Adaptive Packet Routing for Bursty Adversarial Traffic

William Aiello* Eyal Kushilevitz† Rafail Ostrovsky‡ Adi Rosén§

Abstract

One of the central tasks of networking is *packet-routing* when edge bandwidth is limited. Tremendous progress has been achieved by separating the issue of routing into two conceptual sub-problems: *path selection* and *congestion resolution* along the selected paths. However, this conceptual separation has a serious drawback: each packet's path is *fixed* at the source and cannot be modified adaptively en-route. The problem is especially severe when packet injections are modeled by an *adversary*, whose goal is to cause "traffic-jams".

In this paper, we consider this adversarial setting, motivated by the "adversarial queuing theory" model of Borodin et al. [BKR+]. More precisely, we consider an adversary who injects packets, with only their destinations specified, into network nodes in a continuous manner subject to certain limitations on the injection rate. The question whether it is possible to deal with such an adversary and to design protocols that would "discover" routes which avoid "traffic jams" so that nodes only store a bounded number of packets, was left as an open problem by Andrews et al. [AAF+] (who deal with the "non-adaptive" case where the adversary provides routes for the packets). In the present paper, we resolve this open problem. In particular, we present a simple, deterministic, local-control protocol that applies to any network topology. Our protocol guarantees that, for any injection sequence generated by the adversary, the buffers at the nodes are polynomially-bounded and that each packet has a polynomially-bounded delivery time.

*AT&T Shannon Laboratory, C215 Bldg 103, 180 Park Avenue, Florham Park NJ 07932 e-mail: aiello@research.att.com

†Dept. of Computer Science, Technion, Haifa, Israel. e-mail: eyalk@cs.technion.ac.il; Part of this research was done while visiting DIMACS. URL:http//www.cs.technion.ac.il/∼eyalk

‡Telcordia Technologies (formerly Bellcore) MCC-1C365B 445 South Street, Morristown, NJ 07960-6438 e-mail: rafail@research.telcordia.com

§Dept. of Computer Science, University of Toronto, Toronto, Canada. Part of this research was done while visiting DIMACS. e-mail: adiro@cs.toronto.edu

1 Introduction

Packet routing is one of the central issues in the areas of parallel computing and networking (see a survey of Leighton [L]). In this paper, we consider packet routing in the setting of arbitrary synchronous networks. The study of packet routing in this setting can be categorized along two different axes: (1) whether path selection is non-adaptive (i.e., each packet's path through the network is fixed at its source) or adaptive (i.e., each packet's path through the network can be modified en-route); and (2) whether the injection of packets is static (i.e., a finite number of packets are injected once and then routed to completion), or dynamic (i.e., packets arrive continuously and are routed continuously).

NON-ADAPTIVE VS. ADAPTIVE PATH SELECTION: In non-adaptive routing, the entire path of each packet is chosen before the packet crosses its first edge. Thus, any contention resolution protocol at the nodes cannot influence the packets' paths. This feature allows the routing algorithm to be divided into two separate steps: selection of paths for packets and contention resolution at edges congested by the selected paths [LMR]. Separating the path selection from the contention resolution nicely models virtual circuit routing or (randomized) oblivious routing. Moreover, this separation is extremely helpful from the standpoint of algorithmic design and analysis. Indeed, there is a large body of work on path selection and contention resolution for minimizing queue size and latency, that uses this approach. See, for example, [ST, AAP, KT, KPP] for path selection and [AAF+, BFU, LMR, LMRi, PG, PG2, RT, OR] for packet-scheduling.

The non-adaptive approach, however, does not allow packets to dynamically adapt to congestion and faults along their routes. In adaptive path routing, the contention resolution protocol of an intermediate node may "reroute" a packet in an attempt to achieve smaller packet latency or higher network throughput. For example, adaptive routing algorithms for general networks appear in work on end-to-end communication [AMS, AGR, AAG+], and in work on multicommodity flow [AL, AL2]. There, the paths that the packets follow depend on the local traffic conditions.

STATIC VS. DYNAMIC ROUTING: Until recently, routing algorithms which afforded worst-case analysis on latency and queue size were limited to the static routing problem. There is a long history of work in this area. See, for example, [L, LMR, LMRi, RT, OR]. The dynamic routing problem can be solved using a static routing algorithm by running the static routing algorithm periodically, storing packets injected during a given run of the algorithm, and using those packets as input for the next run. However, as this approach will result in large packet latencies and in inefficient use of network resources, better dynamic routing algorithms are desirable.

In analyzing routing algorithms for dynamic routing problems, there are generally many ways to model the problem. First, a parameterized model of the packet injections needs to be specified. Also, the analysis can be done assuming either unlimited or limited buffer sizes. In the case of unlimited buffer sizes, the proper goal is generally to determine the maximum buffer size obtained as a function of the injection parameters. In the case of limited buffer sizes, the goal is to determine the fraction of packets that are dropped as a function of the

injection parameters and the buffer sizes. In this paper, we assume unlimited buffer sizes.

Usually, packet injection is modeled by a probabilistic process, such as a Poisson arrival process at each node with destinations chosen independently and uniformly at random. In such cases the performance is often measured in terms of the *expected* latency and queue size. See, for example, [BU, BFU, HB, HW, SV, STs, M, Mi]. The first attempt to develop a model of dynamic routing for analyzing the queue size and latency in the worst-case was made by Cruz [C, C2]. In his model, one assumes arbitrary virtual circuits are established each with a source of fixed rate (with bounded bursts allowed) subject to the constraint that the total rate of all sessions using a given edge is strictly less than 1. Borodin et al. [BKR+] introduce "adversarial queuing theory" which models general non-adaptive path routing. In their model, packets are not restricted to virtual circuits—each packet is given an arbitrary path at injection by an adversary. The adversary is characterized by a rate constraint, $\varepsilon > 0$: in every window of time of any length t, the number of paths corresponding to packets injected in that window of time which pass through any edge must be at most $\lceil (1-\varepsilon)t \rceil$. Several bounds on the buffer sizes of several protocol are derived in [BKR+] using this model.

In [AAF+] the model of [BKR+] is generalized to allow bounded bursts. Now the adversary is characterized by both a rate parameter $\varepsilon > 0$ and a window parameter w as follows: in every window of time of size w, the number of paths corresponding to packets injected in that window which pass through any edge must be at most $(1-\varepsilon)w$. Note that the larger the value of w, the larger the bursts of injections of packets/paths using specified edges the adversary is allowed. The maximum queue size of a protocol becomes a function of both the network size and the parameters w and $1/\varepsilon$. [1] Andrews et al. show that several well-known and simple deterministic greedy queuing protocols yield queues (and latencies) which are bounded. However, these bounds are exponentially large. They also describe a randomized protocol with expected polynomial queue size and latency.

Our results: We consider the setting of dynamic and adaptive packet routing without probabilistic assumptions on the injection of packets. One of the main open problems in [AAF+] was to adapt the adversarial routing model of the non-adaptive routing problem to an adversarial routing model for the adaptive routing problem (i.e., the adversary injects packets with specified destinations but without specified paths) and to find a universal protocol that keeps buffers bounded in this model.

In this paper we resolve this problem. Our model is a natural extension of the "adversarial queuing theory" model [BKR+, AAF+] to the setting of adaptive path selection. Briefly, we consider the same (w, ε) adversaries as in the non-adaptive case; that is, the limitation on the sequence of injections is that there exist paths for the packets injected in every consecutive w time steps that do not use any edge more than $(1-\varepsilon)w$ times. The difference is that in the adaptive setting these paths are *not* provided to the protocol with the packets; only the destination is specified. For this model we present a protocol to route the packets that can be applied to any network and to any sequence of injections generated by a (w, ε) adversary

[1] The burst models of [C, C2] and [AAF+] are the same for $\varepsilon > 0$. However, due to the fact that the [C, C2] adversary is limited to virtual circuits, "traffic shaping" can be applied at the source so that the traffic across edges is limited to a maximum rate as in [BKR+].

with any $w > 1$ and any $\varepsilon > 0$. Moreover, our protocol is distributed and deterministic, and it requires buffers of only polynomial size (in the size of the network and the parameters w and $\min(1/\varepsilon, w)$).

Actually, we present three protocols with polynomial sized buffers. They differ in what is assumed and what is achieved. The first and simplest protocol we denote BASIC. It assumes that each node knows its neighbors' state at the beginning of each time step and achieves (polynomially) bounded queue sizes. The second we denote DBASIC. In this protocol we do not assume knowledge of neighbors' state, and we use $O(\log n)$ control bits are piggybacked on each packet. We show that each node is able to maintain a sufficient approximation of its neighbors' state so that the queues remain polynomially bounded as in BASIC. The third protocol, BOUNDEDDT, is a modification of DBASIC which guarantees a bound on the latency of each injected packet in addition to a (polynomial) bound on queue sizes.

OUR TECHNIQUES AND RELATED WORK: Our algorithms use and extend several previous techniques. In particular they use "diffusion-type" techniques similar to those used, for example, for communication networks in [AGR, AMS, AAG+], and continuous or discrete load balancing (see [AAMR, GM, GL+, M]). As such, our algorithms are not "greedy" in the sense of [AAF+].

Our protocols and analyses are also related to the [AL, AL2] algorithms. While these algorithms were designed primarily as sequential multicommodity flow algorithms, they can be interpreted as distributed, local control routing algorithms. Their basic routing protocol assumes that each origin and destination pair, (u, v), has a fixed number of packets injected per time step, $d_{u,v}$, subject to the following constraint. For each packet injected, a path can be specified such that no edge carries more than $1 - \varepsilon$ times its capacity. The protocol and analysis assume that each node knows each (non-zero) $d_{u,v}$. This basic protocol and analysis can be extended as follows [L2]. For a fixed window of time of size w, and for each origin and destination pair, (u, v), the adversary can inject $\bar{d}_{u,v}w$ packets in u destined for v in a window of size w subject to the following constraint. For each packet injected in a window, a path can be specified such that no edge carries more than $1 - \varepsilon$ times its capacity times w. The protocol assumes that each node knows each (non-zero) $\bar{d}_{u,v}$ (the analysis makes use of the window size w). In our model of adversarial injection the number of packets injected with origin, destination (u, v) in one window need not have any relationship to the number of such packets injected in another window. Moreover, the algorithms in [AL, AL2] make use of the knowledge of the the average injection rate, and thus are not able to handle our more general injection adversary.

The algorithms in [AL, AL2] assume that each node knows its neighbors' states (like our first protocol BASIC). Maintaining this information in general networks may require a very large number of control bits. Thus, these algorithms do not yield fully distributed protocols. Our protocol DBASIC uses estimates on the states of neighboring nodes, rather than using their exact states, and maintains this information using $O(\log n)$ control bits piggybacked on each packet.

Like the protocols of [AL, AL2], our BASIC and DBASIC protocols are robust even if the network is changing dynamically. We allow the adversary to control the capacity of each

edge. As before the adversary also injects packets. We require that in a given window, the adversary can specify paths for the injected packets in such a way that the average capacity of an edge in the window is not exceeded. See Section 5 for more details.

Like our protocols BASIC and DBASIC, the algorithms in [AL, AL2] are not guaranteed to deliver all injected packets to their destination. Our BOUNDEDDT protocol solves the problem of guaranteeing delivery of all packets and, in fact, bounds the latency of delivery. Briefly, protocol BOUNDEDDT runs protocol DBASIC for most time steps but reserves a few time steps for a static routing protocol that will deliver packets that stayed in the network "for too long". Once the static routing protocol has completed delivering its packets, all the packets in the queues of DBASIC are transferred to the queues of the static routing protocol. Since the queues of DBASIC are bounded, there is a bounded number of packets that are being transfered, and the static protocol can route these packets in a bounded amount of time (see Section 4 for more details).

