the packets into so-called *slope packets* and *representatives*. When a new packet is injected into the system, we declare it a representative of the corresponding schedule. During the lifetime of the schedule, the role of the representative may be passed on to other packets. However, as we will show, there will be a time point before the schedule becomes inactive when we are either able to transform the representative into a slope packet (without passing on the role of the representative to another packet) or the representative must reach the destination. This will limit the number of representatives to be at most the number of active schedules at any time step.

For the proof of the theorem we will take care that all packets in the system can either be exclusively assigned to a slope slot (if it is a slope packet) or to an active schedule (if it is a representative). This will ensure that the number of packets in the system can never exceed $\max[S_{\max}, S_{\max} + T - 1] \cdot \binom{n-1}{2} + S_{\max}$. The mapping of the slope packets to slope slots is called *slope mapping*, and the mapping of the representatives to the active schedules is called *schedule mapping*. Next we define properties these mappings must have. A slope mapping is called *legal* if

1. it is one-to-one and

2. ensures that every slope packet at height h is assigned to a slope slot at height h' with $h' \geq h$.

Note that a legal slope mapping does not require that slope packets are stored in slope slots.

Given a schedule $S = ((e_0, t_0), \ldots, (e_\ell, t_\ell))$ and a time step t, the *position* of S at time t is defined as the node at which its packet would have been if all edge activations in S up to t had been used by it. Now, we define a schedule mapping to be *legal* if

1. it is one-to-one and

2. ensures that every representative at a node of height h is assigned to a schedule at a node of height h' with $h' \geq h$.

Finally, we define the system to be in a *legal state* if

1. the slope mapping is legal,
2. the schedule mapping is legal, and
3. all representatives are stored in non-slope positions.

Properties 1 and 2 ensure that in a legal state the number of packets is bounded. Property 3 is necessary for technical reasons that will become apparent in the proof of the lemma below. Since the balancing algorithm starts with an empty system, which is certainly legal, the following lemma completes the proof of Theorem 3.2. □

Lemma 3.3 *For any adversary and any time step t, the balancing algorithm ensures the following: If the system is in a legal state at the beginning of t, it will also be in a legal state at the end of t.*

Proof. The key to the proof of the lemma is to use two strategies: the *escape strategy* and the *exchange strategy*. As we will see, these strategies will allow us to get always back to a legal state.

Escape strategy: Suppose that it happens that some representative R moves to some slope slot at node v at step t. Then we show that either R can be moved to some non-slope slot at some node w with $h_{w,t} \leq h_{v,t}$ or R can be converted into a slope packet.

If no slope packet points to the slot of R, then R can be converted into a slope packet. If a slope packet is mapped to this slot, we follow the directed list of slope packets formed by the slope mapping backwards from the position of R until we reach some slope packet, say P, that is either in a non-slope slot (that must be at a node w with $h_{w,t} \leq h_{v,t}$ due to property 2 of a legal slope mapping) or in a slope slot that is not assigned to any slope packet. In the former case, we map all slope packets of the list to their own slot except for P, exchange roles between R and P, and map P to its new slot. In the latter case, we map all slope packets of the list to their own slot. Then the slot of R becomes available and R can be transformed into a slope packet.

Exchange strategy: Suppose that we want to pass on the role of a representative R at a non-slope position in node v to some packet in node w with $h_{v,t} > h_{w,t} \geq h_{v,t} - T$ without violating any of the conditions of a legal state. Since the steepness of the slope is $\max[S_{\max}, S_{\max} + T - 1]$, w must have at least $h_{w,t} - ((h_{v,t} - 1) - \max[S_{\max}, S_{\max} + T - 1]) \geq -T + 1 + \max[S_{\max}, S_{\max} + T - 1] \geq S_{\max}$ packets in non-slope positions. Because there can be at most $S_{\max} - 1$ representatives besides R, at least one of these packets must be a slope packet, say P. We now show how to exchange the roles of R and P while remaining in a legal state.

First of all, P cannot be assigned to the slot storing P, since this is a non-slope slot. Hence, P can be seen as the beginning of a directed list of packets formed by the slope mapping. The endpoint of the list must be a slope packet that is assigned to an empty slope slot, say x. Since the difference in height between v and w is at most T and R is in a non-slope slot, there cannot be an empty slope slot between v and w. Thus, property 2 of a legal slope mapping and the fact that P is in a non-slope slot imply that x must have a height that is above the height of R.

Now, we exchange the roles of R and P. That is, P becomes the new representative and R becomes a slope packet. The slope slot x is assigned to R, and all the other slope packets in the directed list are now assigned to their own slot. This results in a legal slot mapping, which brings us back to a legal state.

Now, suppose we have a system that is in a legal state at the beginning of some time step t. In order to prove the Lemma 3.3, we will consider all events that either cause schedule movements or packet movements. Due to our model, schedule movements are always associated with the activation of an edge. Since packet movements of our algorithm are also only possible if the corresponding edge is activated, it suffices to concentrate on the edge activations. Let E_{OPT} denote the set of all edge activations that belong to an active schedule (excluding injection edges), and let E_{BAL} be the set of all edge activations that cause the movement of a packet. (All other activations of edges can be neglected, because they neither involve the movement of a schedule nor involve the movement of a packet.) There are three kinds of edge activations that require our attention:

1. edge activations in $E_{OPT} \setminus E_{BAL}$, which we call *missed movements*,
2. edge activations in E_{BAL}, called *packet movements*, and
3. injections of new packets.

