Optimal Delay Scheduling in Networks with Arbitrary Constraints *

Srikanth Jagabathula
MIT LIDS
77 Massachusetts Avenue
Cambridge, MA 02139
jskanth@mit.edu

Devavrat Shah
MIT LIDS
77 Massachusetts Avenue
Cambridge, MA 02139
devavrat@mit.edu

ABSTRACT

We consider the problem of designing an online scheduling scheme for a multi-hop wireless packet network with arbitrary topology and operating under arbitrary scheduling constraints. The objective is to design a scheme that achieves high throughput and low delay *simultaneously*. We propose a scheduling scheme that – for networks operating under *primary interference constraints* – guarantees a per-flow end-to-end packet delay bound of $^{5d_j}/_{1-\rho_j}$, at a factor 5 loss of throughput, where d_j is the path length (number of hops) of flow j and ρ_j is the effective loading along the route of flow j. Clearly, d_j is a universal lower bound on end-to-end packet delay for flow j. Thus, our result is essentially optimal. To the best of our knowledge, our result is the first one to show that it is possible to achieve a per-flow end-to-end delay bound of $O(\#$ of hops) in a constrained network.

Designing such a scheme comprises two related subproblems: Global Scheduling and Local Scheduling. Global Scheduling involves determining the set of links that will be simultaneously active, without violating the scheduling constraints. While local scheduling involves determining the packets that will be transferred across active edges. We design a local scheduling scheme by adapting the Preemptive Last-In-First-Out (PL) scheme, applied for *quasi-reversible continuous time networks*, to an unconstrained discrete-time network. A global scheduling scheme will be obtained by using *stable marriage algorithms* to emulate the unconstrained network with the constrained wireless network.

Our scheme can be easily extended to a network operating under general scheduling constraints, such as secondary interference constraints, with the same delay bound and a loss of throughput that depends on scheduling constraints through an intriguing "sub-graph covering" property.

*This work was supported in parts by NSF CAREER and NSF collaborative TF grant on flow-level models.

SIGMETRICS'08, June 2–6, 2008, Annapolis, Maryland, USA.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Packet Switching Networks*

General Terms

Algorithms, Design, Performance, Theory

Keywords

Throughput, Delay, Scheduling algorithm

1. INTRODUCTION

A central issue in communication systems is providing various performance guarantees – throughput and delay being the most critical. We consider the problem of designing a scheduling scheme that provides optimal throughput and delay guarantees. These parameters depend on the arrival traffic distribution, network topology and the constraints present in the network. These constraints seriously limit choices and complicate the analysis of any scheduling scheme. Additionally, any online delay optimization scheme that requires learning the arrival traffic rate is impractical. Thus, designing and analyzing scheduling schemes to obtain optimal throughput and delay bounds for a network with arbitrary topology and constraints – without any knowledge of the arrival traffic distribution – is quite challenging. While we may optimize throughput or delay, obtaining a simultaneous performance guarantee will require a trade-off.

We make some natural assumptions for this problem. The network is assumed to be operating under primary interference constraints on simultaneous transmissions i.e., each node can either transmit or receive – but not both – simultaneously. The network is loaded with various traffic flows that are active simultaneously. Each flow consists of a source node, a destination node and a predetermined route through the network. The arrivals of packets for a flow occur according to a continuous-time stochastic process.

Our goal in this paper is to answer the following question: Is it possible to design an online scheduling scheme for a constrained network that achieves high throughput and low delay simultaneously? Specifically, we want to design a scheme that requires no knowledge of arrival traffic rate, and uses only the current state of the network to obtain a delay bound of $O(\#$ of hops) with up to $O(1)$ loss of throughput. Before we give an answer, we discuss some related work addressing various forms of this question.

1.1 Related Work

This question has been of interest for more than two decades. We take note of two of the major relevant results; however, there is a vast literature that addresses various forms of this problem. The two papers we are going to describe subsume most of the relevant work, and hence we shall restrict ourselves to these papers.

Tassiulas and Ephremides (1992) [11] designed a "max-pressure" scheduling scheme that is throughput optimal for a network of *arbitrary topology and constraints*. This work has recently received a lot of attention because of its applicability to a wide class of network scheduling problems [6]. The scheduling scheme they propose is an online scheme that assumes no knowledge of the arrival traffic distribution. In addition to scheduling, their scheme also routes packets in the network. However, the provable delay bounds of this scheme scale with the number of queues in the network. Since the scheme requires maintaining a queue for each flow at every node, the scaling can potentially be very bad. For example, in a very recent work of Gupta and Javidi [7], it was shown through a specific example that such an algorithm can perform very poorly in terms of delay.