Organization: In Section 2 we give some definitions including a formal definition of the adversary. In Section 3, we give our basic protocol. This protocol solves the problem in that it requires only polynomial size buffers but it does not guarantee bounded delivery time for the packets. In Section 4 we modify the protocol so as to have bounded delivery time as well. Section 5 briefly discusses several extensions.

2 The Model

We model a communication network by a graph $G = (V, E)$, where $|V| = n$ and $|E| = m$. Each node $v \in V$ models a processor, and each edge $e \in E$ models a link between two processors. The processors store and forward packets. Packets at each node are stored in *buffers*. The network is synchronous; we number the time steps, known to all processors, by $t \in \mathbb{N} = \{1, 2, 3, \ldots\}$. We model each edge as bidirectional, i.e., in each time step each edge can deliver one packet in each direction (See Section 5 for extensions to capacitated edges, directed edges, and faulty edges). We sometimes consider *windows* of time, which are continuous sequences of time steps. We denote by $\mathcal{W}_w^t$ the time window from the start of time step t to the end of time step $t + w - 1$.

Each time step is conceptually partitioned into three sub-steps: first, packets may be sent between neighbors across edges, at most one packet per edge per time step in each direction; next, each node accepts all new packets injected from the outside; finally each node removes all packets that have reached their destination. We adopt the following convention on the notation of times. For times which are the beginning of a time step we use the simple notation t (without any primes). For times that are the end of a time step we use the notation t'.

New packets may be injected into the network at each time step. Each packet is injected into an arbitrary source processor s, and has some arbitrary destination processor d that it has to reach. Thus, a packet $p = (s, d)$ is specified by the node into which it is injected and the destination of the packet. No route is given. The sequence of injected packets is

controlled by an *adversary*.

Definition 1: We say that the adversary injecting packets is an $A(w, \varepsilon)$ adversary, for some $\varepsilon > 0$ and some integer $w > 1$, if the following holds: for any time $t \in \mathbb{N}$, let $\mathcal{I}^t$ be the set of packets injected during the w time steps from t to $t + w - 1$, inclusive. Then, the adversary can associate with each packet $p = (s, d) \in \mathcal{I}^t$, a simple path from s to d, such that each direction of every edge $e \in E$ is used by these paths at most $\lfloor (1 - \varepsilon)w \rfloor$ times.

Remark: A packet p injected at time t' will be in $\mathcal{I}^t$ for all t, $t' \leq t < t' + w$. The paths that the adversary selects for p in the definition above need not be the same for all t, $t' \leq t < t' + w$. This is in contrast to the adversarial model for non-adaptive routing in which the adversary selects a path which the packet has to follow, and which cannot change for each window.

3 The Main Protocol

In this section we present our main protocol. This protocol guarantees that the buffers at each node remain bounded for any input sequence given by any $A(w, \varepsilon)$ adversary. That is, the size of the buffers is at most some (polynomial) function of the size of the network and the parameter $\min(w, 1/\varepsilon)$ (although the values of w and ε are *not* known to the protocol). In Section 4, we extend this protocol to guarantee that each packet is delivered within a bounded amount of time. Our protocols are local in nature; that is, decisions are taken in each processor separately, based on the information the processor has in its node only. This information includes the sizes of its own buffers as well as information gathered from control bits piggybacked on the packets. Our protocol uses $\lceil \log n \rceil + 7$ control bits per packet (the main protocol given in Section 3 uses $\lceil \log n \rceil + 4$ control bits per packet). Note that $\lceil \log n \rceil$ bits are inherently required to transmit the destination of the packet. For clarity of exposition, we first present our protocol with the assumption that each node knows the sizes of buffers on the other ends of its adjacent edges (and does not use any control bits except for specifying the destination). In Section 3.3 we show how to modify the protocol and proof so as to eliminate this assumption. Also, we note that the protocol need *not* have any topological information about the underlying graph (such information will be needed for the protocol presented in Section 4).

3.1 The Protocol

The protocol maintains several buffers in each node $v \in V$: for each edge $e = (v, u) \in E$ adjacent to v, and for every destination $d \in V$, the protocol maintains a buffer for packets bound for d. Thus, there is a buffer of packets for each triplet (v, u, d), that we denote $Q_{v,u,d}$. Denote the set of packets in $Q_{v,u,d}$ at time t by $Q^t_{v,u,d}$ and by $q^t_{v,u,d}$ the size (i.e., number of packets) of the buffer $Q_{v,u,d}$ at the time t (i.e., $q^t_{v,u,d} = |Q^t_{v,u,d}|$). At each destination d, we

consider the value of $q^t_{d,u,d}$, for any $u \in V$ and any time t which is the beginning of a time step, to be always 0 (as packets that arrive to their destinations are immediately removed).

Let $q^t_{v,*,d}$ be the number of packets at node v destined for node d at time t; i.e., the sum of all $q^t_{v,u,d}$ over $(v,u) \in E$. At the end of every time step, each node v will distribute as evenly as possible all the packets destined for d among its edges, for every d. That is, at the beginning of every time step $t \geq 2$, for all $v, d \in V$ and for all $(v,u) \in E$,

$$\lfloor q^t_{v,*,d}/\delta_v \rfloor \leq q^t_{v,u,d} \leq \lceil q^t_{v,*,d}/\delta_v \rceil, \tag{1}$$

where δ_v is the degree of node v. Since at the beginning of time step 1, when the network is empty, (1) is satisfied, this invariant is maintained by the protocol at the beginning of every time step.

The following protocol is performed by each node $v \in V$, at each time step $t \in \mathbb{N}$.
Protocol BASIC:

1. For each $e = (v,u) \in E$, let $d \in V$ be such that $q^t_{v,u,d} - q^t_{u,v,d}$ is maximal over all $d \in V$ (break ties arbitrarily). If $q^t_{v,u,d} - q^t_{u,v,d}$ is positive then send one packet over the edge e from $Q_{v,u,d}$ to $Q_{u,v,d}$.

2. Accept all packets injected by the adversary to the node v.

3. Remove any packets that arrive at their destination.

4. For every destination $d \in V$, redistribute all packets among the corresponding buffers so as to maintain invariant (1), breaking ties arbitrarily.

3.2 Analysis

In this section we prove the following theorem.

Theorem 1: If the sequence of packets is given by an $A(w,\varepsilon)$ adversary, then the number of packets stored at any given time in any of the buffers of protocol BASIC is at most $O(m^{3/2}n^{3/2}w/\varepsilon)$ where $\varepsilon \geq 1/w$ without loss of generality.[2]

An immediate corollary to the above theorem is that the total number of packets stored by BASIC at any given time is at most $M(w,\varepsilon) = O(m^{5/2}n^{5/2}w/\varepsilon)$.

In the following we assume that the sequence of packets is injected by an $A(w,\varepsilon)$ adversary. Note however that the protocol does *not* know the values of w and ε (i.e., these values are not used by the protocol).

Proof: We start with a simple claim about the change in the size of a given buffer.

[2]Note that since $\varepsilon > 0$ and since an $A(w,\varepsilon)$ adversary is limited to injecting $\lfloor (1-\varepsilon)w \rfloor$ packets at each window (where the floor is used since the number of packets is obviously an integer), then $\varepsilon \geq 1/w$ does *not* impose any additional constraint.

Claim 2: For any $t_1 \leq t_2 \leq t_1 + w - 1$, and for all $(u,v) \in E$ and $d \in V$,

$$q^{t_1}_{v,u,d} - w - 1 \quad \leq q^{t'_2}_{v,u,d} \leq \quad q^{t_1}_{v,u,d} + 2w + 1.$$

(We remark that one can incorporate ε into the statement of this claim. This, however, yields only a minor improvement to our bounds.)

Proof: By invariant (1), $\lfloor q^{t_1}_{v,*,d}/\delta_v \rfloor \leq q^{t_1}_{v,u,d} \leq \lfloor q^{t_1}_{v,*,d}/\delta_v \rfloor + 1$ for all $(v,u) \in E$. Consider the number of packets with destination d stored in v at time t'_2, which is denoted $q^{t'_2}_{v,*,d}$. In time window $[t_1, t'_2]$ at most $\delta_v w$ such packets can arrive into node v across the edges. In addition, at most $\delta_v w$ such packets can be injected by the adversary into v (otherwise, the restriction on the adversary is violated). Thus, at most $2w\delta_v$ such packets can be added to the node. On the other hand, at most $\delta_v w$ such packets can be sent out by v in the time window $[t_1, t'_2]$. We get that $q^{t_1}_{v,*,d} - \delta_v w \leq q^{t'_2}_{v,*,d} \leq q^{t_1}_{v,*,d} + 2w\delta_v$. It follows that

$$q^{t'_2}_{v,u,d} \leq \lfloor q^{t'_2}_{v,*,d}/\delta_v \rfloor + 1 \leq \lfloor q^{t_1}_{v,*,d}/\delta_v \rfloor + 2w + 1 \leq q^{t_1}_{v,u,d} + 2w + 1$$

since $\lfloor q^{t_1}_{v,*,d}/\delta_v \rfloor \leq q^{t_1}_{v,u,d}$. Similarly,

$$q^{t_1}_{v,u,d} - w - 1 \leq \lfloor q^{t_1}_{v,*,d}/\delta_v \rfloor - w \leq \lfloor q^{t'_2}_{v,*,d}/\delta_v \rfloor \leq q^{t'_2}_{v,u,d}$$

since $q^{t_1}_{v,u,d} \leq \lfloor q^{t_1}_{v,*,d}/\delta_v \rfloor + 1$. □

The following, more general, claim can be proved by a straightforward modification of the proof above.

Claim 3: For any $t_1 \leq t_2 \leq t_1 + k \cdot w - 1$, for any integer k, and for all $(u,v) \in E$ and $d \in V$,

$$q^{t_1}_{v,u,d} - k \cdot w - 1 \quad \leq q^{t'_2}_{v,u,d} \leq \quad q^{t_1}_{v,u,d} + 2k \cdot w + 1.$$

We now define a potential function Φ on which our proof of the theorem is based. For each buffer of size q, we assume that each packet p in that buffer is assigned a unique *height* $h(p)$ from 1 to q as if the packets are stored one on top of the other. Let the potential of a buffer be the sum of the heights of packets in the buffer. That is,

$$\Phi^t_{v,u,d} \stackrel{\triangle}{=} \sum_{p \in Q^t_{v,u,d}} h(p) = \binom{q^t_{v,u,d} + 1}{2}.$$

Let $\mathcal{P}^t$ be the set of all packets stored in all nodes at time t. The value of Φ at time t is defined by

$$\Phi^t \stackrel{\triangle}{=} \sum_{p \in \mathcal{P}^t} h(p) = \sum_{d \in V} \sum_{e=(v,u) \in E} \left[\Phi^t_{v,u,d} + \Phi^t_{u,v,d} \right].$$

For the purpose of analysis, assume that the packets that are sent in sub-step 1 of the protocol are the packets with maximum height in the corresponding buffers. That is, in

sub-step 1 of time step t, if a packet is sent from $Q_{v,u,d}$ to $Q_{u,v,d}$ then the packet which is taken from $Q_{v,u,d}$ is of height $q^t_{v,u,d}$. Moreover, when such a packet is injected into $Q_{u,v,d}$ it is assigned height $q^t_{u,v,d}+1$. Thus, the potential decreases by $q^t_{v,u,d} - (q^t_{u,v,d}+1) \geq 0$ due to this packet movement.

Observe that the potential function can only change upon the following events: (1) the addition of a new injected packet into a buffer (increases the potential function); (2) the transfer of a packet across an edge (can either decrease the potential function or not change it); (3) the redistribution of packets in buffers in a node (either decreases the potential function or does not change it); and (4) the removal of packets at their destination (decreases the potential function).