We will consider these categories in the order of their numbers.

1. Missed movements

These can simply be handled by the exchange strategy.

2. Packet movements

In order to simplify the analysis of these movements, we will consider them in a one-by-one fashion. For this we will need the following result.

Claim 3.4 *For any order of moving the packets we have the following property:*
If always the highest packet of a node is moved, then the height of the packet will never increase.

Proof. Consider any order of moving packets along edges that experience packet movements by the balancing algorithm. Since $T \geq 2(B-1)$, we know that a packet is only moved from a node v to a node w at time step t if $h_{v,t} - h_{w,t} > 2(B-1)$. According to the definition of the maximum bandwidth B, every node can receive/send at most B packets during time step t. Hence, in the worst case a packet can be sent from v to w after v has sent $B-1$ other packets, and after w has received $B-1$ new packets. Since the remaining difference in height is at least 1, the claim is true. □

Assume that the system is in a legal state. Suppose that a packet is sent from node v to node w. To simplify the analysis, we first ensure that v is the rightmost node of all nodes of the same height as v and w is the leftmost node of all nodes of the same height as w. The reordering may cause representatives to be moved to slope positions, but we can always get back to a legal state by using the escape strategy.

Let P be the packet of largest height in v. If some schedule S is currently at v for which $((v,w),t)$ belongs to S and w has a lower height than the node u of its representative R, then we replace the movement of P to w by a movement of P to u and R to w (note that this does not change the packet distribution). Since the height of R can be at most as large as the height of P (recall that P is the highest packet), this does not endanger a legal slope mapping. If R is now placed in a non-slope position, then we are back in a legal state. Otherwise, we use the escape strategy described above. If there is no such schedule S, we send P from v to w. If P is a slope packet, Claim 3.4 ensures that we remain in a legal state. Otherwise, we check whether the packet is moved into a slope or non-slope position. If necessary, we use the escape strategy to get back to a legal state.

Finally, it might happen that representatives which have their schedule at v are stored at nodes that have the same height as v before moving P (or, for the special case of $T = 2(B-1)$, might be at node w with $h_{w,t} = h_{v,t} - 1$). Since the height of v decreases by 1 (and the height of w increases by 1), these representatives have to be moved to a lower node (or transformed into a slope packet). This can simply be done by moving all of them to v with the help of the exchange strategy (after, perhaps, using the escape strategy for representatives previously stored in w that ended up in a slope slot). To ensure that this also works for the special case of $T = 0$, we have to adapt the steepness of the slope to $S_{\max}$.

3. Packet injections

We will also study packet injections in a one-by-one fashion. Suppose a packet is injected in node v. For this we first ensure that v is the leftmost node of all nodes of the same height as v (applying the escape strategy to representatives if necessary). Afterwards, we place the newly injected packet into v and declare it the representative of its schedule. If it is placed in a slope slot, we use the escape strategy defined above. Finally, it can happen that due to the injection of the packet there is now some representative R that is at a higher node than its schedule. In this case we can use the exchange strategy to get back to a legal state.

Thus, after all schedule movements, packet movements, and packet injections have been executed, we are still in a legal state, which proves Lemma 3.3 □

One may ask whether small buffer sizes may improve the performance of the balancing algorithm or not. However, the next result demonstrates that too small buffers can easily cause instability.

Claim 3.5 *For $B = 1$, any $T = \sigma \cdot \Delta \geq 0$, and any $S_{\max} \in \mathbb{N}$, there exists a schedule such that a single buffer of size less than $\max[S_{\max}, S_{\max} + T - 1](n-3) + S_{\max}$ already causes instability.*

Proof. The proof basically uses the strategy in the proof of Theorem 3.1. Consider the nodes at the end to be sorted according to their height, with node $n-1$ having the largest height. Suppose that every node except of node $(n-2)$ has an unlimited buffer size, and the buffer size of $n-2$ is bounded by $s < \max[S_{\max}, S_{\max} + T - 1](n-3) + S_{\max}$. Since without any limitation on the buffer size, there must be a time t when $h_{n-1,t} \geq \max[S_{\max}, S_{\max} + T - 1](n-2)$ and $h_{n-2,t} \geq \max[S_{\max}, S_{\max} + T - 1](n-3)$ before the start of a new round, at some point t all s slots of node $n-2$ must be filled. Injecting $S_{\max}$ schedules S_i of the form $(((\times, n-1), t), ((n-1, n-2), t+i), ((n-2, 0), t+S_{\max}+i))$ must in this case increase the potential function defined in the proof of Theorem 3.1, no matter how much larger $h_{n-1,t}$ is compared to $h_{n-2,t}$. Node $n-2$ can then be again filled up using the strategy as before, and then again the schedules S_i above cause an increase in the potential function. Since the potential function never decreases when using the strategy in the proof of Theorem 3.1, repeating the process above infinitely often causes the number of packets in node $n-1$ to go to infinity, and therefore causes instability. □

3.2 Flow injections

For the fractional case, the following upper bound can be shown. The lower bound is the same as for the discrete case.