Andrews, Fernandez, Harchol-Balter, Leighton and Zhang [1] studied the problem of scheduling flows at each node so as to minimize the *worst-case* delay bounds. They show the existence of a periodic schedule that guarantees a delay bound of $O(d_j + 1/\lambda_j)$, where λ_j is the arrival rate of flow j. Their scheme also guarantees constant size queues at the nodes. Their result is true for a network of arbitrary topology and loaded with traffic such that the arrival rate at every edge is < 1. This delay bound is asymptotically optimal for periodic schedules. It is important to notice that this result gives a per-flow delay bound. This is a big improvement on the previously known bound of $O(c + d)$ [8], where $d = \max d_j$, and $c = \max c_j$ with c_j being the maximum congestion along the route of each flow j. A per-flow end-to-end delay bound is more useful because, otherwise existence of a worst-case flow will seriously deteriorate the bound. However, the model they consider is limited – the only scheduling constraints considered are link capacity constraints, the arrival traffic model considered is non-stochastic, and the scheduling scheme proposed is non-adaptive i.e., does not change with change in the state of the network and requires the knowledge of arrival rates. Further, the dependence over $1/\lambda_j$ is due to the adversarial requirement. As we shall see, in our work, this requirement will be relaxed because of the stochastic nature of traffic. It is worth noting that their delay bound subsumes $1/1-\rho$ style term as constant in $O(\cdot)$ notation.

In summary, these two approaches are inherently limited: [11] has poor delay bounds and [1] considers a limited model. Thus, these approaches cannot be directly extended to the problem we are considering. At the same time, it is worth pointing out that [11] considers routing while we assume knowledge of predetermined routes, and [1] considers worst-case delay bounds while we consider *average* delay bounds.

1.2 Our Contribution

Using an approach that overcomes the aforementioned problems, we propose a scheduling scheme that is both throughput and delay optimal. Our main contribution is a scheduling scheme that guarantees a per-flow end-to-end delay bound of $5d_j/1-\rho_j$, with a factor 5 loss of throughput. Some salient aspects of our result are:

1. Our result provides a simultaneous throughput and delay bound, showing an inherent trade-off, unlike other previous results that concentrated either on throughput or delay.
2. Our result does not require any knowledge of the arrival rate or other traffic statistics. Also, it does not learn them.
3. The delay bound we provide is a per-flow bound and not an aggregate bound. As mentioned earlier, this is much better because otherwise some bad flows can deteriorate the bound.
4. We consider a wireless network with primary interference constraints, but our result easily extends to a network operating under arbitrary constraints. For such a network, the delay bound remains the same (up to a constant factor) with the throughput loss dependent on the constraints (see Lemma 1 and the discussion after it).
5. The scheme is online and adaptive.
6. We utilize the concept of emulation of a network – a powerful technique with diverse applications – to decouple the problem into global and local scheduling schemes.
7. Our implementation schemes are simple i.e., have $O(N^2)$ complexity where N is the number of nodes in the network. Further, they can be implemented in a distributed iterative manner (see Section 8).
8. We assume a deterministic service time for each packet. For discrete-time networks, this is a more feasible assumption than stochastic service times. As will become apparent later, this assumption complicates the analysis further.

Before we describe the model we use and formally state the main results of our paper, we will provide an intuitive explanation and motivation of our approach.

1.3 Our Approach

As mentioned earlier, we first recognize that the problem at hand comprises two related sub-problems: Global Scheduling and Local Scheduling. Global Scheduling corresponds to determining a set of links that may be active simultaneously without violating the scheduling constraints of the network. On the other hand, local scheduling corresponds to determining which packets to move across active edges. In general these two sub-problems are not independent. In order to make the problem tractable, our approach will be to consider the two scheduling sub-problems separately and then put everything together. We note that such decoupling is non-trivial, and requires a careful breaking of the problem into different layers as described next.

Our approach begins with realizing the existence of scheduling schemes for a certain class of networks that guarantee delay bounds of $O(\#$ of hops). Continuous time quasi-reversible networks with a Poisson arrival process and operating under Preemptive Last-In-First-Out (PL) have a product form queue size distribution in equilibrium (see Theorems 3.2 and 3.8 of [3]). We use this fact to prove that such a

network under PL scheduling scheme achieves a delay bound of $O(\#$ of hops). Unfortunately, this scheme cannot be directly applied to our wireless network because we are considering discrete-time constrained scheduling in which fractional packets cannot be transmitted. Thus, this scheme should be adapted to our case, though it is not totally straight forward.

Presence of constraints makes this adaptation complicated. Hence, we first consider a "less-constrained" discrete-time network in which we relax the primary interference constraints. For brevity, we make the following definitions:

DEFINITION 1 ($\mathcal{N}_C$). *Continuous-time quasi reversible network with a Poisson arrival process.*

DEFINITION 2 ($\mathcal{N}_2$). *Discrete-time wireless network operating under primary interference constraints. Namely, in every scheduling phase, each node can either transmit or receive – but not both – at most one packet.*

DEFINITION 3 ($\mathcal{N}_0$). *Discrete-time "less constrained" network in which primary interference constraints are relaxed. Each node can transmit at most one packet in each scheduling phase. There is no restriction on the number of packets each node can receive in each scheduling phase.*

The reason for the definition of $\mathcal{N}_0$ is not very clear right now. But, we shall defer its motivation to the end of this section.

