Effective routing and scheduling in adversarial queueing networks

Jay Sethuraman * Chung-Piaw Teo †

December 2003. Revised June 2004

Abstract

In an adversarial queueing network, the incoming traffic is decided by an adversary, who operates under a reasonable rate restriction. This model provides a valuable, complementary point of view to that of the traditional queueing network model in which arrivals are modeled by *stochastic processes*. As a result, the adversarial queueing network model has attracted a lot of attention in recent years, especially as a way of modeling packet injections into a communication network. Our main result is a simple, effective packet routing and scheduling algorithm with a provably-good performance. Specifically, our algorithm keeps the system *stable* (bounded number of packets in the system), with the bound on the number of packets in the system that is $O((1-r)^{-1})$, where r can be interpreted as the arrival rate of the packets into the communication network. This improves upon the work of Gamarnik [12], who designed an algorithm for which the number of packets in the system is $O((1-r)^{-2})$; moreover, our result matches the traditional queueing-theoretic number-in-system bound.

1 Introduction and Contributions

Scheduling and packet-routing have emerged as important issues in modern computer and communication systems. In this paper, we consider one such problem in the setting of an arbitrary synchronous, *adversarial* network. In an adversarial queueing network, the incoming traffic is decided by an adversary, who operates under a reasonable rate restriction. This queueing model has attracted a lot of attention in recent years as it appears to be a convenient and useful way of modeling packet

*IEOR Department, Columbia University, New York, NY, jay.sethuraman@columbia.edu. The research of the author was partially supported by an NSF CAREER Award and by an IBM Faculty partnership award.

†Department of Decision Sciences, National University of Singapore, Singapore 119260, bizteocp@nus.edu.sg. The research of the author was partially supported by a Fellowship from the Singapore-MIT Alliance Program.

injections into a communication network. Moreover, this model has inspired algorithm developers to design robust algorithms that provide a performance guarantee, while allowing for worst-case arrival patterns "locally." Thus, the adversarial input model provides a valuable, complementary point of view to that of the traditional queueing network model in which arrivals are modeled by *stochastic processes*.

Problem description. The communication network is modeled by a directed graph $G = (V, E)$. Nodes represent processors and arcs (or edges) represent links between processors. Packets are injected *over time* by an adversary subject to a rate restriction, which we shall describe in a moment.

Two natural models arise, depending on whether or not the adversary specifies a *route* for the packets she injects:

(a) In the non-adaptive (or circuit-switched) model, the adversary specifies a path P for each packet p; the algorithm is required to route each packet along its associated path.

(b) In the adaptive (or packet-switched) model, the adversary specifies only the origin and destination for each packet, but does not specify a path. In this case, the algorithm is free to route a packet along any path from its origin to its destination.

To get a non-trivial problem, it is necessary to restrict the power of the adversary. A reasonable, even natural, restriction on the adversary for the two models is specified in terms of two parameters r and w. For the non-adaptive model, the packets injected by the adversary (and their associated paths) should be such that in any time window of size w, the number of packets injected during this window requiring a particular arc must be at most $\lfloor rw \rfloor$. The non-adaptive approach, however, does not allow packets to dynamically adapt to congestions and deadlocks along their routes. This limitation can be overcome by allowing the paths to be *selected* adaptively. For the adaptive model, the analogous restriction is that the adversary *must be able to* associate paths to the packets injected in any time window of size w such that the number of packets requiring a particular arc is at most $\lfloor rw \rfloor$. This condition can be most conveniently described by a certain integer multicommodity flow problem having an optimal value at most $\lfloor rw \rfloor$. An alternative model that uses a burstiness parameter b and a rate parameter r has also been used in the literature to restrict the adversary, see [3] for example. This model is also in terms of the "load" on any arc. Specifically, for each $T \geq 1$, the load on any arc due to packets injected in *any* window of length T must be at most $rT + b$. If $r < 1$, which is the case of interest in this paper, Rosén [14] shows that these two models are equivalent. (Rosén's paper contains other nice observations relating these two input models.)

In this paper we focus on the adaptive model, although most of our results can be extended, with minor modifications, to the non-adaptive model as well. In fact, we focus on the adaptive model in which the adversary is allowed to *split* packets and route them using multiple paths; the algorithm, in contrast, is required to route each packet along a single path. This new restriction gives the adversary more freedom, and so makes her more powerful: essentially the new restriction translates to a certain *fractional* multicommodity flow problem having an optimal value at most rw. For this model, we consider the problem of designing effective routing/scheduling algorithms. Our main result is a simple algorithm for this problem that is *stable* (bounded number of packets in the system), with a bound on the number of packets in the system that is $O(w/(1-r))$ (the constant depends on m, n, where m and n are the number of arcs and nodes in the network, but we have explicitly specified the dependence on the parameters of the adversary). A noteworthy feature of this result is that it matches the traditional queueing-theoretic number-in-system bound, which is typically $O(1/(1-r))$. Our main result implies a worst-case delay bound on packets that is relatively small as well.