We will analyze the behavior of the potential function over any window $\mathcal{W}^t_w$ of w time steps. Therefore, to give an upper bound on the increase of the potential function during $\mathcal{W}^t_w$, we consider the increase in potential due to *all* packets injected in time window $\mathcal{W}^t_w$, and the decrease due to *some* packet transfers during the same time. Below we state the central technical lemma of our analysis which quantifies the potential decrease due to some packet movements on edges along a path from s to D, in terms of the number of packets destined to D, and stored in s.

Lemma 4: Let $e_1, e_2, \ldots, e_\ell$, for $e_i = (v_{i-1}, v_i) \in E$, be a simple path in G. Denote v_ℓ by D. Let $t_1, t_2, \ldots, t_\ell$ and t be any set of times satisfying $t \leq t_i \leq t + w - 1$ for all $1 \leq i \leq \ell$. Let Δ_i, for $1 \leq i \leq \ell$, be the *decrease* in the potential function due to a packet transfer in time step t_i on e_i from v_{i-1} to v_i. Then $\sum_{i=1}^{\ell} \Delta_i \geq q^t_{v_0,u,D} - \ell(3w+4)$, for any $(v_0, u) \in E$.

The lemma implies that if $q^t_{v_0,u,D}$ is "very large" then $\sum \Delta_i$ is "very large" as well.

Proof: First, we consider a single Δ_i. The potential, corresponding to the buffers of destination D, along the edge e_i at time t_i (just before any packet was sent at this time step) is $\Phi^{t_i}_{v_{i-1},v_i,D} + \Phi^{t_i}_{v_i,v_{i-1},D}$. If a packet whose destination is D is indeed sent from v_{i-1} to v_i then the size of $Q_{v_{i-1},v_i,D}$ is decreased by 1 and the size of $Q_{v_i,v_{i-1},D}$ is increased by 1. Therefore, the decrease in the potential caused by such packet transfer would be $q^{t_i}_{v_{i-1},v_i,D} - (q^{t_i}_{v_i,v_{i-1},D}+1)$. By the resolution rule that we use (Step 1 of protocol BASIC), the actual packet that is sent along edge e_i at time t_i is one that corresponds to some destination d that makes this difference maximal (and non-negative). Therefore, the decrease in potential can only be bigger; i.e.,

$$\Delta_i \geq q^{t_i}_{v_{i-1},v_i,D} - (q^{t_i}_{v_i,v_{i-1},D}+1).$$

(In the special case where there is no destination d for which $q^{t_i}_{v_{i-1},v_i,d} - q^{t_i}_{v_i,v_{i-1},d}$ is positive no packet is sent on this edge in this direction at this time step; in such a case $\Delta_i = 0$ and the inequality still holds.) Note that we refer here to the size of buffers at time t_i. However, using Claim 2, we can relate these sizes to the sizes of buffers at time t. That is,

$$\begin{aligned} \Delta_i &\geq (q^t_{v_{i-1},v_i,D} - w - 1)) - (q^t_{v_i,v_{i-1},D} + (2w+1) + 1) \\ &= q^t_{v_{i-1},v_i,D} - q^t_{v_i,v_{i-1},D} - (3w+3)\ . \end{aligned}$$

Therefore,

$$\begin{aligned}\sum_{i=1}^{\ell}\Delta_i &\geq \sum_{i=1}^{\ell}[q^t_{v_{i-1},v_i,D} - q^t_{v_i,v_{i-1},D} - (3w+3)] \\ &= -\ell(3w+3) + q^t_{v_0,v_1,D} - q^t_{v_\ell,v_{\ell-1},D} + \sum_{i=1}^{\ell-1}(-q^t_{v_i,v_{i-1},D} + q^t_{v_i,v_{i+1},D}) \ .\end{aligned}$$

Now, since $v_\ell = D$ then $q^t_{v_\ell,v_{\ell-1},D} = q^t_{D,v_{\ell-1},D} = 0$. In addition, each of the terms $(q^t_{v_i,v_{i+1},D} - q^t_{v_i,v_{i-1},D})$ is at least -1, by invariant (1) (with respect to node v_i), and similarly the difference between $q^t_{v_0,v_1,D}$ and $q^t_{v_0,u,D}$ (for any u adjacent to v_0) is at most 1. Altogether, we get that $\sum_{i=1}^{\ell}\Delta_i \geq q^t_{v_0,u,D} - \ell(3w+4)$. □

The following is our main lemma which implies that once some buffer gets sufficiently large the potential function will not increase over w consecutive time steps.

Lemma 5: Let $q^t_{\max}$ be the size of the largest buffer in the whole network at time t. Then $\Phi^{(t+w-1)'} - \Phi^t \leq c_5 nmw^2 - q^t_{\max}\lceil \varepsilon w \rceil$, for $n \geq 3$, $w \geq 4$, $m \geq 4$, and some constant c_5 (e.g., $c_5 = 11$ suffices).

Remark. Whenever a constant is introduced in a lemma, the subscript of the constant will be equal to the number of the lemma.

Proof: Consider the set of all packets $\mathcal{I}^t$ injected in time window $\mathcal{W}^t_w$, for some time t. For each packet $p_k = (v_k, d_k) \in \mathcal{I}^t$, we count the potential increase due to the injection and identify some packet transfers that will cause a decrease in potential.

We first identify the increases in potential. Each packet $p_k = (v_k, d_k) \in \mathcal{I}^t$ was added at some time step τ_k to a buffer Q_{v_k,u_k,d_k} in node v_k. Let α_k be the potential increase due to the injection of this packet. By our definitions, $\alpha_k \leq q^{\tau'_k}_{v_k,u_k,d_k}$. By Claim 2, $\alpha_k \leq q^t_{v_k,u_k,d_k} + 2w + 1$ for each $p_k \in \mathcal{I}^t$.

Now let us identify some of the decrease in potential. For each packet $p_k = (v_k, d_k) \in \mathcal{I}^t$, consider the path from its injection point, v_k, to its destination d_k, guaranteed by the definition of the $A(w,\varepsilon)$ adversary. Let this path be $\pi_k = e^k_1, e^k_2, \ldots, e^k_{\ell_k}$. The set of paths associated with all the packets of $\mathcal{I}^t$ have, by the definition of the adversary, the property that no edge is used more than $\lfloor (1-\varepsilon)w \rfloor$ times in either direction. This leads to the following claim:

Claim 6: For each path π_k, as above, we can associate a sequence of times $T^k = \{t^k_1, \ldots, t^k_{\ell_k}\}$, where $t \leq t^k_j \leq t + w - 1$ for $1 \leq j \leq \ell_k$ (i.e., a time step for each edge in the path), in a way that for each edge, at most $\lfloor (1-\varepsilon)w \rfloor$ distinct time steps (in each direction) are assigned.[3]

[3] We emphasize that the paths are not given to the protocol and are used only for its analysis. In particular, the protocol is likely to use different paths for the packets. Also, we emphasize that we make no hidden assumptions on the times in T^k other than that no edge is assigned more than $\lfloor (1-\varepsilon)w \rfloor$ time steps (e.g., we do not assume that $t^k_1 \leq t^k_2 \leq \ldots \leq t^k_{\ell_k}$).

Thus, each edge, in each direction, will still have at least $\lceil \varepsilon w \rceil$ "free" time steps during time window $\mathcal{W}_w^t$.

Proof: An assignment can be achieved, for example, in a greedy manner by first making the assignment corresponding to π_1, then the assignment corresponding to π_2, etc. For each edge $e \in E$ and each direction $* \in \{+,-\}$ denote by $S_{e,*}^i$ the set of times assigned to edge e in direction $*$ in the first i steps of this process. That is, $S_{e,*}^0 = \emptyset$, and to define $S_{e,*}^i$ we assign times to the edges of π_i as follows. For each edge e and direction $*$, used in π_i, pick an arbitrary element of $\{t, \ldots, t+w-1\} \setminus S_{e,*}^{i-1}$ and add that element to $S_{e,*}^{i-1}$ to obtain $S_{e,*}^i$. Such an element must exist by the definition of the adversary. If $S_{e,*}$ is the set after all paths have been assigned then, again by the restrictions on the adversary, $|S_{e,*}| \leq \lfloor (1-\varepsilon)w \rfloor$ for each $e, *$. This completes the proof of Claim 6. □

By Lemma 4, for every packet $p_k \in \mathcal{I}^t$, the decrease in the potential function, due to packets transfers along π_k at times T^k is at least $q^t_{v_k,u_k,d_k} - \ell_k(3w+4) \geq q^t_{v_k,u_k,d_k} - n(3w+4)$. As observed above, these paths leave $\lceil \varepsilon w \rceil$ "free" time steps (during time window $\mathcal{W}_w^t$) for each edge. We use this "free" time to get an additional reduction in the potential function. Let v^*, u^*, and d^*, be such that $q^t_{\max} = q^t_{v^*,u^*,d^*}$. Using any simple path leading from v^* to d^* and the "free" time slots available for each of the edges along this path, we have by Lemma 4 that the *decrease* in the potential along this path, at these time steps, is at least

$$\lceil \varepsilon w \rceil (q^t_{\max} - n(3w+4)) \geq \lceil \varepsilon w \rceil q^t_{\max} - nw(3w+4).$$

We can now sum all the increases and decreases identified above.

$$\begin{aligned}
\Phi^{(t+w-1)'} - \Phi^t &\leq \sum_{p_k \in \mathcal{I}^t} \alpha_k - \sum_{p_k \in \mathcal{I}^t} [q^t_{v_k,u_k,d_k} - n(3w+4)] \quad - (\lceil \varepsilon w \rceil q^t_{\max} - nw(3w+4)) \\
&\leq \sum_{p_k \in \mathcal{I}^t} [q^t_{v_k,u_k,d_k} + 2w + 1] - \sum_{p_k \in \mathcal{I}^t} [q^t_{v_k,u_k,d_k} - 4nw] - \lceil \varepsilon w \rceil q^t_{\max} + 4nw^2 \\
&\leq \sum_{p_k \in \mathcal{I}^t} [2(w+1) + 4nw] + 4nw^2 - \lceil \varepsilon w \rceil q^t_{\max} \\
&\leq 2mw \cdot (nw + 4nw) + 4nw^2 - \lceil \varepsilon w \rceil q^t_{\max} \\
&\leq 11mnw^2 - \lceil \varepsilon w \rceil q^t_{max} \ ,
\end{aligned}$$

for $n \geq 3$, $w \geq 4$, $m \geq 4$. Hence, we proved Lemma 5. □

To complete the proof of Theorem 1, we consider times $\tau_i = iw + 1$, for $i \geq 0$. Now consider the value of the potential function at Φ^{τ_i} and at Φ^{τ_i+1}. Let $B = (A+1)/\sqrt{2}$ where $A = c_5 mnw/\varepsilon$. We consider two cases. First, suppose that $\Phi^{\tau_i} \geq 2mn \cdot B^2$. This implies that there is at least one buffer with potential at least B^2. Recall that the height q and potential Φ of a buffer satisfy $q^2/2 < \Phi < (q+1)^2/2$. Thus, the height of this buffer is at least A. By Lemma 5, $\Phi^{\tau_i+1} \leq \Phi^{\tau_i}$. Now suppose that $\Phi^{\tau_i} < 2mn \cdot B^2$. By Lemma 5 (since the buffer size is always non-negative), the increase in the potential function is bounded by $c_5 mnw^2$, and we have that $\Phi^{\tau_i+1} \leq \Phi^{\tau_i} + c_5 mnw^2$.