After proving that $\mathcal{N}_C$, operating under PL scheduling scheme, has the required delay bound we adapt this scheme to the discrete-time network $\mathcal{N}_0$. The adaptation we propose will retain the delay bound. This approach of modifying the PL scheme to adapt to a discrete-time network was introduced by El Gamal, Mammen, Prabhakar and Shah in [5]. Thus, we have obtained a scheduling scheme with maximum throughput and the desired delay bound. We have not encountered any throughput-delay trade-off so far, indicating that such a trade-off is the result of the presence of constraints.

Of course, we are not done yet because we need a scheduling scheme for $\mathcal{N}_2$ and not $\mathcal{N}_0$. For that, we utilize the concept of *emulation*, introduced and used in the context of bipartite matching by Prabhakar and McKeown in [9]. Informally, we say that a network $\mathcal{N}$ emulates another network $\mathcal{N}'$, if when viewed as black-boxes the departure processes from the two networks are identical under identical arrival processes. Thus, the delays of a packet in $\mathcal{N}$ and $\mathcal{N}'$ are equal. We propose a mechanism using *stable marriage algorithms* to emulate $\mathcal{N}_0$ with $\mathcal{N}_2$. As we shall show, this requires running the network $\mathcal{N}_2$ at a speedup of 5 i.e., scheduling $\mathcal{N}_2$ 5 times in each time slot. Thus, $\mathcal{N}_2$ running at a speedup of 5 has the required delay characteristics.

Finally, since we have assumed that link capacities are 1, we cannot have 5 schedules in each time slot. Therefore, we establish an equivalence between $\mathcal{N}_2$ operating at speedup 5 and $\mathcal{N}_2$ operating at speedup 1. This equivalence increases the delay by a factor 5 and decreases the throughput by the same factor. Thus, we split the problem into 4 layers and then put everything together. This approach is summarized in Fig. 1.

Further Details: Before we end this section, we resolve the mystery around the constraints of network $\mathcal{N}_0$. We want to consider a constraint set such that the constraints do not restrict our choice of a local scheduling scheme. We note that, if we allow each node to transmit at most one packet in each time slot, a local scheduling scheme automatically determines a set of active links. Thus, we assume that $\mathcal{N}_0$ is constrained such that every node may send at most one packet in each time slot.

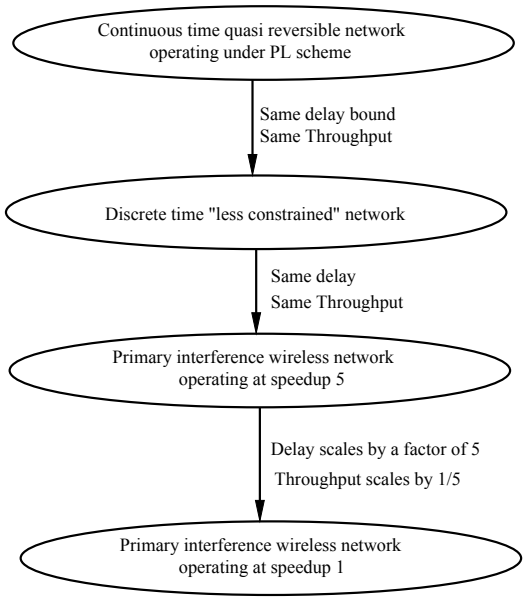

Figure 1: Approach to the design of optimal delay scheduling scheme

The idea of emulation is central to our approach. This concept was introduced by McKeown and Prabhakar in [9] and used by Chuang, Goel, McKeowm and Prabhakar [2] in the context of bipartite matching of switches. The emulation scheme they propose utilizes the stable marriage algorithm and is specific to a single-hop bipartite network with matching constraints. Bipartite graphs with matching constraints have a nice structure, and the emulation scheme they propose does not trivially extend to a general network case with arbitrary constraints. In this paper, we will modify and extend this approach to the case of a general network with arbitrary scheduling constraints. A bipartite graph with matching constraints requires a speedup of 2 for emulation, while we will show that a wireless network with primary interference constraints requires a speedup of 5. If the graph of the network does not contain any odd cycles, then this speedup can be improved to 4. We will also prove the necessity of a speedup of 4 for emulation, implying the optimality of our result. We shall prove the following general result:

LEMMA 1. *For any constrained network $\mathcal{N}$, represented by graph G, let G' denote any sub-graph with maximum degree ≤ 4. Let $\sigma(G')$ denote the minimum number of scheduling phases required to schedule transmissions corresponding to all edges of G', without violating the scheduling constraints. Let σ denote* $\max\{\sigma(G') : G'$ *is degree* 4 *sub-graph of* $G\}$. *Then, there exists an emulation scheme E under which, $\mathcal{N}$ running at a speedup of σ exactly emulates $\mathcal{N}_0$.*

In the case of a wireless network operating under primary interference constraints, no two edges incident on the same

node can be simultaneously active. Vizing's theorem states that a graph with degree Δ can be colored using either Δ or $\Delta+1$ colors. Since the sub-graph in our case has degree 4, Vizing's theorem guarantees the sufficiency of 5 scheduling phases. Thus, it follows from Lemma 1 that a speedup of 5 is sufficient for emulation of $\mathcal{N}_0$ by $\mathcal{N}_2$.