Related work. Adversarial networks have received a lot of attention in recent years. They were first introduced by Borodin et al. [9], and further elaborated on by Andrews et al. [3, 4]. Later, these were seen to be non-trivial generalizations of earlier models of Cruz [10]. The original papers of Borodin et al. [9] and Andrews et al. [3, 4] contain a wealth of interesting results, but mostly on the non-adaptive case.

The models most closely related to our work were first introduced by Aiello et al. [2]. In their work, they provided an elegant extension of the restriction on the adversary, which was previously considered only for the non-adaptive case. Furthermore, they constructed a distributed scheduling/routing protocol such that the number of packets in the system was $O(n^{5/2}m^{5/2}w/(1-r))$. Their results were derived for the *integer* (w,r) adversary. Recall that in this model, the adversary must be able to associate *paths* with each packet injected in any window of size w so that the load on any arc is at most $\lfloor rw \rfloor$. Motivated by the observation that this restriction is not efficiently checkable, Gamarnik [12] introduced the *fractional* (w,r) adversary. In this setting, the adversary is allowed to associate fractional paths ("flows") to the packets to satisfy the load condition. An interesting question, then, is to quantify the performance loss due to the increased power given to the adversary. Gamarnik [12] constructed an algorithm such that the number of packets in the network is $O(w^2/(1-r)^2)$; furthermore, he observed that a naive adaptation of the methods of Aiello et all. [2] can at best lead to a bound of $O(1/(1-r)^3)$.

In more recent work, Andrews et al. [5] derive *distributed* source routing and scheduling algorithms with polynomial delay bounds using a discrete-review like strategy; these delays bounds naturally imply bounds on the number-in-system. The algorithm described in this paper can also be viewed as a source routing/scheduling algorithm, as the route for a packet is determined at its source; the queue-length bounds we prove are stronger than those implicit in [5], but our algorithm is centralized. For the special case in which there is only a single destination, stronger bounds are known [6].

Results. For the dynamic adaptive packet routing problem in an adversarial queueing network with a fractional (w, r) adversary, we design an efficient scheduling/routing algorithm that keeps the queue-lengths bounded. Specifically, we show that the number of packets in the system at any time t, $Q(t)$, satisfies

$$Q(t) \leq \frac{m(m + 2n + mn^3 + w)}{1 - r}, \tag{1}$$

where m and n are the number of arcs and nodes in the network, and $r < 1$. This matches the known bound (as a function of w and r) for the same problem with an integer (w, r) adversary. Our results immediately imply small delay bounds for the packets as well.

Our bounds obviously apply in the special case when rates are associated with origin-destination pairs. Specifically, suppose packets for a particular origin-destination pair i, j arrive at rate r_{ij}. As long as an associated *fractional* multicommodity flow problem has optimal value at most 1, we can find a scheduling policy with the number of packets bounded by the expression (1), where r can be explicitly determined based on the r_{ij} and the network topology alone.

Our main result is achieved by a combination of techniques: We use a discrete-review policy, which reduces the dynamic scheduling/routing problem to a sequence of *static*, adaptive packet routing problems; using a rounding theorem due to Karp et al. [13], we reduce each of these problems to a *non-adaptive* packet scheduling problem; these packet scheduling problems can be solved effectively using algorithms due to Bertsimas and Sethuraman [8] or Sevastyanov [15, 16, 17]. This approach also highlights an intimate connection between two seemingly unrelated problems: the problem of controlling a *dynamic* queueing network effectively, and the problem of finding a "good" schedule for a *static* job-shop scheduling problem.

The rest of this paper is structured as follows: in § 2 we describe the model in more detail; in § 3, we describe the scheduling/routing algorithm, and specify the details in each of the steps informally outlined earlier and in §2. We end the paper by mentioning related open problems for future research.

2 Model

The model we consider is the "adversarial queueing network" model advocated by Borodin et al. [9], as modified by Aiello et al. [2]; we refer the reader to these original papers for a thorough motivation of the adversarial model. The basic model used throughout this paper can be described as follows: The communication network is modeled by a directed graph $G = (V, E)$, where V and E denote the node set and directed arc set in the graph G; this network is populated by *packets*, which originate in some node of the network, and need to reach some other node of the network. Thus, nodes represent processors (or switches), and arcs represent links between processors (switches). Associated with each arc (u, v) is a buffer residing at node u; this buffer stores the packets that need to traverse the arc (u, v). (Buffers are assumed to be large enough so that packets are never dropped because of a full buffer.) We assume a synchronous network, in which time is divided into *steps*, conveniently numbered by the non-negative integers, and indexed by t. Packets require unit time to traverse an arc; thus, each arc can process at most one packet in a time step, and each packet can traverse at most one arc in a time step.