Since $\Phi^{\tau_0} = 0$ we can conclude that for $\tau_i = iw + 1$, $\Phi^{\tau_i} \leq 2mnB^2 + c_5 mnw^2$. Since $B = O(mnw/\varepsilon)$ and since the height of each buffer, $q^t_{v,u,d}$, is at most $\sqrt{2\Phi^t_{v,u,d}}$, it follows that the height of each buffer at times τ_i is at most $O(m^{3/2}n^{3/2}w/\varepsilon)$. Since by Claim 2 each buffer can increase by at most $2w + 1$ in any w consecutive time steps, we have an upper bound on the height of any buffer, at *any time*, of $O(m^{3/2}n^{3/2}w/\varepsilon)$. This completes the proof of Theorem 1. □

3.3 The Fully Distributed Protocol

The protocol BASIC defined above is not fully distributed. In particular, each node v uses for its decisions the size of the buffers at the other end of its edges. In this section we show how to make the protocol fully distributed. For this, we replace in Step 1 of BASIC the values $q^t_{u,v,d}$ stored in node v (that is, the real buffer sizes in adjacent nodes u) with values $a^t_{u,v,d}$, which will be estimates on $q^t_{u,v,d}$. Furthermore, node u will also hold values $A^t_{u,v,d}$, which the protocol will ensure are kept equal to the values $a^t_{u,v,d}$ at adjacent nodes v. We denote this fully distributed version of BASIC by DBASIC. DBASIC will use the following procedure: every node $v \in V$ will update the estimates at each time step $t \in \mathbb{N}$. Each time step is allocated to one of the n destination in a round robin way. For a time step which is allocated to destination d, for each edge (v, u) from node v, v sends to u the difference between the actual size of the buffer $q^t_{v,u,d}$ and the value of the estimate $A^t_{v,u,d}$, but only up to values in the range $[-4n, +4n]$. Thus, the number of bits that v uses for each such message is only $\lceil \log n \rceil + 4$. For convenience, we number the destinations $d \in V$ by numbers from 0 to $n-1$.

Procedure UPDATE at node v:

1. Let $d = (t-1) \bmod n$.
2. (send updates:) For each $e = (v, u) \in E$, let $b^t_{v,u,d} = q^t_{v,u,d} - A^t_{v,u,d}$. If $b^t_{v,u,d} \leq 0$ let $r^t_{v,u,d} = \max(b^t_{v,u,d}, -4n)$. If $b^t_{v,u,d} > 0$ let $r^t_{v,u,d} = \min(b^t_{v,u,d}, 4n)$. Send $r^t_{v,u,d}$ to u (actually, the value $r^t_{v,u,d}$ is piggybacked on the packet sent from v to u on e at time t). Let $A^{t'}_{v,u,d} \leftarrow A^t_{v,u,d} + r^t_{v,u,d}$.
3. (receive updates:) For each $e = (v, u) \in E$, receive the value $r^t_{u,v,d}$ on edge e. Let $a^{t'}_{u,v,d} \leftarrow a^t_{u,v,d} + r^t_{u,v,d}$.

The next lemma shows that the estimates on the buffer sizes that the nodes hold are always close to the real buffer sizes. Throughout this section we will assume that $w \geq 4n$ (See Section 5 for a discussion on removing this assumption). This also implies that every estimate, corresponding to any destination, is updated at least once (actually at least 4 times) during each window of w time steps. Also note that it is possible that, due to bursty injection of packets, during the n time steps between two updates of $a_{u,v,d}$ the change in the buffer size, $q_{u,v,d}$, will be much larger than $4n$ (which is the maximal quantity by which $a_{u,v,d}$ is changed at a single time step). Yet, the next lemma shows that $a_{u,v,d}$ is not too far from $q_{u,v,d}$ at any time.

Lemma 7: For any t, any $e = (v, u) \in E$, and any $d \in V$, $|q^t_{u,v,d} - a^t_{u,v,d}| \leq c_7 w$, for some constant c_7 (e.g., $c_7 = 16$ suffices).

Proof: We make the following observations about the changes in the values $q_{u,v,d}$ and $a_{u,v,d}$ during w time steps. For any t_1, t_2 such that $|t_1 - t_2| \leq w$, the difference in buffer sizes is $|q^{t_1}_{u,v,d} - q^{t_2}_{u,v,d}| \leq 2w + 1 \leq 3w$, by Claim 2. For the same times t_1, t_2, the difference in the estimated buffer size is $|a^{t_1}_{u,v,d} - a^{t_2}_{u,v,d}| \leq 5w$ (if w is divisible by n it is at most $4w$; otherwise it is $4w + 4n \leq 5w$). With these observations, the proof of the lemma is by induction on t. When the protocol starts we have $q^1_{u,v,d} = a^1_{u,v,d} = 0$, and in the next w steps $q_{u,v,d}$ cannot grow to more than $3w$, and $a_{u,v,d}$ cannot grow to more than $5w$ by the above observations. Thus, the difference between the two is at most $5w$ and the lemma holds for $t \leq w$. For the induction step, consider $t > w$ and let $t^* = t - w$. The induction hypothesis implies that $|q^{t^*}_{u,v,d} - a^{t^*}_{u,v,d}| \leq 16w$. Consider two cases: (a) if $|q^{t^*}_{u,v,d} - a^{t^*}_{u,v,d}| \leq 8w$ then by the observations on the changes in $q_{u,v,d}$ and $a_{u,v,d}$ the difference at time t cannot grow to more than $16w$. (b) if $q^{t^*}_{u,v,d}$ is larger (smaller) than $a^{t^*}_{u,v,d}$ by more than $8w$ then it follows, again by the above observations, that $q_{u,v,d}$ will remain larger (smaller) than $a_{u,v,d}$, by at least $4n$ during each of the next $\lfloor \frac{w}{n} \rfloor \cdot w$ time steps. Therefore, in *all* updates of $a_{u,v,d}$ during this time it will be increased (decreased) by $4n$ and so it will be increased (decreased) by total of at least $4n \cdot \lfloor w/n \rfloor$. Note that this is at least $3w$ for $w \geq 4n$. In the worst case, $q_{u,v,d}$ is increased (decreased) at the same time by at most $3w$, and so the difference at t is at most the difference at t^*. □

We now turn to prove an upper bound on the size of the buffers of protocol DBasic. The proof follows the proof for protocol Basic. The main difference stems from the fact that now the decision on sending a packet is based on the estimates of the sizes of buffers in adjacent nodes, rather than on the real sizes (each node still uses for its own buffers the real size). This may cause a different decision, and may lead to an *increase* in the potential function due to a packet transfer. We first give an upper bound on the possible increase in the potential function due to a packet transfer.

Lemma 8: For any time t and any $e = (v, u) \in E$, the transfer of a packet from v to u can increase the potential function by at most $c_7 w$.

Proof: If a packet is sent from $Q_{v,u,d}$ to $Q_{u,v,d}$ at t then the *increase* in the potential function is exactly $q^t_{u,v,d} + 1 - q^t_{v,u,d}$. For the packet to be sent it must hold that $q^t_{v,u,d} - a^t_{u,v,d} \geq 1$. But $q^t_{u,v,d} - a^t_{u,v,d} \leq c_7 w$, by Lemma 7. It follows that

$$q^t_{u,v,d} + 1 - q^t_{v,u,d} \leq c_7 w + a^t_{u,v,d} + 1 - q^t_{v,u,d} \leq c_7 w \ .$$

□

The following is a version of Lemma 4 that applies to protocol DBasic.

Lemma 9: Let $e_1, e_2, \ldots, e_\ell$, for $e_i = (v_{i-1}, v_i) \in E$, be a simple path in G. Denote v_ℓ by D. Let $t_1, t_2, \ldots, t_\ell$ and t be any set of times satisfying $t \leq t_i \leq t + w - 1$ for all $1 \leq i \leq \ell$.

Let Δ_i, for $1 \leq i \leq \ell$, be the *decrease* in the potential function due to a packet transfer in time step t_i on e_i from v_{i-1} to v_i. Then $\sum_{i=1}^{\ell} \Delta_i \geq q^t_{v_0,u,D} - \ell(c_9 w + 4)$, for any $(v_0, u) \in E$ and some constant c_9 (e.g., $c_9 = 2c_7 + 3$ suffices).

Proof: First, we consider a single Δ_i. Let d be such that a packet with destination d is sent along edge e_i at time t_i (we will deal with the case that no packet is sent later). The potential, corresponding to the buffers of destination d, along the edge e_i at time t_i (just before any packet was sent at this time step) is $\Phi^{t_i}_{v_{i-1},v_i,d} + \Phi^{t_i}_{v_i,v_{i-1},d}$. Therefore, the decrease in the potential caused by such packet transfer would be $q^{t_i}_{v_{i-1},v_i,d} - (q^{t_i}_{v_i,v_{i-1},d} + 1)$. Using Lemma 7, we have

$$\begin{aligned} \Delta_i &= q^{t_i}_{v_{i-1},v_i,d} - (q^{t_i}_{v_i,v_{i-1},d} + 1) \\ &\geq q^{t_i}_{v_{i-1},v_i,d} - (a^{t_i}_{v_i,v_{i-1},d} + c_7 w + 1) \ . \end{aligned}$$

Since the protocol sent a packet with destination d, by the resolution rule it follows that

$$q^{t_i}_{v_{i-1},v_i,d} - a^{t_i}_{v_i,v_{i-1},d} \geq q^{t_i}_{v_{i-1},v_i,D} - a^{t_i}_{v_i,v_{i-1},D} \ ,$$

and using Lemma 7 again we get

$$\begin{aligned} \Delta_i &\geq q^{t_i}_{v_{i-1},v_i,d} - (a^{t_i}_{v_i,v_{i-1},d} + c_7 w + 1) \\ &\geq q^{t_i}_{v_{i-1},v_i,D} - (a^{t_i}_{v_i,v_{i-1},D} + c_7 w + 1) \\ &\geq q^{t_i}_{v_{i-1},v_i,D} - (q^{t_i}_{v_i,v_{i-1},D} + 1) - 2c_7 w \ . \end{aligned}$$

In the special case where no packet is sent, $\Delta_i = 0$. This special case occurs only if for all d, $q^{t_i}_{v_{i-1},v_i,d} - a^{t_i}_{v_i,v_{i-1},d} \leq 0$. It follows that that for all d (including D)

$$\begin{aligned} q^{t_i}_{v_{i-1},v_i,d} - (q^{t_i}_{v_i,v_{i-1},d} + 1) - 2c_7 w &\leq q^{t_i}_{v_{i-1},v_i,d} - (a^{t_i}_{v_i,v_{i-1},d} - c_7 w + 1) - 2c_7 w \\ &= q^{t_i}_{v_{i-1},v_i,d} - a^{t_i}_{v_i,v_{i-1},d} - 1 - c_7 w \leq 0 \ . \end{aligned}$$

Thus the lower bound on Δ_i holds also in this case.

Note that we refer in the above to the size of buffers at time t_i. However, using Claim 2, we can relate these sizes to the sizes of buffers at time t. That is,

$$\begin{aligned} \Delta_i &\geq (q^t_{v_{i-1},v_i,D} - w - 1)) - (q^t_{v_i,v_{i-1},D} + (2w + 1) + 1) - 2c_7 w \\ &= q^t_{v_{i-1},v_i,D} - q^t_{v_i,v_{i-1},D} - ((2c_7 + 3)w + 3) \ . \end{aligned}$$

Therefore,

$$\begin{aligned} \sum_{i=1}^{\ell} \Delta_i &\geq \sum_{i=1}^{\ell} [q^t_{v_{i-1},v_i,D} - q^t_{v_i,v_{i-1},D} - ((2c_7 + 3)w + 3)] \\ &= -\ell((2c_7 + 3)w + 3) + q^t_{v_0,v_1,D} - q^t_{v_\ell,v_{\ell-1},D} + \sum_{i=1}^{\ell-1} (-q^t_{v_i,v_{i-1},D} + q^t_{v_i,v_{i+1},D}) \end{aligned}$$

Now since $v_\ell = D$ then $q^t_{v_\ell,v_{\ell-1},D} = q^t_{D,v_{\ell-1},D} = 0$. In addition, each of the terms $(q^t_{v_i,v_{i+1},D} - q^t_{v_i,v_{i-1},D})$ is at least -1, by invariant (1) (with respect to node v_{i+1}), and similarly the difference between $q^t_{v_0,v_1,D}$ and $q^t_{v_0,u,D}$ (for any u adjacent to v_0) is at most 1. Altogether, we get that $\sum_{i=1}^{\ell} \Delta_i \geq q^t_{v_0,u,D} - \ell((2c_7+3)w+4)$. □

We now give an analogue of Lemma 5 that applies to protocol DBasic.