In a network with secondary interference constraints, two directed edges (t_1, r_1) and (t_2, r_2) cannot be simultaneously active if either (t_1, r_2) or (t_2, r_1) is an edge in the network. Thus, every edge has conflicts with at most $4\Delta^2$ edges, where Δ is the degree of the network. Therefore, in this case, a speedup of $4\Delta^2$ is sufficient. Using similar arguments, we can extend our emulation scheme to a network of arbitrary constraints.

2. MODEL AND NOTATION

We consider a constrained communication network consisting of N nodes, connected by M links. The connectivity of the system is represented by a directed graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. The graph is assumed to be connected. A packet may enter or depart the network at any node. Each entering packet is routed through the network to its destination node. There are J classes of traffic flows. Each flow j corresponds to a source node, a destination node and a route through the network.

The routing matrix of the network, denoted by R, represents the connectivity of nodes in different routes. It is an $M \times J$ matrix and the element r_{mj}, corresponding to m^{th} edge and j^{th} route is:

$$r_{mj} = \begin{cases} 1, & \text{if route } j \text{ contains edge } m \\ 0, & \text{otherwise} \end{cases}$$

We assume that external packet arrivals occur according to a continuous-time stochastic process, but the packets become available for transmission only at integral times (say at the end of the time slot). We also assume that the inter-arrival times are i.i.d. with mean $1/\lambda_j$ (i.e., rate λ_j) and finite variance.

The arrival rate vector $\underline{\lambda} = \{\lambda_j : j = 1, \ldots, J\}$ consists of the arrival rates of all classes of traffic. The aggregate traffic flow vector $\underline{\tau} = \{\tau_m : m = 1, \ldots, M\}$, which represents the congestion in each link, is defined as:

$$\underline{\tau} = R\underline{\lambda}$$

Let $Q_n(t)$ denote the length of the queue at node n, $n = 1, \ldots, N$. The vector $\underline{Q}(t) = \{Q_n(t) : n = 1, \ldots, N\}$ comprises the length of the queues at all nodes. $\underline{Q}(t)$ is called the *queue length process*. The network system is said to be *stable* if the queue length process $\underline{Q}(t)$ forms a positive recurrent Markov Chain.

A link is said to be active if it is scheduled to move a packet in that step. An *activation set* is a set of links that may be active simultaneously. The activation set is represented by an *activation vector* $\mathbf{s}$. It is a binary vector with M elements, with the m^{th} element corresponding to the m^{th} edge, and is equal to 1 if the m^{th} edge is in the activation set and to 0 otherwise. The *constraint set* $\mathcal{S}$ consists of all the activation vectors of the system. This set completely specifies the constraints of the network system. We assume that the $\mathbf{0} \in \mathcal{S}$. We will primarily be concerned with primary interference constraints. A node in a network operating under primary interference constraints may either transmit or receive but not both, at most one packet in each scheduling phase. We denote such a constraint set by $\mathcal{S}_2$. We use $\mathcal{S}_0$ to denote the constraint set corresponding to $\mathcal{N}_0$.

We say that an arrival rate vector $\underline{\lambda}$ is feasible, if there exists a scheduling scheme that results in a queue length process that is positive recurrent under the above model. Let $\Lambda(\mathcal{S})$ denote the set of all feasible arrival rate vectors $\underline{\lambda}$. Let $\Lambda'(\mathcal{S})$ denote its interior. It is easy to prove that $\Lambda(\mathcal{S})$ is a convex set (in fact related to the convex hull of $\mathcal{S}$ through the matrix R).

Let notation $v \in j$ or $j \in v$ denote that route j passes through node v. Let λ_j be the rate at which packets are arriving on route j and let $f_v = \sum_{j:j\in v} \lambda_j$ be the net rate at which data is arriving at node v.