Packets are injected into the network by an *adversary*. In the basic model proposed in [9], the adversary specifies the path for each packet she injects into the network. The adversary is free to inject packets and specify the paths for these packets as long as she obeys a *rate* constraint, r, over *any* time interval. Specifically, if $A_P[t_1, t_2)$ denote the number of packets injected into the network during the time interval $[t_1, t_2)$ that require path P, then,

$$\sum_{P:e\in P} A_P[t_1, t_2) \;\leq\; \lceil r(t_2 - t_1)\rceil,$$

for any edge e in G.

This model was generalized to allow (bounded) burstiness by Andrews et al. [3, 4], who required the adversary to obey a rate constraint over a time *window* w. This is captured by requiring

$$\sum_{P:e\in P} A_P[t, t + w) \;\leq\; \lceil rw\rceil.$$

This model was extended in an elegant way by Aiello et al. [2] to the case in which the adversary specifies only the origin node and destination node for the packets she injects, but *does not specify the path*. Let $A_{ij}[t_1, t_2)$ be the set of packets injected into the network during the time interval $[t_1, t_2)$, with origin i and destination j, and let

$$A[t_1, t_2) \;=\; \bigcup_{i,j\in V} A_{ij}[t_1, t_2).$$

In this setting, the restriction on the adversary translates to the following definition:

Definition. An adversary is an *integer* (w, r) adversary for some r $(0 < r < 1)$ and some integer $w \geq 1$ if and only if for any t, the adversary can associate a path to each packet in $A[t, t + w)$ such that every arc belongs to at most $\lfloor rw \rfloor$ paths.

Remark. An important point to note is that the adversary is not constrained to have a single path in her mind for the packets she injects. A packet p injected at time t will belong to w different time windows; the adversary is allowed to associate different paths to packet p at the time instants $t - w + 1, t - w + 2, \ldots, t - 1, t$.

It is easy to see that an adversary is an integer (w, r) adversary if and only if

$$C^*(t) \ \leq \ \lfloor rw \rfloor,$$

where $C^*(t)$ is defined as the optimal value of the *integer* multicommodity flow problem

(IMF) $\min \quad C(t)$

subject to:

$$\sum_{l:(i,l)\in E} x_{ij}^{il} = A_{ij}[t, t + w), \quad \forall i, j \in V,$$

$$\sum_{k:(k,j)\in E} x_{ij}^{kj} = A_{ij}[t, t + w), \quad \forall i, j \in V,$$

$$\sum_{l:(k,l)\in E} x_{ij}^{kl} = \sum_{l:(l,k)\in E} x_{ij}^{lk} \quad \forall i, j \in V, \ k \neq i, j,$$

$$C^{kl} \ = \ \sum_{i,j\in V} x_{ij}^{kl}, \quad \forall (k, l) \in E,$$

$$C(t) \ \geq \ C^{kl}, \quad \forall (k, l) \in E,$$

$$x_{ij}^{kl} \ \geq \ 0, \ \text{ integer.}$$

Note that x_{ij}^{kl} represents the number of packets from the origin i to destination j that use the arc (k, l). The first three constraints are flow balance constraints on the packets with origin i and destination j.

Since the integer (w, r) adversary is defined in terms of an integer multicommodity flow problem, it is difficult to check whether or not an input stream generated by an adversary respects the restrictions imposed. Motivated by this consideration, Gamarnik [12] considers a *fractional* (w, r) adversary, defined below.

Definition. An adversary is a *fractional* (w, r) adversary for some r $(0 < r < 1)$ and some integer $w \geq 1$ if and only if for any t, the adversary can *fractionally* schedule all the packets in $A[t, t+w)$ such that the load on each arc is at most rw. Equivalently, an adversary is a fractional (w, r) adversary if and only if the linear programming relaxation associated with the integer multicommodity flow problem (IMF) has optimal value at most rw.