Theorem 3.6 *For any $S_{\max}, B \in \mathbb{N}$ and any σ and Δ with $\Delta \geq B - 1$ and $T = \sigma \cdot \Delta \geq B$, the fractional (σ, Δ)-balancing algorithm ensures that there is at most $(S_{\max} + T - 1) \cdot \binom{n-1}{2} + S_{\max}$ flow in the system at any time.*

Proof. We essentially use the same proof as for Theorem 3.2 with the following differences.

- Packets represent now packets of flow and can have arbitrary sizes.
- Slope slots can have arbitrary sizes. However, for a legal mapping it has to be ensured that in this case slope packets are always assigned to slope slots of the same size.
- In order to sequentialize the movement of the packets in one time step, we demand that packets arrive at a node in the order of increasing packet size. This guarantees that no piece of flow in the system will ever be moved upwards, which is important to preserve a legal slot mapping.

- If only a part of a slope packet is moved, we cut it into two slope packets: the part that is moving, and the part that is not. The slot mapping is preserved by cutting its slope slot in a corresponding way.
- If only a part of a representative is moved, we cut it into two representatives as for the slope packet above and its schedule into two schedules whose demands correspond to the sizes of the new representatives.

Furthermore, the following lemma is needed to ensure that flow only moves downwards, and therefore the slope mapping will remain legal.

Lemma 3.7 *If the flow pieces are moved sequentially in order of increasing flow values, we have the following property: if always the highest flow piece of a node is moved, then the height of the flow piece will never increase.*

Proof. If a flow piece f of value ϵ is moved from node v to node w, then $\bar{h}_{v,t} - \bar{h}_{w,t} - \sigma \geq \epsilon$, or $h_{v,t} - h_{w,t} \geq \epsilon \cdot \Delta + T \geq \epsilon(B-1) + B$. According to the definition of the maximum bandwidth B, every node can receive/send at most B flow pieces during time step t. Since the smaller flow pieces are preferred, in the worst case f is sent from v to w after v has sent $B-1$ other flow pieces of value ϵ and w has received $B-1$ new flow pieces of value ϵ. After these moves, the remaining difference in height between v and w is

$$\begin{aligned}(h_{v,t} - \epsilon(B-1)) - (h_{w,t} + \epsilon(B-1)) &= h_{v,t} - h_{w,t} - 2\epsilon(B-1) \\ &\geq \epsilon(B-1) + B - 2\epsilon(B-1) \geq 1 ,\end{aligned}$$

since ϵ can be at most 1. Thus the height of the flow piece will not increase and the lemma is true. □

Using the modifications and Lemma 3.7, the proof can be done as before. □

4 Adversarial Injections into Static Networks

In this section we assume that the injections are still adversarial, but the network is now static, i.e. its structure is fixed and no edge breaks down.

4.1 Packet injections

The following result shows that stability in our model can be transferred to stability when using any discrete (w, λ)-bounded adversary (i.e. a bounded adversary that injects discrete packets). This is done by showing that any adversary in the (w, λ)-bounded adversary can be transformed into an adversary for our adversarial model, which demonstrates that our model is at least as general as the (w, λ)-bounded adversarial model.

Theorem 4.1 *For any network of maximum degree B and any discrete (w, λ)-bounded adversary with $\lambda \leq 1$ that uses only one receiver, the discrete (σ, Δ)-balancing algorithm ensures: If $T = \sigma \cdot \Delta \geq 2(B-1)$, then there are at most $(3w \cdot m + T)\binom{n-1}{2} + 3w \cdot m$ packets in the system at any time.*

Proof. In order to prove the theorem, it suffices to show that for any discrete (w, λ)-bounded adversary there exists a routing strategy such that there are at most $w(2n + m)$ packets in the system at any time.

Consider the time to be partitioned into time frames of length w. Due to the restrictions on the (ω, λ)-bounded adversary, there are routing paths for all the packets injected during this time frame with a congestion of at most w. Let R_I be the set of routing paths for time frame I. Now suppose we use the following strategy:

For each time frame I of length w, first buffer all arriving packets in the corresponding injection buffers. During the next time frame I' of length w, send a packet over every edge that occurs in a routing path in R_I. Or more precisely, for every path $p = (v_1, v_2, \ldots, v_k)$ in R_I, the packet injected for p is sent along (v_1, v_2) and for every $i \in \{2, \ldots, k-1\}$, any available packet in a non-injection buffer in v_i is sent along (v_i, v_{i+1}).

The strategy is always feasible, since the congestion of the optimal path collection is always at most w, and therefore it is possible for every edge e to send one packet across e in I for each path crossing e (if such a packet is available). Furthermore, it has the following consequence:

Lemma 4.2 *At the end of every time frame I, every injection buffer only holds the packets injected in I, and every node of in-degree d stores at most $d \cdot w$ packets.*

The number of packets in the injection buffers is upper bounded by $2w \cdot m$, where m is the number of edges, because this is the maximum amount of packets that can be injected into the system within two time intervals of length w. Thus, together with Lemma 4.2 it follows that the total amount of packets in the system at any time cannot exceed $3w \cdot m$. Using this in Theorem 3.2 proves Theorem 4.1. $\square$

4.2 Flow injections

We can also transfer the stability results for our adversarial flow model to stability results in Garmanik's adversarial flow model [13]. The proof is similar to the proof of Theorem 4.1.

