Static and Dynamic Hot-Potato Packet Routing in Communication Networks

David Gamarnik [*] Maxim Sviridenko [†]

Abstract

We consider a problem of scheduling packets in communication networks subject to the "hot potato" restriction. In a static version, the problem is to route a set of packets in a communication graph from the origins to destinations along the pre-specified paths in minimal (makespan) time subject to the "hot potato" constraint that no packets wait in the intermediate nodes of their corresponding paths. In a dynamic version, the packets are injected over time by an adversary and the problem is to construct a stable schedule which satisfies the "hot potato" requirement. For the static version of the problem we prove that the ratio between the optimal makespan time and the natural lower bound - *congestion* asymptotically does not exceed $O(\sqrt{m})$ when the total number of packets diverges to infinity, where m is the number of edges in the graph. We complement this result with an instance of the problem for which this ratio is achieved. We also provide a complete classification of graphs for which this ratio is asymptotically equal to one. Our results have immediate ramifications to the dynamic version of the problem in terms of maximal arrival rates for which there exist a stable "hot potato" scheduling policy against an arbitrary adversary.

1 Introduction

1.1 Model description

The focus of this paper is scheduling of packets in digital communication networks subject to the "hot potato" ("no wait in transit") constraint. The communication network is assumed to be an undirected graph. Each packet is assigned to a simple path that has to be followed in order to get from the origin node to the destination node. Only one packet can cross any given edge at a time in one direction (but two can cross in opposite directions simultaneously) and the crossing time is equal to one unit of time. Once the packet starts moving along its preassigned path it has to get to the destination without stopping - no queueing is allowed in the intermediate nodes. This restriction on the packet routing is known as "hot potato" or "no wait" routing. We consider a static and dynamic versions of "hot potato" scheduling problem. In a static version the goal is to bring all the packets from their origins to the destinations in a minimal possible (makespan) time so that the "hot potato" restriction is honored. Specifically, the start times need to be specified for every individual packet so that after the start times the packets will reach their destinations without the conflict with other packets.

In a dynamic version the packets are injected into the network by an adversary. Each injected packet has a specified path it needs to follow. The packets are initially stored in the buffers of their origins. Once they start moving, the packets need to proceed to their destination subject to "hot potato" constraint - no intermediate queuing is allowed. The goal is to construct an online scheduling

[*]IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA. Email address: gamarnik@watson.ibm.com

[†]IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA. Email address: sviri@us.ibm.com

policy which guarantees finite bounds on the queue size at the origination buffers. The injection of the packets by an adversary is assumed to be subject to the following restriction: for every time t and edge e the total number of packets injected over time interval $[0,t)$ that contain e on their path is at most $rt + B$ for some fixed parameters $r, B \geq 0$. A necessary condition for existence of a stable scheduling policy is $r \leq 1$. This is an adversarial queuing model introduced first by Borodin et al. [8]. Our only additional assumption is the "hot potato" constraint. Our goal is to understand the effect of this additional assumption. Specifically we are interested in values of $r \leq 1$ for which a stable scheduling policy exists. This value naturally depends on the communication graph.

It turns out that the static and dynamic versions of "hot potato" scheduling problem are intimately connected to each other and this connection will be demonstrated later in the paper.

1.2 Related work

Our work lies in the intersection of three somewhat independent streams of research in the theoretical computer science and operations research areas. The first is a line of work on (static) "hot potato" packet routing problem in graphs and "no wait" scheduling problems in machine scheduling literature. "Hot potato" routing has been mostly considered in adaptive version where the scheduler is free to choose the paths for packets (as opposed to our non-adaptive case when scheduler only chooses the starting time and the paths are fixed). Both adaptive and non-adaptive versions are known to be NP-complete and most of the research is focused on obtaining algorithms with some level of approximation to the optimality. Hot potato routing problem was first proposed by Baran [5] and the variety of papers were written on the subject later. Most of the algorithms considered assume some specific structure of the graph like mesh or torus. For a survey of papers on hot potato routing see [14].

In scheduling literature the "no wait" term is usually used instead of "hot potato", but approximately optimal algorithms are also the focus of the research. For a survey on scheduling with "no wait" constraint see [13].

The second relevant line of research is asymptotically optimal scheduling and routing algorithms (without hot potato constraint) pioneered by Sevastjanov [15] and extended in a variety of contexts in [6],[7]. In a pioneering paper [15] showed that the ratio $C^*/L_{\max}$ converges to one when the total number of jobs in the scheduling system diverges to infinity. We will show that this fails to hold when the hot potato constraint is present and the ratio can take a variety of values depending on the structure of the graph.