The fractional (w, r) adversary is less constrained, and hence can generate input streams that are inadmissible for the integer (w, r) adversary. For the integer (w, r) adversary, Aiello et al. [2] constructed a routing and scheduling policy for which the total number of packets in the system is

$$O\Big(\frac{n^{5/2}m^{5/2}w}{1-r}\Big).$$

In fact, their algorithm is *distributed* and uses only local information. Gamarnik [12] designed a *centralized* algorithm for the fractional (w, r) adversary for which the total number of packets in the system is

$$O\Big(\frac{n^4m^3 + w^2m}{(1-r)^2}\Big).$$

Moreover, he observed that a naive application of the algorithm of Aiello et al. results in an algorithm for which the total number of packets in the system grows as $O(1/(1-r)^3)$. Gamarnik [12] left open the problem of designing an algorithm for which the total number of packets in the system is $O(1/(1-r))$, matching the bound of Aiello et al. [2] for the integer (w, r) adversary. Our main result is an algorithm with this performance bound. We achieve this using a combination of techniques that have proved to be useful in a host of other problems: these include a scheduling algorithm for large job shop scheduling problems due to Bertsimas and Sethuraman [8], and the rounding theorem due to Karp et al. [13]. We emphasize that, unlike the adversary, the scheduling algorithm is not allowed to split a packet, but is required to route each packet along a *single* path.

To avoid ambiguity, we specify explicitly the sequence of events occurring at any time step: first, packets traverse arcs; next, the adversary injects new packets into the nodes; and finally, packets that reach their destination are absorbed by the corresponding node.

3 The routing and scheduling algorithm

A high-level overview of the algorithm is as follows:

(a) The *dynamic* routing and scheduling problem in adversarial networks can be reduced to a sequence of *static*, *adaptive* packet routing problems;

(b) Each of these adaptive packet routing problems can be solved as a (non-adaptive) packet *scheduling* problem with a *small* number of paths;

(c) Each of these packet scheduling problems can be solved effectively; and

(d) the performance loss in each of these steps is relatively *negligible*.

The rest of this section is devoted to showing the details involved in each of these steps.

Reduction to static, adaptive, packet routing. The dynamic routing and scheduling problems in adversarial queueing networks can be reduced to a sequence of adaptive packet routing problems. The reduction is achieved by using a class of *discrete-review* policies. In any such policy, the system is reviewed at discrete points in time, say, at

$$T_0 \equiv 0^+, T_1, T_2, \ldots, T_i, T_{i+1}, \ldots.$$

Policies differ in the way in which the review epochs are picked; we shall not expand on this point any further because our algorithm picks these review epochs in a natural way, as described below.

Suppose T_i is a review epoch chosen by our algorithm. At T_i, we solve an adaptive packet routing problem, with the inputs given by $\{A_{kl}[T_{i-1}, T_i)\}$. In other words, the packets considered by the scheduling/routing algorithm at time T_i are *precisely* those that were injected into the network at or after the previous review epoch; these are routed to their respective destinations using a "good" adaptive packet routing algorithm. The epoch at which all of these packets are routed to their destinations defines the next review epoch T_{i+1}. Note that packets that arrived at or after T_i are ignored by the adaptive packet routing algorithm until T_{i+1}. Clearly, the review epochs chosen by are a function of the adaptive packet routing algorithm used; and the effectiveness of such a policy will critically depend on how good the adaptive packet routing algorithm actually is. We shall analyze this next.

At the epoch T_i, we shall process all the packets that arrived during the interval $[T_{i-1}, T_i)$. Let W_i be the optimal value of the associated fractional multicommodity flow problem. It is clear that every scheduling/routing algorithm will require at least W_i units of time to process this input; specifically, in the absence of arrivals at or after T_i, no scheduling/routing algorithm can process all of the input by time $t < T_{i-1} + W_i$.

Suppose our adaptive packet routing algorithm is able to route all of these packets to their destinations in at most $W_i + f$ steps, for some (constant) f that depends only m and n, but not on

the input to the packet routing problem. (It is important that f be independent of W_i.) Thus, f is a measure of the inefficiency of the adaptive packet routing algorithm, and bears directly on the amount of "work" seen by the algorithm at the next review epoch. Given this, how large can W_{i+1} be? Clearly, W_{i+1} represents the maximum load on any arc due to arrivals in $[T_i, T_{i+1})$, which by our assumption is contained in $[T_i, T_i + W_i + f)$. Therefore,

$$W_{i+1} \;\leq\; \left\lceil \frac{(T_{i+1} - T_i)}{w} \right\rceil rw \;<\; \left(\frac{(T_{i+1} - T_i)}{w} + 1 \right) rw \;<\; r(T_{i+1} - T_i) + w \;\leq\; rW_i + f + w, \qquad (2)$$

since $r < 1$.

A recursive application of Eq. (2) implies

$$\limsup_{i \to \infty} W_i \;\leq\; \frac{f + w}{1 - r}.$$

Thus, letting $Q(t)$ denote the total number of packets in the system at time t, we have

$$Q(t) \;\leq\; m \limsup_{i \to \infty} W_i \;\leq\; \frac{m(f + w)}{1 - r}. \qquad (3)$$

Thus, the dynamic routing/scheduling problem in an adversarial queueing network can be solved as a sequence of *static*, adaptive packet routing problems, as long as each of these problems is solved relatively well; in particular, the queue-length bound of Eq. (3) will hold as long as the *makespan* of the static, adaptive packet routing problem is within an *additive constant* of the associated *congestion* lower-bound.