Theorem 4.3 *For any network of maximum degree B and any fractional (w, λ)-bounded adversary with $\lambda \leq 1$ that uses only one receiver, the fractional (σ, Δ)-balancing algorithm ensures: If $T = \sigma \cdot \Delta \geq B$, then there is at most $(3w \cdot m + T)\binom{n-1}{2} + 3w \cdot m$ flow in the system at any time.*

5 Static Networks and Injection Patterns

Finally, we consider the situation that the network and the injection pattern is static.

5.1 Flow injections

In this section we study the performance of the (σ, Δ)-balancing algorithm when used in the situation that we have a flow injection process where the amount of flow injected at any fixed node is the same for all time steps. The next theorem shows that in this case the distribution of flow among the nodes caused by the balancing algorithm will approach a *fixpoint*, i.e. a point in which the amount of flow at every node remains unchanged when applying the balancing algorithm to it.

Theorem 5.1 *For any static network, any sustainable static flow injection pattern, and any $\sigma \geq 0$ and $\Delta \geq B$, the distribution of the flow will converge towards a fixpoint.*

Proof. Let $\bar{h}_{v,t}$ denote the normalized height of node v at time step t and $\delta_{v,t} = \bar{h}_{v,t+1} - \bar{h}_{v,t}$. We will show that if the system starts with $\bar{h}_{v,0} = 0$ for every node v, then $\delta_{v,t} \geq 0$ for all v and t. Since this implies that the (normalized) heights of the nodes can only grow, and since we know from Theorem 4.3 that the total amount of flow in the system must be bounded, this implies that the system must converge towards a state with $\delta_{,t} = 0$ for all v, i.e. a fixpoint.

Lemma 5.2 *If $\bar{h}_{v,0} = 0$ for every node v, then $\delta_{v,t} \geq 0$ for all $t \geq 0$ and all $v \in V$.*

Proof. To simplify the proof, we will view the injected flow as flow along edges coming from imaginary nodes v for which w.l.o.g. we have $\delta_{v,t} \geq 0$ for all $t \geq 0$.

At the beginning, the lemma is obviously true for the empty system. Now suppose that we already showed for some time step t that $\delta_{v,t} \geq 0$ for all $v \in V$. Then we will show that it also holds for step $t + 1$.

Consider an arbitrary node v. We know that $\bar{h}_{v,t+1} = \bar{h}_{v,t} + \delta_{v,t}$ where $\delta_{v,t} \geq 0$. Since also $\bar{h}_{w,t+1} \geq \bar{h}_{w,t}$ for all other nodes w, we get

$$\begin{aligned}
\bar{h}_{v,t+2} &= \bar{h}_{v,t+1} + \frac{1}{\Delta}\left(\sum_{w:\, \bar{h}_{w,t+1} > \bar{h}_{v,t+1}+\sigma} \min[1,\ \bar{h}_{w,t+1} - \bar{h}_{v,t+1} - \sigma] - \sum_{w:\, \bar{h}_{w,t+1} < \bar{h}_{v,t+1}-\sigma} \min[1,\ \bar{h}_{v,t+1} - \bar{h}_{w,t+1} - \sigma]\right) \\
&\geq \bar{h}_{v,t+1} + \frac{1}{\Delta}\left(\sum_{w:\, \bar{h}_{w,t} > \bar{h}_{v,t+1}+\sigma} \min[1,\ \bar{h}_{w,t} - \bar{h}_{v,t+1} - \sigma] - \sum_{w:\, \bar{h}_{w,t} < \bar{h}_{v,t+1}-\sigma} \min[1,\ \bar{h}_{v,t+1} - \bar{h}_{w,t} - \sigma]\right) \\
&\geq \bar{h}_{v,t+1} + \frac{1}{\Delta}\left(\sum_{w:\, \bar{h}_{w,t} > \bar{h}_{v,t}+\sigma} \min[1,\ \bar{h}_{w,t} - \bar{h}_{v,t+1} - \sigma] - \sum_{w:\, \bar{h}_{w,t} < \bar{h}_{v,t+1}-\sigma} \min[1,\ \bar{h}_{v,t+1} - \bar{h}_{w,t} - \sigma]\right) \\
&= \bar{h}_{v,t+1} + \frac{1}{\Delta}\left(\sum_{w:\, \bar{h}_{w,t} > \bar{h}_{v,t}+\sigma} \min[1,\ \bar{h}_{w,t} - (\bar{h}_{v,t} + \delta_{v,t}) - \sigma] - \sum_{w:\, \bar{h}_{w,t} < \bar{h}_{v,t+1}-\sigma} \min[1,\ (\bar{h}_{v,t} + \delta_{v,t}) - \bar{h}_{w,t} - \sigma]\right) \\
&\geq \bar{h}_{v,t+1} + \frac{1}{\Delta}\left(\sum_{w:\, \bar{h}_{w,t} > \bar{h}_{v,t}+\sigma} (\min[1,\ \bar{h}_{w,t} - \bar{h}_{v,t} - \sigma] - \delta_{v,t}) - \sum_{w:\, \bar{h}_{w,t} < \bar{h}_{v,t+1}-\sigma} (\min[1,\ \bar{h}_{v,t} - \bar{h}_{w,t} - \sigma] + \delta_{v,t})\right) \\
&\geq \bar{h}_{v,t+1} - \delta_{v,t} + \frac{1}{\Delta}\left(\sum_{w:\, \bar{h}_{w,t} > \bar{h}_{v,t}+\sigma} \min[1,\ \bar{h}_{w,t} - \bar{h}_{v,t} - \sigma] - \sum_{w:\, \bar{h}_{v,t}-\sigma \leq \bar{h}_{w,t} < \bar{h}_{v,t+1}-\sigma} \min[1,\ \bar{h}_{v,t} - \bar{h}_{w,t} - \sigma]\right. \\
&\qquad \left. - \sum_{w:\, \bar{h}_{w,t} < \bar{h}_{v,t}-\sigma} \min[1,\ \bar{h}_{v,t} - \bar{h}_{w,t} - \sigma]\right) \\
&\geq \bar{h}_{v,t+1} - \delta_{v,t} + \frac{1}{\Delta}\left(\sum_{w:\, \bar{h}_{w,t} > \bar{h}_{v,t}+\sigma} \min[1,\ \bar{h}_{w,t} - \bar{h}_{v,t} - \sigma] - \sum_{w:\, \bar{h}_{w,t} < \bar{h}_{v,t}-\sigma} \min[1,\ \bar{h}_{v,t} - \bar{h}_{w,t} - \sigma]\right) \\
&= \bar{h}_{v,t+1} - \delta_{v,t} + \delta_{v,t} = \bar{h}_{v,t+1}\ .
\end{aligned}$$