The third line of research is the study adversarial queuing networks initiated by Borodin et al. [8] and followed by several other works [3], [2],[4] [12],[1],[11],[10]. Simple stable scheduling policies were constructed in [3] and [10] which achieve stability as long as the arrival rate $r \leq 1$. We will show in this paper that this fails to hold in most of the networks if hot potato constraint on the schedule needs to be satisfied.

1.3 Our results

Our analysis of the static hot potato scheduling problem starts with introduction of an important notion - *relative congestion* of the graph. This is simply the maximal value of of the ratio $C^*/L_{\max}$ over the family of all instances for a given graph, when the total number of packets in the network diverges to infinity. Here again C^* is the optimal makespan time for hot potato scheduling and $L_{\max}$ is the maximal congestion. Defined in this way, the relative congestion is a property of the graph itself. In Section 2 we prove that the relative congestion of every graph is at most $O(\sqrt{m})$, where m is the number of edges, and we construct an example to show that this bound is tight. We then completely characterize graphs for which the value of the relative congestion is equal to one. Specifically, we prove that the relative congestion is equal to one if and only if the graph is a star tree, i.e. a tree that contains only one node with the degree three or higher, or if the graph is a cycle. We close the section by analyzing relative congestion on trees. We prove that it always lies in the interval $[1, 2]$ by constructing an algorithm which achieves ratio $C(\text{algorithm})/L_{\max} \leq 2$. We also provide an example of a tree for which the relative congestion is at least $7/6 > 1$ and conjecture that for certain trees it can be equal to 2.

The implications of the results above to adversarial queuing networks are discussed in Section 3. We first provide a simple proof $r = 1/\textit{relative congestion}$ is the maximal arrival rate for which there exists a stable schedule. In particular, using the result of Section 2, a stable scheduling policy exists for every graph if the arrival rate $r \leq c/\sqrt{m}$ for a certain universal constant c. Also for a some other universal constant $c' > c$ and for a certain family of graphs a stable policy cannot exist if $r = c'/\sqrt{m}$. We conclude in Section 4 with some open questions and conjectures.

2 Static hot potato routing

An instance of a (static) hot potato scheduling problem is given as a graph $G = (V, E)$ and a vector of nonnegative integers $x = (x_p)_{p \in \mathcal{P}}$. Here V and E are the sets of nodes and edges of the graph, respectively, $\mathcal{P}$ is the collection of all simple paths and x_p is understood as the number of packets that need to be processed along the path p. Once the packet reaches the end node of its path p it disappears from the network. We recall that the crossing time for a packet of any given edge is one time unit and only one packet can cross an edge at a time. Let $C^* = C^*(x), x \in \mathcal{Z}_+^{|\mathcal{P}|}$ be the minimal time to bring all the packets from origins to destinations so that the hot potato condition is satisfied. Let $L_{\max} = L_{\max}(x)$ denote the maximal congestion for the instance (V, E, x). That is

$$L_{\max} = \max_{e \in E} \sum_{p : e \in p} x_p. \tag{1}$$

Trivially, $L_{\max} \leq C^*$. The following definition is the key to the rest of the paper.

Definition 1 *Given a graph G let*

$$\alpha(G) = \limsup_{||x||_1 \to \infty} \frac{C^*(x)}{L_{\max}(x)}. \tag{2}$$

The value $\alpha(G)$ is defined to be a relative congestion *of the graph G.*

2.1 Relative congestion for general graphs

The value $\alpha(G)$ is a well-defined function of the graph G. In the absence of the hot potato constraints on the optimal makespan time C^* the Sevastjanov's theorem [15] proves that $C^* \leq L_{\max} + |E|^4$ even for the more general job shop scheduling problem. Specifically, $C^* \approx L_{\max}$ when x is large and $\alpha(G)$ is trivially equal to one. Thus $\alpha(G)$ takes nontrivial values only when the hot potato constraint is enforced.

Since $L_{\max}$ is an easily computed quantity, knowing $\alpha(G)$ amounts to knowing the asymptotic value of the optimal hot potato makespan time when the total number of packets is large. We now present our first main result of this section.