Identifying a small set of "good" paths. Our goal now is to consider a static, adaptive packet routing algorithm. Let t be a review epoch, and let W_t be the optimal value of the (fractional) multicommodity flow problem defined by the packets present in the system at time t. Specifically, if A_{ij} is the number of packets in the system with origin i and destination j at time t, then, W_t is the smallest value for which the set of linear inequalities

$$\begin{aligned}
&\sum_{l:(i,l) \in E} x_{ij}^{il} = A_{ij}, \quad \forall i, j \in V, \\
&\sum_{k:(k,j) \in E} x_{ij}^{kj} = A_{ij}, \quad \forall i, j \in V, \\
&\sum_{l:(k,l) \in E} x_{ij}^{kl} = \sum_{l:(l,k) \in E} x_{ij}^{lk} \quad \forall i, j \in V,\ k \neq i, j, \\
&C^{kl} \;=\; \sum_{i,j \in V} x_{ij}^{kl}, \quad \forall (k,l) \in E,
\end{aligned}$$

$$W_t \geq C^{kl}, \quad \forall (k,l) \in E,$$
$$x_{ij}^{kl} \geq 0.$$

is feasible. Let $(\overline{x})$ be such a feasible solution. Note that without loss of generality, we can assume that $A_{ij} > 0$. Given $\overline{x}$, we can also assume that there does not exist any cycle with positive flow; hence we can decompose the solution (arc-flows) into flows along paths P_k, $k = 1, \ldots, K$ (where $K \leq m$), with the (fractional) flow value on path P_k being y_{P_k}, and such that

$$\sum_{k:(i,j)\in E(P_k)} y_{P_k} = \sum_{u,v\in V} \overline{x}_{u,v}^{i,j} \leq W_t,$$

and

$$\sum_{k:o(P_k)=i,d(P_k)=j} y_{P_k} = A_{ij}.$$

In the expressions above, $o(P_k)$ and $d(P_k)$ denote the origin and destination of path P_k. We refer the reader to Ahuja et al. [1] for a discussion on flow decomposition.

Our task now is to select precisely A_{ij} paths from i to j, without affecting the congestion along any arc adversely; in other words, we need to round the fractional solution $(\overline{x})$ to an integral 0-1 solution in a suitable manner. We do this by using the following rounding algorithm of [13]:

Theorem 1 ([13]) *Let A be a real valued $r \times s$ matrix, and y be a real-valued s-vector. Let b be a real valued vector such that $Ay = b$ and t be a positive real number such that, in* **every column** *of A,* `(i)` *the sum of all the positive entries is at most t* **and** `(ii)` *the sum of all the negative entries is at least $-t$. Then we can compute an integral vector $\overline{y}$ such that for every i, either $\overline{y}_i = \lfloor y_i \rfloor$ or $\overline{y}_i = \lceil y_i \rceil$ and $A\overline{y} = \overline{b}$ where $\overline{b}_i - b_i < t$ for all i. Furthermore, if y contains d non-zero components, the integral approximation can be obtained in time $O(r^3 \lg(1 + s/r) + r^3 + d^2 r + sr)$.*

To use Theorem 1, we first transform our linear system above to the following equivalent form:

$$\sum_{k:(i,j)\in E(P_k)} y_{P_k} \leq W_t \qquad \forall\ (i,j) \in E(G)$$
$$\sum_{k:o(P_k)=i,d(P_k)=j} (-m) y_{P_k} = -mA_{ij} \qquad \forall\ i,j \in V.$$

The set of variables above is $\{y_{P_k} : k = 1, \ldots, K\}$. Furthermore, in this linear system, the positive column sum is bounded by the maximum length of the paths, which in turn is bounded by m, the number of arcs in the graph. The negative column sum is also bounded by $-m$. Thus, the parameter

t for this linear system, in the notation of Theorem 1, can be taken to be m. Hence by Theorem 1, we can obtain in polynomial time an **integral** solution $\overline{y}$ satisfying

$$\sum_{k:(i,j)\in E(P_k)} \overline{y}_{P_k} \leq W_t + m \qquad \forall\ (i,j) \in E(G)$$

$$\sum_{k:o(P_k)=i,d(P_k)=j} (-m)\overline{y}_{P_k} < -mA_{ij} + m \qquad \forall\ i,j \in V.$$

For each i, j, we have

$$\sum_{k:o(P_k)=i,d(P_k)=j} \overline{y}_{P_k} > A_{ij} - 1.$$

Note the crucial role of the *strict* inequality. Thus, we have selected at least A_{ij} paths from i to j; furthermore, the congestion along every arc is bounded by $W_t + m$.