Lemma 10: Let $q^t_{\max}$ be the size of the maximal buffer in the whole network at time t. Then, $\Phi^{(t+w-1)'} - \Phi^t \leq c_{10}mnw^2 - q^t_{\max}\lceil \varepsilon w \rceil$, for $n \geq 3$, $w \geq 4$, $m \geq 4$, and some constant c_{10} (e.g., $c_{10} = 3c_9 + 2c_7 + 5$ suffices).

Proof: We follow and modify the proof of Lemma 5. The difference between the proof of the present lemma and the proof of Lemma 5 is that in the present case we cannot ignore any packet transfers, because packet transfers may *increase* the potential function (since the decision whether to send a packet is made based on estimates rather than the actual buffer sizes which are not known). However, by Lemma 8, we have an upper bound on this increase.

Let p_k, α_k, π_k and T^k be as in the proof of Lemma 5. By Lemma 9, the decrease in the potential function due to packet transfers along π_k at times T^k is at least $q^t_{v_k,u_k,d_k} - \ell(c_9w + 4) \geq q^t_{v_k,u_k,d_k} - n(c_9w+4)$. Using the $\lceil \varepsilon w \rceil$ "free" time steps we have another decrease of at least

$$\lceil \varepsilon w \rceil (q^t_{\max} - n(c_9w+4)) \geq \lceil \varepsilon w \rceil q^t_{\max} - nw(c_9w+4).$$

The increase in the potential function occurs for injection of packets, and may also occur for packet transfers. The increase due to packet injections is $\sum_{p_k \in \mathcal{I}^t} \alpha_k$, while the increase due to packet transfers is upper bounded by $2mw \cdot c_7 w$. Summing up we have

$$\begin{aligned}
&\Phi^{(t+w-1)'} - \Phi^t \\
&\leq \sum_{p_k \in \mathcal{I}^t} \alpha_k + 2mw \cdot c_7 w \\
&\quad - \sum_{p_k \in \mathcal{I}^t} [q^t_{v_k,u_k,d_k} - n(c_9w+4)] - (\lceil \varepsilon w \rceil q^t_{\max} - nw(c_9w+4)) \\
&\leq \sum_{p_k \in \mathcal{I}^t} [q^t_{v_k,u_k,d_k} + 2w + 1] + 2c_7mw^2 \\
&\quad - \sum_{p_k \in \mathcal{I}^t} [q^t_{v_k,u_k,d_k} - (c_9+1)nw] - \lceil \varepsilon w \rceil q^t_{\max} + (c_9+1)nw^2 \\
&\leq \sum_{p_k \in \mathcal{I}^t} [2(w+1) + (c_9+1)nw] + 2c_7mw^2 + (c_9+1)nw^2 - \lceil \varepsilon w \rceil q^t_{\max} \\
&\leq 2mw(nw + (c_9+1)nw) + 2c_7mw^2 + (c_9+1)nw^2 - \lceil \varepsilon w \rceil q^t_{\max} \\
&\leq (3c_9 + 2c_7 + 5)mnw^2 - \lceil \varepsilon w \rceil q^t_{\max} \, .
\end{aligned}$$

The lemma follows. □

Finally, to give an upper bound on the size of any buffer, we use the above lemma and the same arguments as those in the proof of Theorem 1. We get a bound of $O(m^{3/2}n^{3/2}w/\varepsilon)$ on the size of any buffer.

4 A Protocol with a Bound on Delivery Time

Protocol DBasic of the previous section guarantees bounded buffers as long as the packets are injected by some adversary $A(w, \varepsilon)$. However, this protocol does not guarantee a bound on the delivery time of the packets. Indeed, some packets may get stuck in the network forever. In this section we extend the protocol so as to guarantee that each packet is delivered within a (polynomially) bounded number of time steps. We will assume for now that $w \geq 4n$ (In Section 5 we discuss removing this assumption).

4.1 The Case where w is Known

First we define a protocol that provides such guarantees on the delivery time, but has to know an upper bound W on the value w according to which the adversary injects the packets. In Section 4.2 we eliminate this requirement.

The main idea is as follows. We will run, in parallel to the main protocol DBasic, another "underground" drainage protocol. This protocol gets at certain time steps *all* the packets stored in the buffers of the main protocol, and will be responsible for their delivery. The buffers of the main protocol become empty at such an event. The drainage protocol receives more packets from the main protocol only after all its buffers become empty (that is, all its packets are delivered). The advantage of the drainage protocol is that it receives only a bounded amount of packets (since the buffers of the main protocol are bounded), and all of them at the same time, and thus can deliver the packets in some bounded amount of time. To enable the drainage protocol to operate we have to assign it, from time to time, some time steps in which it can move its packets across edges. We will assign the drainage protocol one time step every $2W$ time steps. The main point to be proved in this section is that in spite of the fact that some of the time steps are allocated to the drainage protocol, the main protocol, Basic, still maintains bounded buffers. The line of proof will be similar to the line of proof of the previous section, modified in order to take into account the fact that some of the time steps are allocated to the drainage protocol.

The Drainage Protocol The drainage protocol is a protocol that is injected *once* with packets at the nodes. If the total number of packets injected is M, then all packets are delivered within $T(M)$ time steps (where we count only time steps in which the drainage protocol operates). Clearly there are such protocols, the simplest of them, maybe, being the one that routes any packet with destination d along a shortest path to d, with some arbitrary greedy congestion resolution rule. By the results of Mansour and Patt-Shamir [MP], for such protocols $T(M) \leq M + n$. Obviously, the size of buffers used by such a protocol is bounded by M.

We also devise a procedure called Update that will allow all processors to know if the drainage protocol has delivered all its packets.[4] To do that, we send at every time step an

[4] Alternatively, the processors can simply wait "enough time" (i.e., $T(M)$ steps, for the largest possible M according to the bound on the buffer size). However, since we will later need a notification mechanism

additional bit across any edge (piggybacked on a packet if such is sent). We partition the time into windows of size $2W$ and we want that at the beginning of each such window all processors will be synchronized as to this information. The mechanism works as follows. At the first time step of each such window, each processor checks if its buffers of the drainage protocol are empty or not. It sets a flag `busy` to 0 or 1 accordingly. Then each processor sends the value of this flag along all of its edges. When such a bit is received by a processor the local flag is ORed with the bit received. The new value of the flag is then sent in the next time steps. Since we assume that $W \geq 4n$, at the beginning of the first time step of the next window the value of *every* flag is 1 if and only if there is at least one processor that had packets in the "drainage" buffers exactly $2W$ time steps before.

BOUNDEDDT(W) Our protocol maintains in each node the flag `busy`, all the buffers of protocol DBASIC, and in addition all the buffers and variables that the drainage protocol maintains. The behavior of the protocol is controlled by a variable `w` which (in the current subsection) is set to W, the upper bound that we have on w. We define the protocol below using the drainage protocol, protocol DBASIC and procedure UPDATE as black boxes: at each step either protocol DBASIC or the drainage protocol will assume control in *all* processors, and will send packets according to their own separate decisions. In addition, procedure UPDATE is performed in parallel (whether DBASIC or the drainage protocol assume control), to update the estimates that DBASIC uses in the nodes. The control bits to be sent by this procedure are piggybacked on the packets sent, whether they are sent by DBASIC or the drainage protocol.

Protocol BOUNDEDDT(W):
Initially, in all nodes the flag `busy` is set to 0, and the variable `w` to W. Then, the following is performed by every node $v \in V$, at each time step $t \geq 1$.

1. If $t = 2i\mathtt{w} + 1$ for some i:

   - If `busy` $= 0$ then empty *all* buffers of DBASIC and move the packets as input to the drainage protocol in node v; if `busy` $= 1$ and the buffers of the drainage protocol (in v) are empty set `busy` to 0; otherwise (if `busy` $= 1$ but the buffers of the drainage protocol are not empty) set `busy` to 1.
   - Allow the drainage protocol to operate for one time step (that is, send and accept packets according to its decisions).

   Otherwise (i.e., $t \neq 2i\mathtt{w} + 1$ for any i), run protocol DBASIC (that is, send and accept packets according to the decisions of DBASIC).

2. For every t, send the value of `busy` on all outgoing edges (this bit is to be piggybacked on packets if such are sent according to Step 1).
   OR the flag `busy` with all the values for `busy` received from the adjacent nodes.

for other purposes, we introduce it here.

3. For every t apply procedure UPDATE (the bits sent are piggybacked on packets if such are sent in Step 1; the bits received are received during the same step).

We now prove that this protocol has bounded buffers and, at the same time, guarantees delivery in a bounded amount of time. We prove the following theorem.

Theorem 11: If the packets are injected by an $A(w, \varepsilon)$ adversary, for any $\varepsilon > 0$, and any integer $w \leq W$, then the total number of packets stored by BOUNDEDDT(W) at any time is at most $O(m^{5/2}n^{5/2}w/\varepsilon)$. In addition, each packet is delivered at most $O(m^{5/2}n^{5/2}wW/\varepsilon) = O(m^{5/2}n^{5/2}W^2/\varepsilon)$ times steps after its injection.

(Note that the bound on the number of packets is in terms of the actual adversary parameter w, while the bound on the delivery time is in terms of the upper bound W.) To prove the above theorem, we use the following two lemmas. The first lemma gives a bound on the size of any buffer of DBASIC maintained by BOUNDEDDT(W). The second lemma gives a bound on the delivery time of any packet. Note that a bound on the size of any buffer of DBASIC implies a bound on the total number of packets DBASIC holds, and therefore also a bound on the number of packets the drainage protocol holds. This implies a bound on the total number of packets BOUNDEDDT(W) stores.

Lemma 12: If the sequence of packets is given by an $A(w, \varepsilon)$ adversary, for $w \leq W$, then the number of packets stored at any given time in any buffer of DBASIC maintained by BOUNDEDDT(W), is at most $O(m^{3/2}n^{3/2}w/\varepsilon)$.

Lemma 13: If the sequence of packets is given by an adversary $A(w, \varepsilon)$, for $w \leq W$, then each packet is delivered by BOUNDEDDT(W) in at most $O(m^{5/2}n^{5/2}W^2/\varepsilon)$ time steps.

To prove the above two lemmas, we first note that Claim 2 and Lemmas 7 and 8, hold for the present protocol as well since they are only based on the limitations on the adversary and the fact that the buffers of DBASIC in each node are kept balanced. We give below a version of Lemma 9 applicable to the present protocol. The modification is in the conditions about the times set for the edges of the path. First, they all have to be time steps in which DBASIC has control of the network (i.e., we have to show that the time steps allocated to the drainage protocol do not significantly disturb the main protocol). Secondly, the time steps span over a period of $2w$ time steps rather than w time steps.