Theorem 2.1 *For any graph G, $\alpha(G) \leq \sqrt{|E|}$. There exists a graph for which $\alpha(G) \geq \sqrt{|E|}/(4\sqrt{2})$.*

Proof. First we prove that $\alpha(G) \leq \sqrt{|E|}$ by explicitly constructing an algorithm with makespan time $C \leq L_{\max}\sqrt{|E|} + F$ where F is a value depending only on number of edges in G and number of different paths used by packets in an instance. We first process all the packets with paths p having length $|p| \geq \sqrt{|E|}$. Denote these paths by $p_1, p_2, \ldots, p_M$. We schedule the corresponding packets as follows. Send the x_{p_1} packets of the path p_1 at times $0, 1, \ldots, x_{p_1}$. It takes $x_{p_1} + |p_1| - 1$ time units to process these packets and clearly the process satisfies hot potato constraint. Then we process $x_{p_i}, i = 2, 3, \ldots, M$ packets for the path p_i in the interval $[\sum_{j=1}^{i-1}(x_{p_j} + |p_j| - 1), \sum_{j=1}^{i}(x_{p_j} + |p_j| - 1)]$ similarly. Again, this schedule is hot potato. Finally, we process the remaining packets with path lengths smaller than $\sqrt{|E|}$ using a greedy algorithm as follows. Sort all the packets in an arbitrary order. Fix any packet f and assume that all the previous packets are already scheduled. We send packet f at the earliest possible time so that it does no conflict with previously scheduled packets. That is if f uses the path $p = \{e_1, e_2, \ldots, e_k\}, 1 \leq k \leq \sqrt{|E|} - 1$ then we find the smallest time t so that edge e_i is clear during the interval $[t + i - 1, t + i]$ for all $i = 0, 1, \ldots, k$.

We analyze the makespan time $\hat{C}$ of this algorithm. Suppose the packet f_{last} scheduled at the latest time uses the path $p = \{e_1, e_2, \ldots, e_k\}$ with $|k| < \sqrt{|E|}$. That is this packet was sent at time $\hat{C} - k$. Then for every $t < \hat{C} - k$ some edge $e_i, 1 \leq i \leq k$ processes some other packet during the interval $[t+i-1, t+i]$. Each edge e_i can process at most $L_{\max}$ packets. Therefore the maximal such t is $L_{\max}k \leq L_{\max}\sqrt{|E|}$. Suppose now that packet f_{last} uses path $p = \{e_1, e_2, \ldots, e_k\}$ with $|k| \geq \sqrt{|E|}$ then by construction of the algorithm the length of the schedule $\hat{C}$ is at most $\sum_{j=1}^{M}(x_{p_j} + |p_j| - 1)$. For any $t \leq \hat{C} - 1$ let $w(t)$ be the total number of edges that processed packets during the time interval $[t, t+1]$. Let $W(t) \equiv \sum_{\tau=0}^{t} w(\tau)$ be the total "work" done in the system by time t. Note that for every $i = 1, 2, \ldots, M$ we have $W(\sum_{j=1}^{i}(x_{p_j} + |p_j| - 1)) - W(\sum_{j=1}^{i-1}(x_{p_j} + |p_j| - 1))$ is at least $|p_i|x_{p_i}$, since during this interval each edge of the path p_i processed x_{p_i} packets. We conclude that $W(\hat{C}) \geq \sum_{i=1}^{M} |p_i|x_{p_i} \geq \sqrt{|E|}\sum_{i=1}^{M} x_{p_i}$. On the other hand $W(\hat{C})/|E|$ is the average work per edge and cannot exceed $L_{\max}$. Combining, $\sum_{i=1}^{M} x_{p_i} \leq L_{\max}\sqrt{|E|}$ or $\hat{C} = \sum_{j=1}^{M}(x_{p_j} + |p_j| - 1) \leq L_{\max}\sqrt{|E|} + \sum_{j=1}^{M}(|p_j| - 1)$. Finally, note that $L_{\max} \to \infty$ as $\sum_{i=1} x_{p_i}$. We conclude that $\alpha(G) \leq \sqrt{|E|}$.

We now construct an example of a graph for which $\alpha(G) = \sqrt{|E|}$. The graph $G = (V, E)$ is a grid $[0, 2n] \times [-n + 1, n]$, see Figure 1. We now identify a set of n paths $p_1, \ldots, p_n$ as follows. Path p_1 is a straight line from $(0, 0)$ to $(0, 2n)$, i.e $\{(0, 0), (1, 0), (2, 0), \ldots, (2n, 0)\}$. The path $p_k, k = 2, 3, \ldots, n$ consists of three consecutive paths: $\{(k - 1, -k + 1), (k - 1, -k + 2), \ldots, (k - 1, 0)\}$, $\{(k - 1, 0), (k, 0), (k, 1), (k+1, 1) \ldots, (2k-2, k-1)\}$ and $\{(2k-2, k-1), (2k-1, k-1), (2k, k-1), \ldots, (2n, k-1)\}$. Note that for every $1 \leq k < l \leq n$ the paths p_k and p_l have exactly one common edge $((l - 1 + (k - 1), k - 1), (l - 1 + (k - 1), k))$. and both of them have lengths $l - 1 + 2(k - 1)$ before the common edge. Therefore, for every t we cannot send packets along the paths p_k and p_l simultaneously. To complete the construction, suppose we have x packets to process for each path $p_k, k = 1, 2, \ldots, n$, where x is a large integer. Then $L_{\max} = 2x$. Let C^* be the optimal hot potato makespan time. By the argument above, at most one packet can be sent at any time t. Therefore $C^* \geq nx = (n/2)L_{\max}$. Finally note that for our grid graph $|E| = 8n^2$. Therefore $C^* \geq \frac{1}{4\sqrt{2}}\sqrt{|E|}L_{\max}$ or $\alpha(G) \geq \frac{1}{4\sqrt{2}}\sqrt{|E|}$. This completes the proof of the theorem. □