To summarize what we have achieved: starting from an arc flow solution, we used flow decomposition and an application of the rounding theorem to derive an integer solution such that the load on any arc is increased by at most m. Each "commodity" (i.e., origin-destination pair) is now routed along at most m paths. We can now reformulate this adaptive packet routing problem as a (non-adaptive) packet *scheduling* problem as follows: think of each path from i to j as a *type*, and assume that $\overline{y}_k$ packets have to be sent from i to j along path P_k. (To avoid cumbersome notation, we have dropped the dependence of $\overline{y}$ on the origin-destination pair.) In essence, we have used the rounding algorithm to compute a small set of good paths for the adaptive packet routing problem; we now pretend that the problem to be solved is really a packet scheduling problem in which an explicit path is associated with each packet; the number of packets to be routed along a given path is determined by applying the rounding algorithm on an optimal (fractional) multicommodity flow solution.

Solving the packet scheduling problem. The dynamic routing/scheduling problem on an adversarial network is now reduced to a simpler, *static*, packet *scheduling* problem. For convenience, we describe the input to this packet scheduling problem slightly differently. The packet scheduling problem consists of K *types* of packets; packets of type k require a path P_k through the network, are initially available at $o(P_k) \in V$, and need to reach $d(P_k) \in V$; there are n_k packets of type k. The objective is to find a schedule for all of these packets that minimizes makespan. Each packet requires unit time to traverse an arc; each arc can process one packet per unit time. Our goal now is to find a schedule for this packet scheduling problem whose makespan is within an additive constant of the associated congestion lower bound. Note that this additive constant could depend on m, n, K, but

cannot depend on $n_1, n_2, \ldots, n_k$ themselves; this is because in the packet scheduling instances that will arise in the solution of the adversarial network, m, n, and K will be independent of r and w, the parameters of the adversary, whereas the n_k will depend on r and w. We briefly outline two solution methods to this packet scheduling problem, and specify the corresponding bounds.

Fluid synchronization algorithm. The packet scheduling problem outlined here is a special case of the job shop scheduling problem with the makespan objective considered by Bertsimas and Sethuraman [8]. In that work, they consider a *fluid relaxation* of the job shop scheduling problem, which can be viewed as a continuous analog of the discrete job shop scheduling problem. Using an optimal solution to the fluid relaxation, they find nominal start times for each packet at each of the arcs it has to visit; these nominal start times are carefully constructed in a recursive manner, based on both the optimal fluid solution and the partial discrete schedule.

To make this more precise, suppose type k packets need to visit arcs $a_{k,1}, a_{k,2}, \ldots, a_{k,i_k}$ in that order. Suppose W is the maximum load on any arc. The scheduling algorithm discussed in [8] first determines the *fluid* start and completion times for each packet at each stage, defined as follows:

Fluid Start time ($FS_{k,j}(n)$): This is the start time of the n^{th} type k packet at (its) stage j (arc $a_{k,j}$) in the fluid relaxation given by

$$FS_{k,j}(1) = 0, \tag{4}$$

$$FS_{k,j}(n) = FS_{k,j}(n-1) + \frac{W}{n_k}, \quad n > 1. \tag{5}$$

Fluid Completion time ($FC_{k,j}(n)$): This is the completion time of the n^{th} type k packet at (its) stage j in the fluid relaxation, and is given by

$$FC_{k,j}(n) = FS_{k,j}(n) + \frac{W}{n_k}. \tag{6}$$

Notice that the fluid relaxation processes packets continuously; each type k packet is processed by *all* its stages simultaneously at a uniform rate n_k/W, so that all of the n_k packets of type k will be scheduled by time W. In trying to "round" this fluid schedule to an implementable discrete schedule, we need to overcome two difficulties: first, the fluid relaxation treats packets as continuous entities, with the effect that the same packet can be "scheduled" by multiple arcs simultaneously; and second, the fluid relaxation allows arcs to split their effort across multiple packet types, as long as the overall effort allocated by each arc is at most 1 per unit time. In other words, the fluid relaxation

views both the packets and the processing resources as being infinitely divisible. The resulting lower bound is naturally just the congestion lower bound; the dilation bound does not arise because of the continuous nature of the jobs.

The fluid start time of a given packet at a given stage may be viewed as the ideal start time of that packet at that stage, but clearly, this is an unrealistic ideal. Motivated by the question of defining a more realistic target start time for each packet at each stage, Bertsimas and Sethuraman [8] defined *nominal start times*; these are defined in terms of the fluid start and completion times as well as the partial discrete schedule. The precise definitions are as follows.