We define effective loading $\rho_j(\underline{\lambda})$ along a flow j as:

$$\rho_j(\underline{\lambda}) = \max_{v \in j} f_j$$

$D_\Sigma(\underline{\lambda})$ denotes the average delay experienced in a network operating with a scheduling scheme Σ and arrival rate vector $\underline{\lambda}$. The delay of a packet in the network is defined as the number of time steps it takes for the packet to reach its destination node after it arrives at the network. Let $D_\Sigma^{ji}(\underline{\lambda})$ denote the delay of the i^{th} packet of traffic flow j, $j = 1, \ldots, J$, under scheduling scheme Σ and arrival rate vector $\underline{\lambda}$. Then, the sample mean of the delay over all packets is:

$$D_\Sigma^j(\underline{\lambda}) = \limsup_{k\to\infty} \frac{1}{k} \sum_{i=1}^{k} D_\Sigma^{ji}(\underline{\lambda})$$

When equilibrium distribution exists, $D_\Sigma^j(\underline{\lambda})$ is just the expected delay of the packet along the path of flow j. When the context is clear, we also denote D_Σ^j simply by D^j. We also note here that the length of the path d_j is a universal lower bound on the delay.

3. MAIN RESULTS

We now state the main results of our paper:

THEOREM 2. *Consider a network $\mathcal{N}_2$ with a Poisson arrival rate vector $\underline{\lambda}$. If $\underline{\lambda} \in \frac{1}{5}\Lambda'(\mathcal{S}_0)$ then there exists a scheduling scheme Σ^* that is constant factor delay optimal, i.e, $D_{\Sigma^*}^j(\underline{\lambda}) \le \frac{5d_j}{1-\rho_j(5\underline{\lambda})}$, for every flow j.*

Theorem 2 is true for a network with arbitrary topology. In addition, if the network does not contain any odd cycles, then the constant factor 5 in the theorem can be improved to 4. We will also prove the necessity of a speedup 4 for our approach, in Lemma 11 in Section 7, implying the essential optimality of our result.

The theorem assumes that the arrival process is Poisson. This is in no way restrictive because every arrival process can be converted into a Poisson process using the procedure of "Poissonization" described in Section 9.

We shall briefly comment on the factor $1/1-\rho_j(5\underline{\lambda})$ appearing in the delay bound. A factor of the form $1/1-\rho_j(\underline{\lambda})$ is inevitable in the delay bound. At a first glance, it might seem that $1/1-\rho_j(5\underline{\lambda})$ can be arbitrarily smaller than $1/1-\rho_j(\underline{\lambda})$. While this is true, we should realize that the capacity region is shrinking by a factor 5 and hence effective loading in the network will be a multiple of 5.

In accordance with the approach we described earlier, this theorem will be a consequence of Lemma 1 and the following two Lemmas:

LEMMA 3. *Consider a network $\mathcal{N}_0$ with Poisson arrival rate vector $\underline{\lambda} \in \Lambda'(\mathcal{S}_0)$. There exists a scheduling scheme Σ_0 such that the per-flow end-to-end delay of flow j, denoted by D^j, in the network $\mathcal{N}_0$ is bounded as:*

$$D^j \leq \frac{d_j}{1 - \rho_j(\underline{\lambda})}$$

LEMMA 4. *Consider a network $\mathcal{N}'$ with an arrival traffic of rate $\underline{\lambda}$ and running at a speedup of σ. Let $\mathcal{N}$ denote the same network running at speedup 1. Suppose there is a scheduling scheme Σ for $\mathcal{N}'$ that guarantees a delay of D^j for flow j. Then, the same scheduling scheme Σ results in a delay of σD^j for $\mathcal{N}$, operating with an arrival rate of $\underline{\lambda}/\sigma$.*

In the theorem and lemmas stated above, the proof is going to be constructive i.e., we are going to prove the existence of Σ^* and Σ_0 by explicitly constructing them and then proving that they possess the stated properties.

We will now prove Theorem 2 and postpone the proofs of Lemmas 1, 3 and 4 to later sections. The proof of the theorem will essentially be the approach we described earlier, but with more formalism.

PROOF OF THEOREM 2. Consider a network $\mathcal{N}_0$ operating with a Poisson arrival process of rate $5\underline{\lambda}$. Since, $\underline{\lambda} \in \frac{1}{5}\Lambda'(\mathcal{S}_0)$, we have $5\underline{\lambda} \in \Lambda'(\mathcal{S}_0)$. Thus, it follows from Lemma 3 that $\exists$ a scheduling scheme Σ_0 such that the per-flow end-to-end delay of flow j, denoted by D^j, is bounded by $\frac{d_j}{1-\rho_j(5\underline{\lambda})}$.

Under the primary interference constraints of $\mathcal{N}_2$, no two edges incident on the same node can be active simultaneously. Hence, it follows from Vizing's theorem that a speedup of 5 is sufficient to schedule any degree 4 sub-graph of $\mathcal{N}_2$. Thus, Lemma 1 tells us that $\exists$ an emulation scheme E under which $\mathcal{N}_2$ running at speedup 5 can exactly emulate $\mathcal{N}_0$.