2.2 Characterization of graphs with $\alpha(G) = 1$

Since the value of $\alpha(G)$ can be very large for general graphs it is natural to ask for which graphs G the lower bound $\alpha(G) \geq 1$ is achieved? We now provide a complete classification of such graphs and in particular show that for most of the graphs the inequality is strict.

Theorem 2.2 *Given a graph G suppose $\alpha(G) = 1$. Then either G is a simple cycle or G is a tree with at most one node with degree three or higher.*

Proof. Part I. We first prove that $\alpha(G) = 1$ for the two types of graphs mentioned above. Suppose $G = \{v_1, v_2, \ldots, v_n\}$ is a cycle. That is $E = \{(v_1, v_2), (v_2, v_3), \ldots, (v_n, v_1)\}$. Note that the problem is decomposable into two subproblems for packets going clock-wise and counter-clock-wise respectively, since these packets do not interfere. We show that for each individual problem there exist a schedule with makespan time $L_{\max} + n - 1$ where $L_{\max}$ is the congestion for the subproblem. Specifically, consider packets going counter-clock-wise. Let x_j be the set of packets originating at the vertex $v_j, j = 1, \ldots, n$. At time 0 start one packet p_{0j} for all nonempty sets $x_j, j = 1, \ldots, n$ then delete these packets from the sets of available packets, i.e. $x_j = x_j \setminus p_{0j}$. Then for every time t and every j we send a packet from the current set x_j if and only if the edge (v_j, v_{j+1}) is not occupied during $[t, t+1]$ by some previously scheduled packet. We delete the sent packet from x_j.

Clearly, this schedule is hot potato. Moreover, every packet from the set x_j can be delayed at most $L_{\max} - 1$ times, since the congestion of the edge (v_j, v_{j+1}) is at most $L_{\max}$. Since every path has path length at most $n - 1$, then the length of this schedule is at most $L_{\max} + n - 2$. □

Lemma 2.3 *Let G be a star tree depicted on Figure 2. That is G is an arbitrary tree with exactly one node w having degree three or larger. Then $\alpha(G) = 1$.*

Proof. Given an instance $x = (x_p)_{p \in \mathcal{P}}$ we first modify this instance in a way which can only worsen the optimal makespan time. First we extend all the branches of the star to the length $j_{\max}$ of the longest branch. Then for every edge (v, v') which has congestion smaller than maximal $L_{\max}$, we add dummy packets crossing only (v, v') to make the congestion equal to $L_{\max}$. Finally, for every two packets that use consecutive non-overlapping paths of the form $\{v_1, v_2, \ldots, v_j\}$ and $\{v_j, v_{j+1}, \ldots, v_k\}$ we create a single packet which uses a concatenated path $\{v_1, v_2, \ldots, v_k\}$. In the modified version all the packets start and end in the leafs of the tree or in w. For all such packets we extend their paths by creating dummy branches, so that in the result all the packets start and end in the leafs of the modified tree.

Let $v_1, v_2, \ldots, v_N$ be the set of all (real and dummy) leafs of our graph. Consider an $N \times N$ bipartite graph on the set of nodes $v_1^1, \ldots, v_N^1, v_1^2, \ldots, v_N^2$ with the following edges. For each packet which uses a path $\{v_i, \ldots, w, \ldots, v_j\}, 1 \leq i, j \leq N$ construct an edge (v_i^1, v_j^2) in the bipartite graph. We obtain a multigraph since different packets can have the same path. From the construction, the degree of every node $v_i^r, r = 1, 2,\ i = 1, 2, \ldots, N$ is at most $L_{\max}$. It is a well-known fact in graph theory that a bipartite multigraph with maximal degree Δ has a legal edge coloring in Δ colors and there is a polynomial time algorithm for finding such coloring (see for example [9]). Note that each color class is a matching in bipartite graph. We denote these matchings by $M_1, \ldots, M_{L_{\max}}$ (in our case $\Delta = L_{\max}$) and construct the schedule as follows. Consider all the packets which correspond to the matching M_1. We send all these packets at time 0. In general, for every time $t = 0, 1, 2, \ldots, L_{\max} - 1$ we send packets corresponding to the matching M_t at time t. For every time $t \geq j_{\max}$ the edges of the form (w, v) will be processing during $[t, t+1]$ packets sent at time $t - j_{\max}$, that is, packets from $M_{t-j_{\max}}$. Since $M_{t-j_{\max}}$ is a matching, at most one packet requires any individual edge (w, v). Clearly, no conflicts will occur in this schedule for other edges and the makespan time is $L_{\max} + 2j_{\max} - 1$. We conclude that $\alpha(G) = 1$. □