Hence, also $\delta_{v,t+1} \geq 0$, which proves the lemma. □

This completes the proof of Theorem 5.1. □

In the fixpoint, the heights of the nodes do not change any more. This implies that, when the fixpoint is reached, the amount of flow entering a node is equal to the amount of flow leaving a node. Since flow can only move from higher nodes to lower nodes, this implies that the flow movements form fixed, connected, and loop-free paths. The sum of the capacities of these paths is equal to the amount of flow injected into the system in every time step. When using the LIFO (last-in-first-out) rule to send flow through the system, we obtain a worst case delay of n (a path may visit in the worst case all nodes). Let the average delay $\bar{T}_{BAL}$ of these paths be defined as follows.

$$\bar{T}_{BAL} = \frac{1}{\lambda} \sum_{\text{flow paths } p} \ell_p \cdot \lambda_p$$

where ℓ_p is the length of flow path p, λ_p is the amount of flow following path p, and $\lambda = \sum_{\text{flow paths } p} \lambda_p$. The question is how close $\bar{T}_{BAL}$ can be to a best possible average case delay, $\bar{T}_{OPT}$. The next theorem gives the answer to this question.

Theorem 5.3 *For any static network, any sustainable static flow injection pattern, any* $\sigma \in \mathbb{N}$*, and any* $\Delta \geq B$ *the* (σ, Δ)*-balancing algorithm together with the LIFO rule ensures that, in the fixpoint,* $\bar{T}_{BAL} \leq (1 + 1/\sigma)\bar{T}_{OPT}$.

Proof. We know that at the fixpoint of the balancing algorithm we have a fixed collection P of paths whose demands sum up to the total injection rate. Let Q be any collection of optimal paths for the given injection process. W.l.o.g. we can assume that P and Q have the same number of paths, and the ith path of P and Q has the same demand and connects the same source-destination pair (otherwise split the paths of P and Q into subpaths such that this property is fulfilled). Thus, let $p_1, \ldots, p_r$ be the paths in P, and let $q_1, \ldots, q_r$ be the paths in Q. Let ℓ_i denote the length of p_i and k_i denote the length of q_i. Furthermore, let λ_i be the demand of path p_i and q_i. Suppose now that

$$\sum_i \lambda_i \ell_i > \frac{\sigma+1}{\sigma} \sum_i \lambda_i k_i \ . \tag{1}$$

We will show via contradiction that this is not possible, which would prove the theorem.

For each $i \in \{1, \ldots, r\}$, let s_i denote the source of path p_i and q_i. Furthermore, for any node v let h_v denote the height of node v in the fixpoint, and for any edge $e = (v, w)$ let $\delta_e = h_v - h_w$. We define the *potential* of a collection C of paths p with demands λ_p as

$$\Phi(C) = \sum_{p \in C} \lambda_p \sum_{(v,w) \in p} (h_v - h_w) \ .$$

Then we obtain for any collection of paths C connecting the same set of endpoints as P and Q that

$$\Phi(C) = \sum_i \lambda_i h_{s_i} \ .$$

Thus, $\Phi(P) = \Phi(Q)$. For any edge e and path collection C, let $\lambda_e(C)$ denote the amount of capacity of e used by the paths in C. Furthermore, for any two path collections C_1 and C_2 let $\lambda_e(C_1 \cap C_2) = \min[\lambda_e(C_1), \lambda_e(C_2)]$ and $\lambda_e(C_1 \setminus C_2) = \max[\lambda_e(C_1) - \lambda_e(C_2), 0]$. Then we obtain from $\Phi(P) = \Phi(Q)$ that

$$\sum_{e \in E} \delta_e \lambda_e(P \cap Q) + \sum_{e \in E} \delta_e \lambda_e(P \setminus Q) = \sum_{e \in E} \delta_e \lambda_e(P \cap Q) + \sum_{e \in E} \delta_e \lambda_e(Q \setminus P)$$

and thus

$$\sum_{e \in E} \delta_e \lambda_e(P \setminus Q) = \sum_{e \in E} \delta_e \lambda_e(Q \setminus P) \ .$$