Discrete Start time ($DS_{k,j}(n)$): This is the start time of the n^{th} type k packet at its stage j (arc $a_{k,j}$) in the discrete schedule. (This is what we would like to determine.)

Nominal Start time ($NS_{k,j}(n)$): The nominal start time of the n^{th} type k packet at its stage j (arc $a_{k,j}$) is defined as follows.

$$\begin{aligned} NS_{k,1}(n) &= FS_{k,1}(n), \\ NS_{k,i}(1) &= DS_{k,i-1}(1) + 1, \quad i > 1, \\ NS_{k,i}(n) &= \max\left\{NS_{k,i}(n-1) + \frac{W}{n_k},\ DS_{k,i-1}(n) + 1\right\}, \quad n, i > 1. \end{aligned}$$

Bertsimas and Sethuraman [8] proposed a simple scheduling rule (called "fluid synchronization algorithm") based on these nominal start times: whenever a node has to make a processing decision, it schedules an available packet with the earliest nominal start time. Note that whenever a packet is chosen to be scheduled at a certain node, its nominal processing time at its next stage can be calculated; so the nominal start times for every packet queued at a node will be known.

The main result of [8] adapted to this special case can be stated as follows:

Theorem 2 *Consider a (non-adaptive) packet scheduling problem with K job types and m arcs. Given initially n_k jobs of type $k = 1, 2, \ldots, K$, suppose the maximum load on any arc is W, and let W^* be the optimal makespan. Then, the fluid synchronization algorithm produces a schedule with makespan time W_{D} such that*

$$W \ \le\ W^* \ \le\ W_{\text{D}} \ \le\ W + n(K+2). \tag{7}$$

Sevastyanov's algorithm. In the mid-seventies, interesting approximation algorithms were derived for several shop scheduling problems. These algorithms were based on beautiful, geometric arguments, and were discovered independently by Belov and Stolin [7], Sevastyanov [15], and Fiala [11]. These methods constructed schedules for job shop scheduling problems with an additive error term that depended only on the number of machines, and the maximum processing time of a job, but not on the number of jobs. Since it is not central to this paper (and in the interest of space), we do not discuss these algorithms in detail; we refer the interested reader to the original papers cited earlier as well as the excellent survey of Sevastyanov [18]. The strongest of these results is due to Sevastyanov [16, 17], which provides a schedule of length with an additive error of at most $(n-1)(mn^2+2n-3)$.

Remark. Note that depending on K, this may or may not be better than the schedule provided by the fluid synchronization algorithm. For the adaptive case, it is seen that the guarantee provided by the fluid synchronization algorithm is slightly better than the one provided by Sevastyanov's algorithm. Moreover, the fluid-based algorithm is not computationally intensive at all, and is very simple to implement. On the other hand, for the non-adaptive case, the adversary may insist that the algorithm route packets along exponentially many paths; in this case, the guarantee provided by the fluid-based method is unattractive, and Sevastyanov's method is clearly better.

The main result. Our main result is obtained by putting all of these steps together. Fix a review epoch i, with W_i being the work seen by the scheduler at this epoch. Then, step 2 results in an instance of the non-adaptive packet scheduling problem with maximum congestion at most $W_i + m$; using the fluid synchronization algorithm for this packet scheduling problem results in a schedule with length at most $W_i + m + n(K+2)$. Noting that there are at most n^2 commodities, and that each of which may use at most m paths, we conclude that the schedule computed at epoch i will have length at most $W_i + m + n^3 m + 2n$. Thus, the inefficiency parameter f is at most $m + 2n + mn^3$; using this in Eq. (3), we have

$$Q(t) \;\leq\; \frac{m(f+w)}{1-r} \;\leq\; \frac{m(m+2n+mn^3+w)}{1-r}, \tag{8}$$

where $Q(t)$ represents the number of packets in the system at time t. For Sevastyanov's algorithm a similar guarantee can be shown to hold. We omit the details.

Our results can now be formally stated as the following theorem.

Theorem 3 *Consider an adversarial queueing network under a fractional (w, r) adversary. If $r < 1$, then the discrete-review scheduling policy constructed keeps the number of packets in the system bounded at all times. In particular, the total number of packets in the system at time t, $Q(t)$, satisfies*

$$Q(t) \leq \frac{m(m + 2n + mn^3 + w)}{1 - r}.$$

■

An immediate corollary is that for adversarial queueing networks in which the arrival rates for packets with origin i and destination j is r_{ij}, an effective scheduling/routing algorithm, for which the number in system is $O(w/(1 - r))$, can be constructed, where r can be explicitly computed based on the r_{ij} using a fractional multicommodity flow problem. Gamarnik [12] considered this model and showed that stable policies exist for this system if and only if the associated fractional multicommodity flow problem has value at most 1. (The r in the expression for the number-in-system bound is exactly the optimal solution to this multicommodity flow problem.)