Part II. We now turn to the proof of the second part. We show that for both of the graphs $G = G_{\text{bug}}$ and $G = G_{\text{glass}}$ depicted on a Figure 2, $\alpha(G) > 1$. The extension of the constructions below to trees with two or more nodes with degree at least three and to graphs containing a cycle with extension is simple and is omitted in the interest of space.

Lemma 2.4 *For the graph G_{bug} on Figure 2, $\alpha(G_{\text{bug}}) \geq 7/6$.*

Proof. In the graph G_{bug} we create packets requesting six paths: $p_1 = \{v_4, w_2, w_1, v_2\}, p_2 = \{v_3, w_2, w_1, v_1\}, p_3 = \{v_2, w_1, w_2, v_4\}, p_4 = \{v_1, w_1, w_2, v_3\}, p_5 = \{v_1, w_1, v_2, \}, p_6 = \{v_4, w_2, v_3\}$. For each path we have x packets where x is a large integer. It is easy to check that the maximal congestion is $L_{\max} = 2x$. We now prove that any feasible hot potato schedule has length at least $7/3x$. Let us add a node v_1^1 and edge (v_1, v_1^1) to the graph and let us suppose that packets using path p_5 start in fact from v_1^1. If we had a schedule for the original problem we can easily convert this schedule to the new problem by sending packets using p_5 one time unit earlier. This clearly does not create a conflict and increases the makespan time at most by one. Let T be a makespan time of any feasible schedule. It easy to check that no four packets can be sent simultaneously, because these packets must use different paths and because of the conflict of these paths. Let T_3 be the number of times that three packets were sent simultaneously and let T_2 be the number of times two or one or none packets were sent simultaneously. Then $T_2 + T_3 = T$ and $3T_3 + 2T_2 \geq 6x$. Note that a packet with path p_1 cannot be sent simultaneously with packets using paths p_2, p_5 or p_6 because of the conflicts on edges $(w_1, w_2), (w_1, v_2)$ and (v_4, w_2) respectively. Note also that packets using paths p_3 and p_4 cannot be sent simultaneously because of the conflict on edge (w_1, w_2). We conclude that a packet using path p_1 can be sent simultaneously with at most one other packet. Since we have x packets using p_1 then $T_2 \geq x$. Then we obtain $3T_3 + 3T_2 \geq 6x + T_2 \geq 6x + x$. Since $T = T_3 + T_2$ is the length of the schedule, then $T \geq 7/3x = 7/6L_{\max}$. □

Lemma 2.5 *For the graph G_{glass} on Figure 2, $\alpha(G_{\text{glass}}) \geq 7/6$.*

Proof. In this graph we create x packets for each of the following six paths $p_1 = \{v_1, v_2, v_3, v_4\}, p_2 = \{v_4, v_2, v_3\}, p_3 = \{v_2, v_4, v_3\}, p_4 = \{v_1, v_2, v_4\}, p_5 = \{v_4, v_3, v_2, v_1\}, p_6 = \{v_3, v_4, v_2, v_1\}$. Then $L_{\max} = 2x$. We now show that any feasible makespan time has length at least $7/3x$. First we create nodes v_3^1, v_3^2, v_1^4 with edges $(v_3, v_3^1), (v_3^1, v_3^2), (v_4, v_4^1)$. Packets following paths p_5 and p_6 initiate in v_3^2 and $v_4^3 1$ respectively in the new graph. As discussed in the proof of Lemma 2.4, it suffices to consider this case. Let T be a feasible makespan time. It is easy to check that no four packets can be sent simultaneously. Let T_3 be the number of times that three packets were sent simultaneously and let T_2 be the number of times that two or one or no packets were sent simultaneously. Then $T_3 + T_2 = T$ and $3T_3 + 2T_2 \geq 6x$. A simple check shows that if three packets are sent simultaneously, then none of this packets uses path p_1. Since we have x packets using this path, then $T_2 \geq x$. Then $3T_3 + 3T_2 = 6x + T_2 \geq 6x + x$. Therefore $T = T_3 + T_2 \geq (7/3)x = (7/6)L_{\max}$. We proved $\alpha(G_{\text{glass}}) \geq 7/6$. □

This completes the proof of Theorem 2.2. □

2.3 Greedy algorithm for trees

In this subsection we analyze the value of $\alpha(G)$ when G is a tree. We already showed in the previous subsection that $\alpha(G) > 1$ for all trees with two or more nodes having degree ≥ 3. We now construct an upper bound.