We know that for all edges e with $\lambda_e(P \setminus Q) > 0$ we have $\delta_e \geq \sigma \cdot \Delta$, and for all edges e with $\lambda_e(Q \setminus P) > 0$ we have $\delta_e \leq (\sigma + 1)\Delta$. Hence,

$$\sigma \cdot \Delta \sum_{e \in E} \lambda_e(P \setminus Q) \leq \sum_{e \in E} \delta_e \lambda_e(P \setminus Q)$$

and

$$\sum_{e \in E} \delta_e \lambda_e(Q \setminus P) \leq (\sigma + 1)\Delta \sum_{e \in E} \lambda_e(Q \setminus P)$$

and so

$$\sum_{e \in E} \lambda_e(P \setminus Q) \leq \frac{\sigma+1}{\sigma} \sum_{e \in E} \lambda_e(Q \setminus P) \ .$$

On the other hand, (1) implies that

$$\sum_{e \in E} \lambda_e(P \cap Q) + \sum_{e \in E} \lambda_e(P \setminus Q) > \frac{\sigma+1}{\sigma} \left(\sum_{e \in E} \lambda_e(P \cap Q) + \sum_{e \in E} \lambda_e(Q \setminus P) \right)$$

and therefore

$$\sum_{e \in E} \lambda_e(P \setminus Q) > \frac{\sigma+1}{\sigma} \sum_{e \in E} \lambda_e(Q \setminus P) ,$$

which is a contradiction. $\Box$

Since there are examples that show that $\bar{T}_{BAL}$ is not approached in a monotonic way, it is extremely difficult to prove how quickly the system converges to $\bar{T}_{BAL}$. However, we can show that the flow always has a very special property formulated in the next theorem.

Theorem 5.4 *When using the LIFO rule, any flow piece that reaches the destination has been travelling without waiting.*

Proof. For any time step t and node v, let $h_{v,t}^{(o)}$ denote the amount of flow in node v after the flow has been sent out of v and before new flow has been received by v at step t. We will show by induction that for every v and t, $h_{v,t+1}^{(o)} \geq h_{v,t}^{(o)}$. This will then immediately imply the theorem.

At the beginning, the hypothesis is certainly true. So assume that it is true for all time steps $\tau \leq t$. Then we will show that it is also true for step $t+1$. Using the notation in the proof of Theorem 5.1, it holds that

$$\begin{aligned}
\bar{h}_{v,t+1}^{(o)} &= \bar{h}_{v,t+1} - \frac{1}{\Delta} \sum_{(v,w)\in E} \min[1,\ \max[\bar{h}_{v,t+1} - \bar{h}_{w,t+1} - \sigma,\ 0]] \\
&= \bar{h}_{v,t}^{(o)} + \frac{1}{\Delta} \sum_{(w,v)\in E} \min[1,\ \max[\bar{h}_{w,t} - \bar{h}_{v,t} - \sigma,\ 0]] \\
&\quad - \frac{1}{\Delta} \sum_{(v,w)\in E} \min[1,\ \max[\bar{h}_{v,t+1} - \bar{h}_{w,t+1} - \sigma,\ 0]] \\
&= \bar{h}_{v,t}^{(o)} + \frac{1}{\Delta} \left(\sum_{(v,w)\in E} \min[1,\ \max[\bar{h}_{v,t} - \bar{h}_{w,t} - \sigma,\ 0]] \right) + (\bar{h}_{v,t+1} - \bar{h}_{v,t}) \\
&\quad - \frac{1}{\Delta} \sum_{(v,w)\in E} \min[1,\ \max[\bar{h}_{v,t+1} - \bar{h}_{w,t+1} - \sigma,\ 0]] \\
&\geq \bar{h}_{v,t}^{(o)} + \frac{1}{\Delta} \left(\sum_{(v,w)\in E} \min[1,\ \max[\bar{h}_{v,t} - \bar{h}_{w,t} - \sigma,\ 0]] \right) + (\bar{h}_{v,t+1} - \bar{h}_{v,t}) \\
&\quad - \frac{1}{\Delta} \sum_{(v,w)\in E} \min[1,\ \max[\bar{h}_{v,t} + (\bar{h}_{v,t+1} - \bar{h}_{v,t}) - \bar{h}_{w,t} - \sigma,\ 0]] \\
&\geq \bar{h}_{v,t}^{(o)} + \frac{1}{\Delta} \left(\sum_{(v,w)\in E} \min[1,\ \max[\bar{h}_{v,t} - \bar{h}_{w,t} - \sigma,\ 0]] \right) + (\bar{h}_{v,t+1} - \bar{h}_{v,t}) \\
&\quad - \frac{1}{\Delta} \left(\sum_{(v,w)\in E} \min[1,\ \max[\bar{h}_{v,t} - \bar{h}_{w,t} - \sigma,\ 0]] + (\bar{h}_{v,t+1} - \bar{h}_{v,t}) \right) \\
&= \bar{h}_{v,t}^{(o)} .
\end{aligned}$$