Since the number in system is relatively small, one can expect that the scheduling algorithm could guarantee small delays for all the packets as well. This, of course, turns out to be true. By construction, every packet stays in the system for at most two review periods, and our analysis implies an explicit bound on the length of these delay periods. Note that these techniques lead to excellent performance guarantees for the non-adaptive version of the problem as well.

4 Conclusions and future work

Our main contribution in this paper is the design of an effective algorithm for routing and sequencing packets in adversarial queueing networks that achieves a small worst-case delay. Several outstanding question remain, two of which deserve to be mentioned: First, the "critical" case $r = 1$ is of interest; this seems especially difficult to understand, and may in fact exhibit very different behavior depending on whether the adversary is fractional (w, r) or integer (w, r) restricted. Second, the algorithm we propose is (semi) centralized, although the queue-length information is used only at the discrete review epochs. In contrast, Aiello et al. [2] proposed a distributed algorithm for the integer (w, r) adversary. It will be interesting to design a distributed algorithm for the problem considered here.

Acknowledgement. We thank the anonymous referees for their comments and suggestions on an earlier version of this manuscript.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosén. Adaptive Packet Routing for Bursty Adversarial Traffic. *Journal of Computer and System Sciences*, **60**, 482–509, 2000. Preliminary version in *Proceedings of the 30th STOC*, 359–368, 1998.

[3] M. Andrews, B. Awerbuch, A. Fernandez, J. Kleinberg, T. Leighton, and Z. Liu. Universal-stability results for greedy contention-resolution protocols. *Proceedings of the 37th FOCS*, 380–389, 1996.

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

[5] M. Andrews, A. Fernandez, A. Goel, and L. Zhang. Source Routing and Scheduling in Packet Networks. *Proceedings of the 42nd FOCS*, 168–177, 2001.

[6] B. Awerbuch, P. Berenbrink, A. Brinkmann, C. Scheideler. Simple Routing Strategies for Adversarial Systems. *Proceedings of the 42nd FOCS*, 158–167, 2001.

[7] I. S. Belov and Ya. N. Stolin. An algorithm in a single path operations scheduling problem. *Mathematical Economics and Functional Analysis*, pages 248–257, 1974. (in Russian).

[8] D. Bertsimas and J. Sethuraman. From fluid relaxations to practical algorithms for job shop scheduling: the makespan objective. *Mathematical Programming*, 92(1):61–102, 2002.

[9] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial Queueing Theory. *Journal of the ACM*, **48**(1):13–38, 2001. Preliminary version in *Proceedings of the 28th STOC*, 376–385, 1996.

[10] R. Cruz. A Calculus for network delay, part II: network analysis. *IEEE Transactions on Information Theory*, 37:132–141, 1991.

[11] T. Fiala. Kozelíto algorithmus a három gép problémára. *Alkalmazott Matematikai Lapok*, 3:389–398, 1977.

[12] D. Gamarnik. Stability of Adaptive and Non-Adaptive Packet Routing Policies in Adversarial Queueing Networks. *SIAM Journal on Computing*, **32**, pp. 371-385, 2003. Preliminary version in *Proceedings of 31st STOC*, 1999.

[13] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. V. Vazirani, and V. V. Vazirani. Global Wire Routing in Two-Dimensional Arrays. *Algorithmica*, 2:113–129, 1987.

[14] A. Rosén. A note on models for non-probabilistic analysis of packet switching networks. *Information Processing Letters*, **84**(5):237–240, 2002.

[15] S. V. Sevastyanov. On an asymptotic approach to some problems in scheduling theory. In *Abstracts of papers at* 3^{rd} *All-Union Conference of Problems of Theoretical Cybernetics*, pages 67–69, Novosibirsk, 1974. Inst. Mat. Sibirsk. Otdel. Akad. Nauk SSSR.

[16] S. V. Sevastyanov. Efficient construction of schedules close to optimal for the cases of arbitrary and alternative routes of parts. *Soviet Math. Dokl.*, 29(3):447–450, 1984.

[17] S. V. Sevastyanov. Bounding algorithm for the routing problem with arbitrary paths and alternative servers. *Kibernetika*, 22(6):74–79, 1986. Translation in *Cybernetics*, 22:773–780.

[18] S. V. Sevastyanov. On some geometric methods in scheduling theory: a survey. *Discrete Applied Mathematics*, 55:59–82, 1994.