Theorem 2.6 *For every tree G, $\alpha(G) \leq 2$.*

Proof. We construct the following simple greedy algorithm. Choose any node r of the given tree as a root node. We say that a node is on level i if the distance in the tree from the node to the root is equal to i. Greedy algorithm schedules packets level by level starting from the root. On the first phase the algorithm chooses unscheduled packets intersecting root one by one in any order and starts them in the schedule as soon as possible without conflicts with already scheduled packets. Let p be the last packet scheduled on the first phase we claim that it starts no later than $2L_{\max}-2$. Assume that packet p does not start or end at the root. Let d_p be the distance between the starting vertex of the packet p and the root and let v_{1p} and v_{2p} be vertices in the path corresponding to p just before and after root, respectively. If p starts in some time moment T then for all integers $t \in [0, T-1]$ either the edge (v_{1p}, r) is busy in time interval $[t+d_p-1, t+d_p]$ or the edge (r, v_{2p}) is busy in time interval $[t+d_p, t+d_p+1]$. Since $L_{\max}$ is an upper bound on number of packets intersecting any edge there are at most $L_{\max}-1$ starting points t in interval $[0, T]$ such that we cannot start p at time t because the edge (v_{1p}, r) is busy in time interval $[t+d_p-1, t+d_p]$, analogously there at most $L_{\max}-1$ "bad" starting points t such that edge (r, v_{2p}) is busy in time interval $[t+d_p, t+d_p+1]$. Therefore, $T \leq 2L_{\max}-2$. Similarly, if packet p starts or ends at the root then in the greedy schedule p starts no later then $L_{\max}-1$. On the phase l the algorithm greedily schedules all unscheduled packets intersecting vertices on the level $l-1$. The algorithm takes unscheduled packets intersecting nodes on level l in any order and schedules them as soon as possible. The argument similar with the one for the first phase works also for phase l. So, all packets in the greedy schedule start no later than $2L_{\max}-2$ and therefore the overall makespan of the schedule is at most $2L_{\max}-2+d_{\max}$ where $d_{\max}$ is a length of the longest path in the tree. □

The above analysis clearly can be improved for the greedy algorithm applied to instances such that all packets travel either up or down, i.e. there are no packets intersecting some node traveling from higher levels and going to higher levels again. For such instances we can prove that all packets will start no later than $L_{\max}-1$.

3 Stable hot potato scheduling policies in adversarial queueing networks

Given a graph G let $r(G)$ denote the maximal value of an arrival rate $r \leq 1$ for which there exists a stable hot potato schedule for every adversarial arrival of packets. That is for every pattern of arrivals there exists a hot potato scheduling policy such that the queue length (at the initial nodes of the packet paths) are uniformly bounded by some constant C. Note: we do not know whether this maximal value $r(G)$ is achieved. It is possible that for every $r < r(G)$ there exists a stable policy, but no stable policy exists for some arrival pattern with $r = r(G)$. First of all we prove a lower bound on $r(G)$.

Theorem 3.1 *For every graph G, $r(G) \geq 1/\alpha(G)$, i.e. for all $r < 1/\alpha(G)$ there exists a stable hot potato schedule in G for every adversarial arrival of packets.*

Proof. We prove the theorem by constructing the policy explicitly from static optimal schedules. The policy is of the batch clearing type and is very similar to the one used in [10] for stable adaptive routing of packets. For every path $p \in \mathcal{P}$ let $x_p(0)$ denote the number of packets which require path p and which are present in the system at time $t=0$. Let $x(0) = (x_p(0))_{p\in\mathcal{P}}$. By definition, there exists a hot potato schedule of these packets with makespan time $T_1 \equiv \alpha(G)L_{\max}(x(0)) + o(L_{\max}(x(0)))$, where $L_{\max}(x(0))$ is the maximal congestion corresponding to the vector of packets $x(0)$. We schedule these initial $x(0)$ packets optimally ignoring the packets that arrived later. Let $x(T_1) = (x_p(T_1))$ be the packets present at time T_1. We repeat the procedure. In general, given that $T_0 = 0, T_1, \ldots, T_k$ are defined for some k, let $x(T_k)$ denote the packets present in the system at time T_k. We process these packets optimally ignoring the later arriving packets. The processing will end in some time $T_{k+1} \leq T_k + \alpha(G)L_{\max}(x(T_k)) + o(L_{\max}(x(T_k)))$, where $L_{\max}(x(T_k))$ is the maximal congestion corresponding to the instance $x(T_k)$. We then repeat. Let us prove the stability of this scheduling policy. Note that for every path p, $x_p(T_{k+1}) - x_p(T_k)$ is the number of packets that arrived in $[T_k, T_{k+1})$ and that require path p. Then, by definition, for every edge e