We now combine Σ_0 and E to obtain a scheduling scheme Σ^* that guarantees a delay bound of $\frac{d_j}{1-\rho_j(5\underline{\lambda})}$ for $\mathcal{N}_2$ running at a speedup of 5. By Lemma 4, this scheme results in a delay bound of $\frac{5d_j}{1-\rho_j(5\underline{\lambda})}$ for $\mathcal{N}_2$ operating with an arrival traffic of rate $\underline{\lambda}$ and speedup 1.

This completes the proof of the theorem. □

4. ORGANIZATION

The rest of the paper is organized as follows: Section 5 establishes the equivalence of $\mathcal{N}_2$ operating at speedup 5 and $\mathcal{N}_2$ operating at speedup 1 and proves Lemma 4. Section 6 describes the properties of $\mathcal{N}_C$, adapts the PL scheme to $\mathcal{N}_0$ and proves Lemma 3. Section 7 describes a mechanism to emulate $\mathcal{N}_0$ with $\mathcal{N}_2$. It also proves Lemma 1 and provides a counter-example to prove the necessity of speedup of 4 for our approach. These three sections are independent of each other and can be read in any order. Section 8 discusses the running time complexity of our emulation schemes. Section 9 describes "Poissonization" procedure and finally Section 10 concludes.

5. SPEEDUP = LOSS OF THROUGHPUT

Under the assumption that the capacity of the links is 1, we cannot run a network at a speedup > 1. Hence, we need to establish a one-one correspondence between a network running at a speedup 1 and a network running at a higher speedup. This will enable us to implement a scheduling scheme that requires the network to run at a higher speedup. We establish this equivalence through the proof of Lemma 4.

PROOF OF LEMMA 4. A network running at speedup σ is similar to a network running at speedup 1, when the time is dilated by a factor of σ. In this dilated time frame, the arrivals to the network occur no more than once in every σ time slots. This dilation of arrivals can be achieved by sending the packets of each flow into the network of speedup 1 through an external buffer. The service process of the buffer is such that, it sends out all the packets in the buffer once every σ time slots.

We establish a correspondence between the two networks $\mathcal{N}$ and $\mathcal{N}'$ as follows: Let $A_j(t)$ denote the number of arrivals of flow j, $j = 1, 2, \ldots, J$, occurring at node s_j in time slot t. Let $\Delta_j(t)$ denote the number of packets departing from the external buffer for flow j, in time slot t. By construction, $\Delta_j(t) = 0$ for $t \neq k\sigma$, for some integer k and $\Delta_j(t) = \sum_{i=1}^{\sigma} A_j((k-1)\sigma+i)$ for $t = k\sigma$. The arrival process to $\mathcal{N}'$ is denoted by $A'_j(t)$ and is defined as $A'_j(t) = \Delta_j(\sigma t)$. Thus, the arrival rate to network $\mathcal{N}'$ is $\sigma\frac{1}{\sigma}\underline{\lambda} = \underline{\lambda}$. With this construction, $\mathcal{N}$ is just $\mathcal{N}'$ running in a dilated time frame. Hence, we can use the same scheduling scheme in both the networks. When the same scheduling scheme is applied to both the networks, the departure process from $\mathcal{N}$ will be a σ-dilated version of the departure process from $\mathcal{N}'$, i.e., a packet that departs from $\mathcal{N}'$ in time slot t will depart from $\mathcal{N}$ in time slot σt. Thus, the delay experienced by a packet in $\mathcal{N}$ is σ times the delay experienced in $\mathcal{N}'$.

This completes the proof of this lemma. □

6. SCHEDULING POLICY

In this section we will design a scheduling scheme Σ_0 that achieves a delay bound of $\frac{d_j}{1-\rho_j(\underline{\lambda})}$. As mentioned before, we leverage the results from a continuous-time quasi-reversible network with a Poisson arrival process and operating under PL queue management policy. We shall denote such a network by $\mathcal{N}_C$. We will first describe some of the properties of $\mathcal{N}_C$ and then prove that this network indeed has the desirable delay properties. Once we prove that, we will modify the PL scheme to obtain a scheduling scheme Σ_0 for the discrete-time network $\mathcal{N}_0$. There is a one-one correspondence between $\mathcal{N}_0$ and $\mathcal{N}_C$ operating with the same Poisson arrival process and under PL and Σ_0 scheduling schemes respectively. We prove that the delay of a packet in $\mathcal{N}_0$ operating under Σ_0 is bounded above by the delay of the corresponding packet in $\mathcal{N}_C$ operating under PL scheme. The desired delay properties of $\mathcal{N}_0$ operating under Σ_0, immediately follow from this bound.