This completes the induction and therefore the proof of the theorem. $\Box$

Furthermore, we can bound the time it takes until the system reaches a maximum possible throughput. A system is called to reach its ϵ-*almost stable* state at time t if for any time step $t' \geq t$ the difference between the flow injected into the network and the flow absorbed in the destination is at most ϵ.

Theorem 5.5 *For any static graph of n nodes and m edges, any sustainable static flow injection pattern, and any $\epsilon > 0$, it takes at most $m \cdot n^2/\epsilon$ steps to reach an ϵ-almost stable state.*

Proof. From the proof of Lemma 5.2 we know that the heights of the nodes are monotonically increasing. Since the height of the destination will always be 0, this means that the amount of flow reaching the destination is monotonically increasing over the time. Let t be the first time step in which the amount of flow absorbed in the destination is by at most ϵ less than the injected flow. Then for all time steps $t' < t$ at least an ϵ amount of the injected flow remained in the system. Under the assumption that the total amount of flow in the system can be bounded by F, t can be at most F/ϵ. It therefore remains to give a bound for F.

Lemma 5.6 *For any static network with n nodes and m edges and any sustainable static injection process, σ and Δ can be chosen so that the amount of flow in the system at any time when using the fractional (σ, Δ)-balancing algorithm is at most $m \cdot n^2$.*

Proof. If we have a sustainable static injection process, then there must be some fixed set of flow paths that can be used to send the injected flow to the destination. This set of paths can be easily obtained by solving the single commodity flow problem using, for instance, the Edmonds-Karp method [10]. Given these paths, an optimal solution to send the injected flow is simply to send it along these paths without waiting. In this case, the amount of flow in the system (and therefore the amount of active flow) is bounded by the sum of the lengths of the paths. This sum is always bounded by the total number of edges in the system, m. Hence, $S_{\max} \leq m$. Furthermore, B can be at most n. So we can choose $\sigma \cdot \Delta = n$. In this case, Theorem 3.6 implies that the total amount of flow in the system can be at most $(m + n - 1) \cdot n^2/2 \leq m \cdot n^2$. □

The lemma completes the proof of Theorem 5.5. □

5.2 Packet injections

Next we investigate the situation that we have a static packet injection process, that is, the amount of packets injected at a node is the same for all time steps. We use simulations of the fractional balancing algorithm instead of considering directly the discrete algorithm, because this tremendously simplifies the analysis. It also allows us to argue about the behavior in the fixpoint, although the discrete algorithm itself does not converge to one.

Every edge $e = (v, w)$ gets a counter z_e. Initially, $z_e = 0$ for all e. Every time a flow of value f crosses e, we increase z_e by f. If $z_e \geq 1$, then we send a packet along e (if v has at least one packet available for this) and subtract 1 from z_e.

Let $g_{v,t}$ denote the number of packets in node v at time t and $h_{v,t}$ denote the amount of flow in node v at time t. The next theorem shows that with the simulation strategy above the distribution of the packets will, under certain circumstances, always be close to the distribution of flow in the system.

Theorem 5.7 *Suppose that we use the fractional (σ, B)-balancing algorithm with $\sigma \geq 2$. Then the deterministic simulation ensures that for every node v and time step t, $|h_{v,t} - g_{v,t}| \leq B$.*

Proof. We call a node v *active* at time step t if $\bar{h}_{v,t} \geq \sigma$. Let the flow reaching v at time t be defined as $f^i_{v,t}$ and the flow leaving v at time t be defined as $f^o_{v,t}$. From the balancing rule of the (σ, Δ)-balancing algorithm we know that a node v can only send flow to some node w if $\bar{h}_{v,t} - \bar{h}_{w,t} \geq \sigma$. Hence, any node that sends out flow must be active. From the proof of Theorem 5.1 we also know that the heights of the nodes are monotonically increasing. Thus, for every time step the incoming flow is always at least as large as the outgoing flow. We sum up these important observations in the following lemma.

Fact 5.8

1. *Every node v that becomes active at time step t for the first time must fulfill*

$$\sum_{\tau \leq t} f^i_{v,\tau} \geq \sigma \cdot \Delta \quad \textit{and} \quad \sum_{\tau \leq t} f^o_{v,\tau} = 0 \; .$$

2. *For every node v and every time step t, $f^i_{v,t} \geq f^o_{v,t}$.*

The next lemma states that active nodes will never run out of packets during the simulation.