$$\sum_{p:e\in p} (x_p(T_{k+1}) - x_p(T_k)) \leq r(T_{k+1} - T_k).$$

It follows,

$$L_{\max}(x(T_{k+1})) \leq r(T_{k+1} - T_k) \leq r\alpha(G)L_{\max}(x(T_k)) + o(L_{\max}(x(T_k))).$$

By assumption $r\alpha(G) < 1$. It follows that, for some large constant $C > 0$, if $L_{\max}(x(T_k)) > C$, then

$$L_{\max}(x(T_{k+1})) < L_{\max}(x(T_k)).$$

We conclude that the sequence $L_{\max}(x(T_k))$ is bounded. Since $\sum_p x_p(T_k) \leq |E|L_{\max}(x(T_k))$, then $x(T_k)$ is bounded as well. Finally, for every time t, $x_p(t) \leq x_p(T_{k+1})$ where k is the unique integer for which $T_k \leq t < T_{k+1}$. Therefore $x(t) = (x_p(t))$ is bounded as well. This proves $r(G) \geq 1/\alpha(G)$. □

We now want to prove that it is sufficient to construct an infinite series of instances for the static problem such that optimal makespan is at least $\alpha L_{\max}$ to prove that any scheduling policy with $r > 1/\alpha$ is unstable. We prove this statement under some reasonable assumptions on the infinite series of instances. Given a set of paths $p_1, \dots, p_n$ and a set of integers $a_1, \dots, a_n$ consider static hot potato routing problem with γa_i packets using path $p_i, i = 1, 2, \dots, n$, where γ is a large constant. We say that γ is a *multiplicity* of the instance. Let $L_{\max}(\gamma)$ be the maximal congestion of the instance with multiplicity γ. Observe, that $L_{\max}(\gamma) = \gamma L_{\max}(1)$. We say that an instance $(p_1, \dots, p_n, a_1, \dots, a_n)$ is a *high multiplicity example* for the static hot potato routing problem if for any $\gamma \geq \gamma(G)$, it has an optimal makespan $C^* \geq \alpha L_{\max}(\gamma)$ where $\gamma(G)$ is some big constant. Note that all examples constructed in this paper are high multiplicity examples.

Theorem 3.2 *If there exists a high multiplicity example for the static hot potato routing in graph G then the dynamic hot potato routing problem is unstable in G for every $r > 1/\alpha$.*

Proof. Assume that $r = (1+\varepsilon)/\alpha$ for some small positive constant ε. We should built an adversary strategy such that the length of queues in the system is increasing over time no matter which hot potato scheduling policy is used. Assume that the adversary injects packets of type i regularly at intervals $1/r_i$ where

$$r_i = \frac{a_i r}{L_{\max}(1)},$$

i.e. during any time interval of length Δ the adversary injects $r_i\Delta$ packets of type i. The edge rate for this adversary is $\max_e \sum_{i:e\in p_i} r_i = r$ since $\max_e \sum_{i:e\in p} a_i = L_{\max}(1)$.

Consider sufficiently big time interval $[0, T]$ such that the number of packets of each type inserted to the system is much bigger then $|E|$ and $\gamma(G)$. We now will estimate the queue length at time T. Let γ be a number such that $\alpha L_{\max}(\gamma - 1) \leq T < \alpha L_{\max}(\gamma)$. Since $L_{\max}(\gamma) = \gamma L_{\max}(1)$ we know that the number of packets of type i injected to the system is

$$r_i T \geq \frac{a_i r}{L_{\max}(1)} \cdot \alpha L_{\max}(\gamma - 1) = (1+\varepsilon)a_i(\gamma - 1).$$

For any schedule in time interval $[0, T]$ there exists a type i' such that the total number of packets of this type processed in this time interval is at most $a_{i'}\gamma$ since if for all types i we processed more than $a_i\gamma$ packets then the length of such schedule $C^* \geq \alpha L_{\max}(\gamma) > T$. Therefore, the queue length on step T is at least

$$(1+\varepsilon)a_{i'}(\gamma - 1) - a_{i'}\gamma - |E(G)| = \varepsilon a_{i'}\gamma - \varepsilon a_{i'} - |E(G)| = \Omega(\gamma)$$

since $|E(G)|$ packets can be started before T but not finished yet. Therefore the queue length is increasing over time. □

Note, that the theorem above does not prove that $r(G) \leq 1/\alpha(G)$, since the existence of a high-multiplicity example is essential.