6.1 Properties of $\mathcal{N}_C$

We consider a continuous-time open network of queues $\mathcal{N}_C$, with the same topology and external arrival process as $\mathcal{N}_0$. We assume that the service process of each server is deterministically equal to 1 and a Preemptive LIFO (PL) policy is used at each server. (Chapter 6.8 of [10]) The queue size distribution for the continuous time network $\mathcal{N}_C$ with PL queue management has a product form in equilibrium (see Theorems 3.7 and 3.8 of [3]) provided the following conditions are satisfied: (a) the service time distribution is either phase-type or the limit of a sequence of phase-type

distributions and (b) total traffic at each server is less than its capacity. In our case, the second condition is trivially satisfied. For the first condition, we note that the sum of n exponential random variables each with mean $\frac{1}{n}$ has a phase-type distribution and converges in distribution to a constant random variable, as n approaches infinity.

We now state and prove a theorem that bounds the expected delay experienced along each flow. Let D^j denote the net delay of a packet traveling on route j, $j = 1, 2, \ldots, J$, in equilibrium. Let notation $v \in j$ or $j \in v$ denote that route j passes through node v. Let λ_j be rate at which packets are arriving on route j and let $f_v = \sum_{j:j\in v} \lambda_j$ be the net rate at which data is arriving at node v.

THEOREM 5. *In the network $\mathcal{N}_C$, in equilibrium*

$$\mathbb{E}[D^j] = \sum_{v:v\in j} \frac{1}{1-f_v}.$$

PROOF. The network $\mathcal{N}_C$ described above is an open network with Poisson exogenous arrival process. Each queue in the network is served as per LIFO-PL policy. The service requirement of each packet at each queue is equal to unit. That is, the service requirement of all packets are i.i.d. with bounded support. Therefore, Theorem 3.10 [3] implies that the network $\mathcal{N}_C$ is an open network of quasi-reversible queues. Therefore, by Theorem 3.8 [3] this network has product-form distribution in equilibrium. Further, the marginal distribution of each queue in equilibrium is the same as if the queue is operating in isolation. For example, consider queue at node v, whose size is denoted by Q_v (in equilibrium). In isolation, Q_v is distributed as if the queue has Poisson arrival process of rate f_v. The queue is quasi-reversible and hence it has distribution (by Theorem 3.10 [3]) such that

$$\mathbb{E}[Q_v] = \frac{f_v}{1-f_v}.$$

Let Q_v^j be the number of packets at node v of type j in equilibrium. Then, another implication of Theorem 3.10 [3] implies that

$$\mathbb{E}[Q_v^j] \;=\; \frac{\lambda_j}{f_v}\mathbb{E}[Q_v] \;=\; \frac{\lambda_j}{1-f_v}.$$

By Little's Law applied to the stable system concerning only packets of route j at node v, we obtain that the average delay experienced by packets of route j at node v, $\mathbb{E}[D_v^j]$, is such that

$$\lambda_j \mathbb{E}[D_v^j] = \mathbb{E}[Q_v^j].$$

That is,

$$\mathbb{E}[D_v^j] = \frac{1}{1-f_v}.$$

Therefore, the net delay experienced by packet along route j, D^j is such that

$$\mathbb{E}[D^j] = \sum_{v:v\in j} \mathbb{E}[D_v^j] \;=\; \sum_{v:v\in j} \frac{1}{1-f_v}.$$

This completes the proof of the Theorem. □

6.2 Scheduling Scheme Σ_0

As mentioned before, the PL policy in $\mathcal{N}_C$ cannot be directly implemented in $\mathcal{N}_0$ because complete packets have to be transferred in $\mathcal{N}_0$ unlike fractional packets in $\mathcal{N}_C$ and packets generated at time t become eligible for service only at time $\lceil t \rceil$.

Hence, we need to adapt the PL policy to our discrete-time case. This adaptation was introduced and used in [5]. We will now present the adaptation of PL scheme and the proof of a lemma relating the delays in both networks, that is described in [5].

The PL queue management scheme is modified to using a Last-In-First-Out (LIFO) scheduling policy using the arrival times in $\mathcal{N}_C$. As described above, we assume that $\mathcal{N}_C$ has the same topology and exogenous arrival traffic pattern as $\mathcal{N}_0$. Let the arrival time of a packet at some node in $\mathcal{N}_C$ be α_C and in $\mathcal{N}_0$ at the same node be α_0. Then, it is served in $\mathcal{N}_0$ using a LIFO policy with the arrival time as $\lceil \alpha_C \rceil$ instead of α_0.

This scheduling policy makes sense only if each packet arrives before its scheduled departure time in $\mathcal{N}_0$. According to this scheduling policy, the scheduled departure time can be no earlier than $\lceil \alpha_C \rceil$, whereas the actual arrival time is α_0. Hence, for this scheduling policy to be feasible, it is sufficient to show that $\alpha_0 \leq \lceil \alpha_C \rceil$ for every packet at each node. Let δ_C and δ_0 denote the departure times of a packet from some node in $\mathcal{N}_C$ and $\mathcal{N}_0$ respectively. Since the departure time at a node is the arrival time at the next node on the packet's route, it is sufficient to show that $\delta_0 \leq \lceil \delta_C \rceil$ for each packet in every busy cycle of each node in $\mathcal{N}_C$. This will be proved in the following lemma.