Lemma 14: Let $e_1, e_2, \ldots, e_\ell$, for $e_i = (v_{i-1}, v_i) \in E$, be a simple path in G. Denote v_ℓ by D. Let $t_1, t_2, \ldots, t_\ell$ and t be any set of times satisfying $t \leq t_i \leq t + 2w - 1$ for all $1 \leq i \leq \ell$, and such that for every t_i DBASIC has control of the network at time step t_i. Let Δ_i, for $1 \leq i \leq \ell$, be the *decrease* in the potential function due to a packet transfer in time step t_i on e_i from v_{i-1} to v_i. Then $\sum_{i=1}^{\ell} \Delta_i \geq q^t_{v_0,u,D} - \ell(c_{14}w + 4)$, for any $(v_0, u) \in E$, and some constant c_{14} (e.g, $c_{14} = 2c_7 + 6$ suffices).

Proof: The proof is identical to the proof of Lemma 9 up to the point where we relate the size of the buffers at t_i to their size at time t. That is, as in the proof of Lemma 9 we get that

$$\Delta_i \geq q^{t_i}_{v_{i-1},v_i,D} - (q^{t_i}_{v_i,v_{i-1},D} + 1) - 2c_7 w \ .$$

Now, using Claim 3, we get that

$$\begin{aligned} \Delta_i &\geq (q^{t}_{v_{i-1},v_i,D} - 2w - 1)) - (q^{t}_{v_i,v_{i-1},D} + (4w+1) + 1) - 2c_7 w \\ &= q^{t}_{v_{i-1},v_i,D} - q^{t}_{v_i,v_{i-1},D} - ((2c_7+6)w + 3) \ . \end{aligned}$$

Continuing the same calculations as in the proof of Lemma 9 with the above bound we get the required result. □

We give now a version of Lemma 10 (which in turn is a version on Lemma 5) applicable for BOUNDEDDT(W). The difference in our case is that, from time to time, protocol DBASIC does not have control of the network when run under BOUNDEDDT(W) (i.e., in those time steps where the drainage protocol is active). However, this happens only at most once every $2w$ time steps. Therefore, we prove the following lemma for windows of time of size $2w$ rather than w. We use the same potential function Φ defined in the previous section.

Lemma 15: Assume that the sequence of packets is given by a $A(w,\varepsilon)$ adversary, and assume that DBASIC is run, such that it is denied control of the network at most once in every $2w$ time steps. Let $q^t_{\max}$ be the size of the maximal buffer in the whole network at time t, Then $\Phi^{(t+2w-1)'} - \Phi^t \leq c_{15}mnw^2 - q^t_{\max}(2\lceil \varepsilon w \rceil - 1)$, for $n \geq 3$, $w \geq 4$, $m \geq 4$, and some constant c_{15} (e.g., $c_{15} = 6c_{14} + 4c_7 + 13$ suffices).

Proof: Denote by $\mathcal{I}^t$ the set of all packets injected in time window $\mathcal{W}^t_{2w}$. Denote by τ_k the time step at which packet $p_k = (v_k, d_k) \in \mathcal{I}^t$ is injected, and let Q_{v_k,u_k,d_k} be the buffer in node v_k to which p_k was injected. Denote by $\pi_k = e^k_1, \ldots, e^k_{\ell_k}$ the path guaranteed by the adversary for packet p_k.

We first identify for each packet the increase in the potential function due to its injection. Let α_k be the increase in the potential due to the injection of packet $p_k \in \mathcal{I}^t$. Clearly $\alpha_k \leq q^{\tau'_k}_{v_k,u_k,d_k}$. Using Claim 3, we have that $\alpha_k \leq q^t_{v_k,u_k,d_k} + 4w + 1$. Next, we argue about the decrease in the potential associated with the paths guaranteed by the adversary. Since in the window $\mathcal{W}^t_{2w}$ at most one time steps is not under the control of DBASIC, there are at least $2w-1$ time steps in $\mathcal{W}^t_{2w}$ is which DBASIC has control of the network. Thus we apply a procedure similar to the one of Claim 6 to assign time sets to the paths of the packets, guaranteed by the adversary. We can do that in a way that all time steps assigned are such that DBASIC has control of the network, and there will still be at least $2\lceil \varepsilon w \rceil - 1$ "free" time steps in which DBASIC has control of the network.

In all, we assign times to all paths, and have $2\lceil \varepsilon w \rceil - 1 \geq 1$ "free" time steps. Now, using Lemma 14, the decrease in the potential function due to the path of each packet is at least

$$q^t_{v_k,u_k,d_k} - \ell(c_{14}w + 4) \geq q^t_{v_k,u_k,d_k} - n(c_{14}w + 4) \ .$$

From the "free" time steps, we get another reduction of

$$(2\lceil \varepsilon w \rceil - 1)(q^t_{\max} - n(c_{14}w + 4)) \geq (2\lceil \varepsilon w \rceil - 1)q^t_{\max} - 2nw(c_{14}w + 4).$$

The increase in the potential function occurs for injection of packets, and may also occur for packet transfers. The increase due to packet injections is $\sum_{p_k \in \mathcal{I}^t} \alpha_k$, while the increase due to packet transfers is upper bounded by $2m2w \cdot c_7 w$. Summing up we have

$$\begin{aligned}
&\Phi^{(t+w-1)'} - \Phi^t \\
&\leq \sum_{p_k \in \mathcal{I}^t} \alpha_k + 2m2w \cdot c_7 w \\
&\quad - \sum_{p_k \in \mathcal{I}^t} [q^t_{v_k,u_k,d_k} - n(c_{14}w + 4)] - ((2\lceil \varepsilon w \rceil - 1)q^t_{\max} - 2nw(c_{14}w + 4)) \\
&\leq \sum_{p_k \in \mathcal{I}^t} [q^t_{v_k,u_k,d_k} + 4w + 1] + 4c_7 mw^2 \\
&\quad - \sum_{p_k \in \mathcal{I}^t} [q^t_{v_k,u_k,d_k} - (c_{14} + 1)nw] - (2\lceil \varepsilon w \rceil - 1)q^t_{\max} + 2(c_{14} + 1)nw^2 \\
&\leq \sum_{p_k \in \mathcal{I}^t} [4(w + 1) + (c_{14} + 1)nw] + 4c_7 mw^2 + 2(c_{14} + 1)nw^2 - (2\lceil \varepsilon w \rceil - 1)q^t_{\max} \\
&\leq 2m2w(2nw + (c_{14} + 1)nw) + 4c_7 mw^2 + (2c_{14} + 1)nw^2 - (2\lceil \varepsilon w \rceil - 1)q^t_{\max} \\
&\leq (6c_{14} + 4c_7 + 13) \cdot mnw^2 - (2\lceil \varepsilon w \rceil - 1)q^t_{\max} \ .
\end{aligned}$$

□

Observe that BOUNDEDDT(W), with $W \geq w$, satisfied the conditions of the above lemma, and thus the lemma applied to this protocol.

Proof of Lemma 12. We consider times $\tau_i = 2i \cdot w + 1$, for $i \geq 0$. Now, consider the value of the potential function at Φ^{τ_i} and at Φ^{τ_i+1}. Let $B = (A + 1)/\sqrt{2}$ where $A = c_{15}mnw/\varepsilon$. There are two cases. First, suppose that $\Phi^{\tau_i} \geq 2mn \cdot B^2$. This implies that there is at least one buffer with potential at least B^2. Recall that the height q and potential Φ of a buffer satisfy $q^2/2 < \Phi < (q + 1)^2/2$. Thus, the height of this buffer is at least A. By Lemma 15, $\Phi^{\tau_i+1} \leq \Phi^{\tau_i}$. Now suppose that $\Phi^{\tau_i} < 2mn \cdot B^2$. By Lemma 15 (since the buffer size is always non-negative), the increase in the potential function is bounded by $c_{15}mnw^2$, and we have that $\Phi^{\tau_i+1} \leq \Phi^{\tau_i} + c_{15}mnw^2$.

Since $\Phi^{\tau_0} = 0$ we can conclude that for $\tau_i = iw + 1$, $\Phi^{\tau_i} \leq 2mnB^2 + c_{15}mnw^2$. Since $B = O(mnw/\varepsilon)$ and since the height of each buffer, $q^t_{v,u,d}$, is at most $\sqrt{2\Phi^t_{v,u,d}}$, it follows that the height of each buffer at times τ_i is at most $O(m^{3/2}n^{3/2}w/\varepsilon)$. Since, by Claim 3, the size of each buffer can grow by at most $4w + 1$ in any $2w$ consecutive time steps, we have an upper bound on the height of any buffer, at *any time*, of $O(m^{3/2}n^{3/2}w/\varepsilon)$. □

A corollary of Lemma 12 is that the total number of packets stored at any time by the buffers of DBASIC is at most $O(m^{5/2}n^{5/2}w/\varepsilon)$. This implies that the total number of packets stored by the drainage protocol at any time is the same, and the total number of

packets stored by BOUNDEDDT(W) at any given time is at most $M'(w,\varepsilon) = O(M(w,\varepsilon)) = O(m^{5/2}n^{5/2}w/\varepsilon)$.

Proof of Lemma 13. First, note that once a packet is moved to the drainage protocol, the number of time steps until it is delivered is at most $(M'(w,\varepsilon)+n)\cdot 2W$. This is because at most $M'(w,\varepsilon)$ packets are moved to the empty buffers of the drainage protocol, which thus needs at most $M'(w,\varepsilon)+n$ time steps *in which it has control* to deliver all packets [MP]. However, the drainage protocol has control of the network once every $2W$ time steps. By the same argument, the buffers of the drainage protocol get emptied at most $(M'(w,\varepsilon)+n)\cdot 2W$ time steps after packets have been moved to them. Thus a packet, if not delivered earlier by DBASIC, will be transferred to the drainage protocol at most $(M'(w,\varepsilon)+n)\cdot 2W + 2W$ time steps after it is injected (the additional $2W$ time steps come from the fact that it takes the network time to "realize" that the buffers are empty). Altogether, we get that a packet can spend in the network at most

$$2((M'(w,\varepsilon)+n)\cdot 2W) + 2W = O(m^{5/2}n^{5/2}wW/\varepsilon) = O(m^{5/2}n^{5/2}W^2/\varepsilon)$$

time steps, as needed. □

Both Lemmas and the corollary above imply Theorem 11.

4.2 The Case where w is Unknown

We now consider the general case of a protocol that guarantees a bound on the delivery time, and without the protocol having any information on the window size, w, that defines the adversary. The high-level structure of this protocol, BOUNDEDDT, is to have an estimate of the window size, `w` (a variable stored by each processor), and run BOUNDEDDT(W) until one of the processors realizes that the estimate is too low with respect to the real value w according to which the adversary injects the packets. Then, the estimate is doubled and the new version of the protocol is run. Each node maintains a variable `maxbuf` which holds the maximum size of a buffer it has ever locally seen. This value is used to decide if a processor is "happy" with the estimate `w` or not. At the first time step of every window of 2`w` time steps, each processor sets its local `happy` flag according to the current estimate and its variable `maxbuf`. Then, a procedure similar to the one used for the `busy` flag is used, in a way that after 2`w` time steps *all* the processors are "not happy" with the current estimate if and only if there was at least one processor that was not happy. Then, each unhappy processor doubles its estimate. We show below that the buffers remain (polynomially) bounded, and that there exists an upper bound on the delivery time, in spite of the fact that the estimate `w` may at times be incorrect.

We first formally define protocol BOUNDEDDT. It maintains in each node a variable `maxbuf` that will hold the maximum buffers size (of DBASIC) that was ever seen at the node, and a flag `happy`, which will indicate if the estimate `w` of w is still in accordance with the variable `maxbuf`. The protocol is defined using two black-boxes: procedure ESTIMATE_W,

and protocol BOUNDEDDT(W) of Section 4.1. We first define the procedure ESTIMATE_W. Note that this procedure updates the variable `w` used by BOUNDEDDT(W). Define $f(w) \stackrel{\triangle}{=} m^{3/2}n^{3/2}w^2$.