Lemma 5.9 *For any edge $e = (v, w)$ and time step t in which $z_{e,t}$ exceeds 1, a packet can be sent from v to w.*

Proof. We prove the lemma by complete induction over the amount of time considered so far. At the beginning, the lemma is certainly true. So assume now that it is true up to some time step $t \geq 1$. Then we will show that it is also true for time step $t+1$. Consider some fixed node v. Let $f^o_{v,t}$ and $f^i_{v,t}$ be defined as above. Recall that we want to choose $\Delta = B$ (see the theorem). Thus, according to Fact 5.8, there is some flow value f so that

$$\sum_{\tau=1}^{t} f^i_{v,\tau} \geq f + \sigma \cdot B \quad \text{and} \quad \sum_{\tau=1}^{t} f^o_{v,\tau} \leq f \; .$$

Hence, at most f packets have been sent out by v up to step t. Furthermore, we know from the induction hypothesis that v has received at least $f + \sigma \cdot B - B$ packets up to step t. Since $\sigma \geq 2$, there must be at least B packets that are stored at v at the beginning of step $t+1$. Hence, no matter which edges decide to send out packets at step $t+1$, there will be enough packets available for this. □

Lemma 5.9 implies at every time step t it holds for the number of packets $g^i_{v,t}$ that reached v up to t and the amount of flow $h^i_{v,t}$ that reached v up to t that $h^i_{v,t} - g^i_{v,t} \in [0, B]$. Furthermore, it holds for the number of packets $g^o_{v,t}$ that left v up to t and the amount of flow $h^o_{v,t}$ that left v up to t that $h^o_{v,t} - g^o_{v,t} \in [0, B]$. Hence,

$$|h_{v,t} - g_{v,t}| = |(h^i_{v,t} - h^o_{v,t}) - (g^i_{v,t} - g^o_{v,t})| = |(h^i_{v,t} - g^i_{v,t}) - (h^o_{v,t} - g^o_{v,t})| \leq B \; .$$

□

6 Conclusions and Open Problems

In this paper we presented simple balancing algorithms for adversarial packet and flow routing in adversarial and static networks. We showed that for both ends of the spectrum (a completely unpredictable network and injection process, and a static network and injection pattern), the balancing algorithms can compete with best possible off-line strategies.

Many open questions remain. The most important is: does our model also allow stability in the case of multiple receivers? How does the balancing algorithm have to look like in this case? How do fixpoints look in this case, if there are any? How good can the average delay of packets be compared to best possible off-line strategies? Also for the case of a single destination there are several questions left. For instance, is the (σ, Δ)-balancing algorithm stable for any $\Delta \geq B$, or even for any $\Delta > 0$? How much time does it take to move from one fixpoint to another if the injection is changed from one static process to another? Are there good parameters for describing this effect?

References

[1] Y. Afek, B. Awerbuch, E. Gafni, Y. Mansour, A. Rosen, and N. Shavit. Slide – the key to polynomial end-to-end communication. *Journal of Algorithms*, 22(1):158–186, 1997.

[2] Y. Afek and E. Gafni. End-to-end communication in unreliable networks. In *Proc. of the 7th IEEE Symp. on Principles of Distributed Computing (PODC)*, pages 131–148, 1988.

[3] W. Aiello, B. Awerbuch, B. Maggs, and S. Rao. Approximate load balancing on dynamic and synchronous networks. In *Proc. of the 25th ACM Symp. on Theory of Computing (STOC)*, pages 632–641, 1993.

[4] W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosén. Adaptive packet routing for bursty adversarial traffic. In *Proc. of the 30th ACM Symp. on Theory of Computing (STOC)*, pages 359–368, 1998.

[5] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention-resolution protocols. In *Proc. of the 37th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 380–389, 1996.

[6] B. Awerbuch and F. Leighton. A simple local-control approximation algorithm for multicommodity flow. In *Proc. of the 34th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 459–468, 1993.

[7] B. Awerbuch and F. Leighton. Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks. In *Proc. of the 26th ACM Symp. on Theory of Computing (STOC)*, pages 487–496, 1994.

[8] B. Awerbuch, Y. Mansour, and N. Shavit. End-to-end communication with polynomial overhead. In *Proc. of the 30th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 358–363, 1989.

[9] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queueing theory. In *Proc. of the 28th ACM Symp. on Theory of Computing (STOC)*, pages 376–385, 1996.

[10] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[11] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.

[12] D. Gamarnik. Stability of adversarial queues via fluid models. In *Proc. of the 29th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 60–70, 1998.

[13] D. Gamarnik. Stability of adaptive and non-adaptive packet routing policies in adversarial queueing networks. In *Proc. of the 31st ACM Symp. on Theory of Computing (STOC)*, pages 206–214, 1999.

[14] A. Goel. Stability of networks and protocols in the adversarial queueing model for packet routing. In *Proc. of the 10th ACM Symp. on Discrete Algorithms (SODA)*, pages 911–912, 1999.

[15] A. Goldberg. A new max-flow algorithm. Technical Report MIT/LCS/TM-291, MIT, 1985.

[16] A. Goldberg and R. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35(4):921–940, 1988.

[17] C. Scheideler and B. Vöcking. From static to dynamic routing: Efficient transformations of store-and-forward protocols. In *Proc. of the 31st ACM Symp. on Theory of Computing (STOC)*, pages 215–224, 1999.

[18] P. Tsaparas. Stability in adversarial queueing theory. Master's thesis, Dept. of Computer Science, University of Toronto, 1997.