We conclude this section with the following corollary, the proof of which follows from Theorems 2.1, 2.2, 3.1, 3.2.

Corollary 3.3 *1. For every graph G, $r(G) = O(\frac{1}{\sqrt{|E(G)|}})$. For a graph $G = G_{\text{grid}}$, depicted on Figure 1, $r(G) = \Omega(\frac{1}{\sqrt{|E(G)|}})$.*

2. Given a graph G, $r(G) = 1$ if and only if G is a cycle or G is a tree with at most one node having degree ≥ 3.

4 Conclusions

We considered the problem of hot potato routing of communication packets in two cases: all the packets are given at time zero (static version) and the packets arrive over time (dynamic version). For the static version we proved that the natural lower bound optimal hot potato makespan time - maximal congestion - may be very weak lower bound. Specifically, we showed that optimal makespan time is $O(\sqrt{|E|})$ for every instance, and for some instances this bound is achieved. We also provided a simple classification of graphs for which optimal makespan time and maximal congestion have the same order of magnitude, when the total number of packets diverges to infinity. Our results have implications in dynamic setting in the sense of maximal arrival rates for which stable hot potato scheduling policy exists.

We conclude with the following set of open problems. Given a graph G can the relative congestion $\alpha(G)$ be computed by any efficient means? Specifically, given a graph G and a value α can we check efficiently whether $\alpha(G) \leq \alpha$? We proved that if G is a tree then $\alpha(G) \leq 2$ and for some trees $\alpha(G) > 7/6$. We conjecture that for certain trees $\alpha(G) = 2$. Closing this gap is an interesting open problem. Better understanding of dynamic hot potato scheduling policies is also of interest.

References

[1] W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosen. Adaptive packet routing for bursty adversarial traffic. *Proc. 30th Ann. ACM Symposium on the Theory of Computing*, 1998.

[2] M. Andrews. Instability of FIFO in session-oriented networks. *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[3] M. Andrews, B. Awerbuch, A. Fernandez, Jon Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention-resolution protocols. *Proc. 37th IEEE Conf. on Foundations of Computer Science*, 1996.

[4] M. Andrews and L. Zhang. The effects of temporary sessions on network performance. *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[5] P. Baran. On distributed communications networks. *IEEE Transactions on Communications*, pages 1–9, 1964.

[6] D. Bertsimas and D. Gamarnik. Asymptotically optimal algorithm for job shop scheduling and packet routing. *Journal of Algorithms*, 33(2):296–318, 1999.

Figure 1: Graph $[0, 2n] \times [-n, n]$. Paths p_2 and p_4.

[7] D. Bertsimas and J. Sethuraman. From fluid relaxations to practical algorithms for job shop scheduling: the makespan objective. submitted.

[8] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. Williamson. Adversarial queueing theory. *Proc. 28th ACM Symposium on Theory of Computing.*To appear in *Journal of the ACM*, 1996.

[9] S. Fiorini and R. Wilson. *Edge-colourings of graphs*, volume distributed by Fearon-Pitman Publishers, Inc., Belmont, Calif. 1977.

[10] D. Gamarnik. Stability of adaptive and non-adaptive packet routing policies in adversarial queueing networks. *Proc. 31st ACM Symposium on Theory of Computing.* To appear in *SIAM Journal on Computing*, 1999.

[11] D. Gamarnik. Using fluid models to prove stability of adversarial queueing networks. *IEEE Transactions on Automatic Control*, 4:741–747, 2000.

[12] A. Goel. Stability of networks and protocols in the adversarial queueing model for packet routing. *Proc. 10th ACM-SIAM Symposium on Discrete Algorithms*, 1999.

[13] N. Hall and C. Sriskandarajah. A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, 4:510–525, 1996.

[14] A. Schuster. *Bounds and analysis techniques for greedy hot-potato routing.* Chapter 11, pages 283-354. Optical interconnections and parallel processing:the interface. Kluwer Academic Publishers, 1997.

Figure 2: Graphs $G_{\mathrm{star}}, G_{\mathrm{bug}}, G_{\mathrm{glass}}$.

[15] S. V. Sevast'janov. On some geometric methods in scheduling theory: a survey. *Discrete Applied Math.*, 55:59–82, 1994.