Procedure ESTIMATE_W:
Initially in all nodes $v \in V$ the variable `maxbuf` is set to 0, the flag `happy` is set to 1 and the variable `w` is set to $2^{\lceil \log n \rceil + 2} \geq 4n$. Then, the following is executed in any node $v \in V$ and for any time step $t \in \mathbb{N}$.

1. If $t = 2i\mathtt{w} + 1$ for some i and `happy` is 0, then set `w` to 2`w`. If `maxbuf` $> f($`w`$)$ then set `happy` to 0; otherwise set `happy` to 1.
2. For every $e = (v, u) \in E$ and any $d \in V$, set `maxbuf` to $\max($`maxbuf`$, q^t_{v,u,d})$.
3. Send the flag `happy` to all adjacent nodes (this bit is piggybacked on packets if sent). Receive all `happy` flags from adjacent nodes. AND all received bits with `happy`.

We now define the protocol BOUNDEDDT, using the above procedure and BOUNDEDDT(W) as black-boxes. Note that the procedure ESTIMATE_W updates the variable `w` that controls the behavior of BOUNDEDDT(W). However this is done only at intervals of at least 2`w` time steps, for the current value of `w`.

Protocol BOUNDEDDT:
Initiate all variables for ESTIMATE_W, and BOUNDEDDT(W).
For every $t \in \mathbb{N}$ run both BOUNDEDDT(W) and ESTIMATE_W.

In the following we prove a bound on the size of the buffers of the combined protocol, and a bound on the delivery time of any packet. These bounds are in terms of the parameter w which defines the adversary. The protocol does not have any knowledge about this parameter (not even an upper bound). Note that if we work with a too-low estimate this will cause the drainage protocol to be activated "too often" which may cause the buffers of DBASIC to overflow (since DBASIC will be denied control "too often"). If we work with a too-high estimate, then the drainage protocol will not be given control often enough, and we will not be able to guarantee delivery time in terms of the real parameter w. In the following we prove that this cannot happen: we can have both bounded buffers, and bounded delivery time.

The following lemma provides a bound on the the size of any buffer of DBASIC when run under BOUNDEDDT.

Lemma 16: If the sequence of packets is given by an $A(w, \varepsilon)$ adversary, for any $w > 1$, and any $\varepsilon > 0$, then the number of packets stored at any given time in any buffer of DBASIC run under BOUNDEDDT is at most $O(m^2n^2w^2) = c_{16}m^2n^2w^2$, for some constant c_{16}.

Proof: Let j be such that $2^{j-1} \leq w < 2^j$. Observe that when the value of `w` is changed the main difference in the behavior of the protocol is in how often does the drainage protocol get control of the network. If no buffer ever exceeds $f(2^{j-1}) \leq f(w) = O(m^{3/2}n^{3/2}w^2)$, then the lemma holds. Now let t be the first time that in some processor v some buffer exceeds $f(2^{j-1})$. By the end of this time step this buffer is at most $f(2^{j-1}) + 2w$. Assume that, at time t, the current estimate on w used by the processors is $w_i = 2^i$ (i.e., the value of `w` at t is w_i). Thus, by time step $t + 2w_i$ processor v will set its `happy` flag to false, by procedure ESTIMATE_W (Recall that the initial value of `w` is $w_{\lceil \log n \rceil + 2} = 2^{\lceil \log n \rceil + 2} \geq 4n$, and we assume that $w \geq 4n$). The `happy` flag will continue to be false as long as the estimate is not increased to 2^j (possibly more if the size of the buffers grows even higher). After $j - i$ doublings of the variable w occur by procedure ESTIMATE_W, the value of w reaches $w_j = 2^j$. This will take at most $(5/2)w_j \leq 5w$ time steps ($2w_i$ steps until the `happy` flag is set to false, $2w_i$ until the first doubling, $2w_{i+1}$ until the second doubling and so on, where finally we need $2w_{j-1}$ for the last doubling). During these time steps any buffer can grow by at most $11w$, by Claim 3. We get that if the size of any buffer becomes bigger than $f(2^{j-1})$ then, after at most $5w$ time steps, the value of `w` in all nodes reaches 2^j, and that up until this time all buffers are bounded by $f(2^{j-1}) + 13w = O(m^{3/2}n^{3/2}w^2)$. Thus, we consider now times after the estimate (the variable `w`) reaches 2^j. To prove that the lemma holds after this time, we use arguments similar to those used for BOUNDEDDT(W). Note that the difference here is in two points only. First, the buffers do not start empty, and secondly, the parameter `w` could grow over time.

Let t^* be the time at which the estimate reaches 2^j. We consider times $t_i = t^* + i2w + 1$. Since for all times t^* and later the variable `w` is at least $2^j > w$, we have the DBASIC, run under BOUNDEDDT, is denied control at most once in every $2w$ time steps. Thus Lemma 15 holds for any time $t \geq t^*$. We can thus repeat the arguments of the proof of Lemma 12 to get an upper bound on the size of any buffer, where the modification is in that the buffers of DBASIC do not start empty at time t^* (but rather with size at most $O(m^{3/2}n^{3/2}w^2)$).

Since the size of the buffers at t^* is at most $O(m^{3/2}n^{3/2}w^2)$, the value of the potential function at this time is at most $O(2mn(m^{3/2}n^{3/2}w^2)^2 = O(m^4n^4w^4)$. Using Lemma 15 and arguments as those in the proof of Lemma 12, we have that the value of the potential function at times $t_i = t^* + i2w + 1$ is bounded by $O(m^4n^4w^4)$. This means that the height of any buffer will not exceed $O(m^2n^2w^2)$, at times t_i. Using Claim 3, between these times any buffer can grow by at most another $4w + 1$, which gives us an upper bound of $O(m^2n^2w^2)$ on the size of any buffer at any time. Observe that this bound holds even if the estimate variable `w` grows above 2^j. □

We now give an upper bound on the delivery time of any packet.

Lemma 17: If the sequence of packets is given by an adversary $A(w, \varepsilon)$, then each packet is delivered by BOUNDEDDT in at most $O(m^{13/4}n^{13/4}w^3)$ time steps.

Proof: By Lemma 16, we know that each buffer of DBASIC never holds more than $c_{16}n^2m^2w^2$ packets at any given time. This yields $2mn \cdot c_{16}m^2n^2w^2 = O(m^3n^3w^2)$ packets altogether. Observe that we can express this bound on the size of any single buffer as

$f(\sqrt{c_{16}} \cdot m^{1/4}n^{1/4}w) = c_{16}n^2m^2w^2$. Therefore, the estimate that the protocol uses on w (i.e., the variables `w`) will never exceed $2 \cdot \sqrt{c_{16}} \cdot m^{1/4}n^{1/4}w$. This means that, every $4\sqrt{c_{16}} \cdot m^{1/4}n^{1/4}w$ time steps, or more frequently, the drainage protocol is given control of the network. Thus, every packet will be delivered within $(O(m^3n^3w^2)+n) \cdot 4\sqrt{c_{16}}m^{1/4}n^{1/4}w^2 = O(m^{13/4}n^{13/4}w^3)$ time steps. □

We conclude with the following theorem.

Theorem 18: If the sequence of packets is given by an $A(w, \varepsilon)$ adversary, then the total number of packets stored by BOUNDEDDT at any given time is at most $\mathcal{M}(w, \varepsilon) = O(m^3n^3w^2)$. Every packet is delivered to its destination in at most $\mathcal{T}(w, \varepsilon) = O(m^{13/4}n^{13/4}w^3)$ time steps.

Remark: Note that our bound for the final protocol are *not* in terms of ε. The reason is that our protocol estimates the adversary's parameters, and does that by estimating w, and assuming the smallest possible ε, namely $\varepsilon = 1/w$.

5 Discussion and Extensions

In this section we discuss the assumptions made in the proof and some extensions of our results.

5.1 Removing Assumptions

In the statement of results and in the proofs of our protocols we have made a number of assumptions on the parameters n, m and w. Namely, we have assumed that $n \geq 3$, $m \geq 4$ and $w \geq 4$, for BASIC and that $n \geq 3$, $m \geq 4$ and $w \geq 4n$ for DBASIC and BOUNDEDDT. We now argue that our results hold with only small modifications even when these assumptions do not hold.

We first deal with the assumptions $n \geq 3$, $m \geq 4$ and $w \geq 4$. In the proofs these assumptions were used to bound the change in potential drop in a window of size w by the simple expression $c \cdot mnw^2 - \lceil \varepsilon w \rceil q^t_{\max}$ for a constant c. If any of these assumptions do not hold, the proofs still go through with the same expression, but with larger constants. Another way to view the case when n and m are small is to add an additional connected component of $n = 3$ nodes and $m = 4$ edges. Nothing in our proofs assumes that the network is connected, and thus the proofs will go through for this modified network and the original sequence of packets. This shows that the results hold, with larger constants, even if n and m are small. For the case that w is small, we note that any sequence of packets given by an $A(w, \varepsilon)$ adversary, can also be given by an $A(k \cdot w, \varepsilon)$ adversary for any integer k. Therefore, when $w < 4$ we can assume that the sequence of packets is given by an $A(4w, \varepsilon)$ adversary rather than an $A(w, \varepsilon)$ adversary. The results thus hold with a larger constant.

The assumption $w \geq 4n$ is used in the proof of Lemma 7, and in the proof of Lemma 16. If this assumption does not hold we proceed as follows. If $w < 4n$, we consider the sequence

as being given by an $A(w', \varepsilon)$ adversary, for $w' = \lceil \frac{4n}{w} \rceil w$. For this adversary the assumption clearly holds. The results of Lemma 7 are now $c \cdot w' = c \cdot \lceil \frac{4n}{w} \rceil w \leq c \cdot ((4n/w + 1)w) = c \cdot (4n + w) \leq 2c \cdot \max(4n, w)$. The same holds for Lemma 16. That is, our final results, without any assumption, should be in terms of $w^* \stackrel{\triangle}{=} \max(4n, w)$ rather than w, and with a larger constant. The protocols BOUNDEDDT(W) and BOUNDEDDT can be modified to work in the case that $\mathtt{w} \leq 4n$ but w^* will remain a factor in the bounds.

5.2 Larger Capacity

The results given above are for networks composed of bidirectional edges with unit capacities. However, they can be extended to networks composed of bidirectional edges with integer capacities. We define a c-capacity edge to be an edge which can deliver c packets in each direction in each time step. A simple method of extending the results to capacitated networks is as follows. Denote by C the sum of capacities over all edges in the network. The routing algorithm views each c-capacity edge as representing c 1-capacity bidirectional parallel edges, or "p-edges." The number of 1-capacity p-edges is C.

First, we observe that nothing in the analysis of the network of 1-capacity edges was incompatible with parallel edges. Thus, if the adversary is also limited to view c-capacity edges as c 1-capacity p-edges, then all the proofs hold with m substituted by C. We denote the parameterized class of such adversaries by $\tilde{A}(w, \varepsilon)$. For concreteness, we state below the restrictions on any $\tilde{A}(w, \varepsilon)$ adversary. For any time $t \in \mathbb{N}$, let $\mathcal{I}^t$ be the set of packets injected during the w time steps from t to $t + w - 1$, inclusive. Then, the adversary can associate with each packet $p = (s, d) \in \mathcal{I}^t$, a simple path of 1-capacity p-edges from s to d, such that each direction of every 1-capacity p-edge $e \in E$ is used by these paths at most $\lfloor (1 - \varepsilon)w \rfloor$ times.

A slightly more involved view of a capacitated network yields the same bounds on buffer size and delivery time but allows the adversary to inject more packets. We denote the new parameterized class of adversaries by $\bar{A}(w, \varepsilon)$. An $\bar{A}(w, \varepsilon)$ adversary can associate with each packet $p = (s, d) \in \mathcal{I}^t$, a simple path of capacitated edges from s to d, such that each direction of every c-capacity edge $e \in E$ is used by these paths at most $cw - \lceil \varepsilon w \rceil$ times.

In this view of the network each c-capacity edge is composed of c 1-capacity "channels." The modified algorithm maintains a buffer at each node for each channel, destination pair. So, the number of buffers is the same as in the previous approach. However, as described above, an $\bar{A}(w, \varepsilon)$ adversary can "have in mind" a path composed of capacitated edges, whereas the $\tilde{A}(w, \varepsilon)$ adversary must "commit" to the 1-capacity p-edge used in each capacitated edge of a path. Lemma 4 is easily modified to achieve a potential drop using a path of channels rather than edges. Claim 6 is modified as follows. For each packet $(p_k, d_k) \in \mathcal{I}^t$, let π_k be the path of *edges* from p_k to d_k given by the $\bar{A}(w, \varepsilon)$ adversary.

Claim 19: For each path π_k, as above, we can associate a sequence of times $T^k = \{t_1^k, \ldots, t_{\ell_k}^k\}$ *and* a sequence of channels $\{\bar{e}_1^k, \ldots, \bar{e}_{\ell_k}^k\}$ where $t \leq t_j^k \leq t + w - 1$ for $1 \leq j \leq \ell_k$ in a way that for each channel, at most w distinct time steps (in each direction) are assigned *and*

each edge, in each direction will still have at least $\lceil \varepsilon w \rceil$ "free" time steps among its channels during time window $\mathcal{W}_w^t$.

The proof of this claim is very similar to the proof of Claim 6. The remainder of the proof that BASIC has bounded buffers then follows. For DBASIC, the Update protocol is slightly modified. Each channel sends $O(\log n)$ bits of update information each step. Again, using Claim 19, the remainder of the proof for bounded buffers remains the same. The proofs of the bounds for BOUNDEDDT(W) follow with very simple modifications. For an $\bar{A}(w, \varepsilon)$, the protocol is modified slightly. Rather than having all the channels of each capacitated edge be devoted to the drainage protocol for one step every $2W$ steps, only one channel of each capacitated edge is devoted to the drainage protocol every $2W$ steps.

5.3 Other Issues

Robustness: Diffusion-type protocols are often robust to edge failure. To model edge failure in our context, Definition 1 should be modified as follows. An $A(w, \varepsilon)$ adversary is allowed to inject packets, as before, and to have edges operational, i.e., "up", during certain time steps or not operational, i.e., "down," subject to the following constraint. For any time $t \in \mathbb{N}$, let $\mathcal{I}^t$ be the set of packets injected during time window $\mathcal{W}_w^t$. Then, for every $t \in \mathbb{N}$, the adversary can associate with each packet $p = (s, d) \in \mathcal{I}^t$, a simple path from s to d, such that for each edge $e \in E$ and each direction of e, the number of time steps that the edge is up minus the number of paths in $\mathcal{I}^t$ which use this edge in that direction is at least $\lceil \varepsilon w \rceil$. With this definition, protocol BASIC should only be modified as to not send a packet over a "down" edge (the buffers associated with such edge are still maintained as before). If we assume that when an edge comes up after being down, the nodes are notified as to the status of their neighbors' buffers, then the proof for BASIC goes through with only minor modifications. For protocol DBASIC the only modification needed is that when an edge comes up after being down, all the control bits that would have been sent during the steps in which the edge was down are assumed to be sent during the procedure for bringing the edge up. (Alternatively, if the edge is down for a long time, it may be desirable to send the size of the buffers associated with the edge).

For capacitated networks, the adversary may vary the capacity of each edge. The definitions of the adversaries for capacitated networks are easily generalized to allow the adversaries to vary the capacity of the edges. For $\tilde{A}(w, \varepsilon)$ adversaries, for each p-edge and each direction, the number of time steps that the edge is up minus the number of paths that pass through the edge in that direction is at least $\lceil \varepsilon w \rceil$. For $\bar{A}(w, \varepsilon)$ adversaries, for each edge and each direction, the sum of the number of steps each channel of the edge is up minus the number of paths that pass through the edge in that direction is at least $\lceil \varepsilon w \rceil$.

Directed networks: Our results still hold even if the network is directed. If the network is not strongly connected then we have to add a slight modification to our protocol; namely, node u never sends a packet with destination d, over an edge leading to node v, if there is no

directed path from v to d (and hence u does not have at all a buffer for such packets). With this modification, our proofs still hold in the directed case. We still need to allow control bits to flow across edges in both directions (or to assume that nodes know the size of the buffers across their adjacent edges).

Other potential functions: In this paper we analyze our protocol using a "linear potential function" Φ (it is "linear" in the sense that the contribution of every packet is linear in its height). When "diffusion-type" algorithms have been used in other contexts, "exponential potential function" variants of the algorithms have yielded improved bounds (See [AL2, GL+]). It is worthwhile checking whether exponential potential function variants of our protocols would yield improved bounds.

Acknowledgments We thank Matthew Andrews, Allan Borodin, Tom Leighton and Yuval Rabani for useful discussions. We also thank an anonymous referee for useful comments.

References

[AAF+] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu, "Universal Stability Results for Greedy Contention-Resolution Protocols", *Proc. of 37th FOCS*, pp. 380–389, 1996.

[AAG+] Y. Afek, B. Awerbuch, E. Gafni, Y. Mansour, N. Shavit, A. Rosén. Slide - The Key to Polynomial End-to-End Communication. *Journal of Algorithms*, Vol. 22, No. 1, pp. 158–186, 1997.

[AAMR] W. Aiello, B. Awerbuch, B. Maggs, and S. Rao, "Approximate Load Balancing on Dynamic and Asynchronous Networks," *Proc. of 25th STOC*, pp. 632–641, 1992.

[AAP] B. Awerbuch, Y. Azar, and S. Plotkin, "Throughput Competitive On-Line Routing," *Proc. of 34th FOCS*, pp. 32–40, 1993.

[AAPW] B. Awerbuch, Y. Azar, and S. Plotkin, and O. Waarts, "Competitive Routing of Virtual Circuits with Unknown Duration," *Proc. of 5th SODA*, pp. 321–330, 1994.

[AFH+] M. Andrews, A. Fernández, M. Harchol-Balter, and T. Leighton, L. Zhang, "General Dynamic Routing with Per-Packet Delay Guarantees of O(distance + 1/session rate)," *Proc. of 38th FOCS*, pp. 294–302, 1997.

[AGR] Y. Afek, E. Gafni, A. Rosén, "Slide—A Technique for Communication in Unreliable Networks," *Proc. of 11th PODC*, pp. 35–46, 1992.

[AL] B. Awerbuch and T. Leighton, "A Simple Local-Control Approximation Algorithm for Multicommodity Flow", *Proc. of 34th FOCS*, pp. 459–468, 1993.

[AL2] B. Awerbuch and T. Leighton, "Improved Approximation Algorithms for the Multi-commodity Flow Problem and Local Competitive Routing in Dynamic Networks", *Proc. of 26th STOC*, pp. 487–496, 1994.

[AMS] B. Awerbuch, Y. Mansour, N. Shavit, "End-to-End Communication with Polynomial Overhead," *Proc. of 30th FOCS*, pp. 358–363, 1989.

[BFU] A.Z. Broder, A.M. Frieze, and E. Upfal, "A General Approach to Dynamic Packet Routing with Bounded Buffers", *Proc. of 37th FOCS*, pp. 390-399, 1996.

[BKR+] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. Wiliamson, "Adversarial Queuing Theory", *Proc. of 28th STOC*, pp. 376–385, 1996.

[BU] A.Z. Broder, and E. Upfal, "Dynamic Deflection Routing on Arrays," *Proc. of 28th STOC*, pp. 348-355, 1996.

[C] R. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *IEEE Transactions on Information Theory*, pp. 114–131, 1991.

[C2] R. Cruz, "A Calculus for Network Delay, Part II: Network Analysis," *IEEE Transactions on Information Theory*, pp. 132–141, 1991.

[CMSV] R. Cypher, F. Meyer auf der Heide, C. Scheideler, and B. Vöcking. "Universal algorithms for store-and-forward and wormhole routing", *Proc. of 28th STOC*, pp. 356-365, 1996.

[GL+] B. Ghosh, T. Leighton, B. Maggs, S. Muthukrishnan, G. Plaxton, R. Rajaraman, A. Richa, R. Tarjan, and D. Zuckerman, "Tight Analyses of Two Local Load Balancing Algorithms," *Proc. of 27th STOC*, pp. 548–558, 1995.

[GM] B. Ghosh and S. Muthukrishnan, "Dynamic Load Balancing on Parallel and Distributed Networks by Random Matchings," *Proc. of 6th SPAA*, pp. 226–235, 1994.

[HB] M. Harchol-Balter and P. Black, "Queuing Analysis of Oblivious Packet-Routing Algorithms," *Proc. of 5th SODA*, pp. 583-592, 1994.

[HW] M. Harchol-Balter and D. Wolfe "Bounding Delays in Packet Routing Networks," *Proc. of 27th STOC*, pp. 248-257, 1995.

[KPP] A. Kamath, O. Palmon, and S. Plotkin, "Routing and Admission Control in General Topology Networks with Poisson Arrivals," *Proc. of 7th SODA*, pp. 269-278, 1996.

[KT] J. Kleinberg and E. Tardos. "Disjoint Paths in Densely Embedded Graphs." *Proc. of 36th FOCS*, pp. 52-61, 1995.

[L] F.T. Leighton. "Methods for message routing in parallel machines". Invited paper in *Proc. of 24th STOC*, pp. 77-96, 1992.

[L2] F.T. Leighton, personal communication, 1998.

[LMR] T. Leighton, B. Maggs, S. Rao, "Packet Routing and Job-Shop Scheduling in O(congestion+dilation) Steps," *Combinatorica*, Vol. 14, No. 2, pp. 167–180, 1994.

[LMRi] T. Leighton, B. Maggs and A. Richa, "Fast Algorithms for Finding O(Congestion+Dilation) Packet Routing Schedules," *Combinatorica*, to appear.

[M] M. Mihail, "Conductance and Convergence of Markov Chains—A Combinatorial Treatment of Expanders," *Proc. of 30th FOCS*, pp. 526–531, 1989.

[Mi] M. Mitzenmacher, "Bounds on the Greedy Routing Algorithm for Array Networks", J. Comput. System Sci. 53 (1996), No. 3, pp. 317–327.

[MP] Y. Mansour, and B. Patt-Shamir, "Greedy Packet Scheduling on Shortest Paths", *Journal of Algorithms*, Vol. 14, No. 3, pp. 99–129, 1993.

[MV] F. Meyer auf der Heide and B. Vöcking. "A packet routing protocol for arbitrary networks". *Proc. of STACS '95.*

[OR] R. Ostrovsky and Y. Rabani, "Local Control Packet Switching Algorithm," *Proc. of 29th STOC*, pp. 644–653, 1997.

[PG] A. Parekh, and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Transactions on Networking*, 1 (3) pp. 344–357, 1993.

[PG2] A. Parekh, and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node Case," *IEEE/ACM Transactions on Networking*, 2 (2) pp. 137–150, 1994.

[RT] Y. Rabani and É. Tardos. "Distributed packet switching in arbitrary networks". *Proc. of 28th STOC*, pp. 366-375, 1996.

[ST] A. Srinivasan and C.-P. Teo. "A constant-factor approximation algorithm for packet routing, and balancing local vs. global criteria". *Proc. of 29th STOC*, pp. 636-643, 1997.

[STs] G. Stamoulis and J. Tsitsiklis, "The Efficiency of Greedy Routing in Hypercubes and Butterflies," *IEEE Transactions on Communications*, 42 (11), pp. 3051–208, 1994.

[SV] C. Scheideler and B. Vöcking, "Universal Continuous Routing Strategies," *Proc. of 8th SPAA*, 1996.