LEMMA 6. *Let a packet depart from a node in $\mathcal{N}_C$ and $\mathcal{N}_0$ at times δ_C and δ_0 respectively. Then $\delta_o \leq \lceil \delta_C \rceil$.*

PROOF. The proof will be through induction on the length of the busy cycle k. Fix a server and a busy cycle of length k of $\mathcal{N}_C$. Let it consist of packets numbered $1, \ldots, k$ with arrivals at times $\alpha_1 \leq \ldots \leq \alpha_k$ and departures at times $\delta_1, \ldots, \delta_k$. Let the arrival times of these packets in $\mathcal{N}_0$ be $A_1, \ldots, A_k$ and departures be at times $\Delta_1, \ldots, \Delta_k$. By assuming that $A_i \leq \lceil \alpha_i \rceil$ for $i = 1, \ldots, k$, we need to show that $\Delta_i \leq \lceil \delta_i \rceil$ for $i = 1, \ldots, k$.

Clearly this holds for $k = 1$ since $\Delta_1 = \lceil A_1 \rceil + 1 \leq \lceil \alpha_1 \rceil + 1 = \lceil \delta_1 \rceil$. Now suppose it holds for all busy cycles of length k and consider any busy cycle of $k+1$ packets.

If $\lceil \alpha_1 \rceil < \lceil \alpha_2 \rceil$, then because of the LIFO policy in $\mathcal{N}_D$ based on times α_i, we have $\Delta_1 = \lceil \alpha_1 \rceil + 1 \leq \lceil \alpha_1 \rceil + k + 1 = \lceil \delta_1 \rceil$. The last equality holds since in $\mathcal{N}_C$, the PL service policy dictates that the first packet of the busy cycle is the last to depart. Also, the remaining packets would have departure times as if they are from a busy cycle of length k.

Otherwise if $\lceil \alpha_1 \rceil = \lceil \alpha_2 \rceil$ then the LIFO policy in $\mathcal{N}_D$ based on arrival times α_i results in $\Delta_1 = \lceil \alpha_1 \rceil + k + 1 = \lceil \delta_1 \rceil$ and the packets numbered $2, \ldots, k$ depart exactly as if they belong to a busy cycle of length k. This completes the proof by induction. □

We now prove Lemma 3 using Lemma 6

PROOF OF LEMMA 3. Suppose $\mathcal{N}_0$ is operating under a LIFO scheduling policy using the arrival times in $\mathcal{N}_C$. Let's call this scheduling scheme Σ_0. Let D^j and D_C^j denote the expected delays of a packet along flow j in $\mathcal{N}_0$ and $\mathcal{N}_C$ respectively. Then, it follows from Lemma 6 that $D^j \leq D_C^j$. Theorem 5 tells us that:

$$D_C^j = \sum_{v:v\in j} \frac{1}{1-f_v}$$

Therefore

$$D^j \leq \sum_{v:v\in j} \frac{1}{1-f_v}$$

By definition, $\frac{1}{1-f_v} \leq \frac{1}{1-\rho_j(\underline{\lambda})}$, $\forall v \in j$. Hence,

$$D^j \leq \frac{d_j}{1-\rho_j(\underline{\lambda})}$$

This completes the proof of this lemma. □

7. EMULATION SCHEME

In this section we will prove Lemma 1 by describing an emulation scheme E, for a network $\mathcal{N}$ with a general constraint set, to emulate $\mathcal{N}_0$. This problem is not tractable when considered in such generality. Hence, as before, our approach will be to carefully break the problem into layers, in order to make it more tractable. For that, we will first consider the problem of designing an emulation scheme E', to emulate $\mathcal{N}_0$ by a network with d-matching (directed matching) constraints. By d-matching constraints, we mean that any node can either send or receive or simultaneously send and receive at most one packet in each scheduling step. We shall denote such a network by $\mathcal{N}_1$. It will turn out that $\mathcal{N}_1$ requires a speedup of 2 to emulate $\mathcal{N}_0$. We will state this result as a the following lemma:

LEMMA 7. *A speedup of* 2 *is sufficient to emulate* $\mathcal{N}_0$ *using* $\mathcal{N}_1$.

Using this lemma we will now prove Lemma 1.

PROOF OF LEMMA 1. By Lemma 7, $\mathcal{N}_1$ can emulate $\mathcal{N}_0$ by running at a speedup of 2. Thus, there are 2 scheduling phases in each time slot. In each of the scheduling phases, the network of active edges of $\mathcal{N}_1$ forms a sub-graph of max-degree at most 2. When combined, it forms a sub-graph G' of max-degree at most 4. A network with arbitrary set of constraints may require more than two scheduling phases to schedule the edges in G'. With $\sigma = \max\{\sigma(G') : G'$ is degree 4 sub-graph of G$\}$, it follows that operating the network at a speedup of σ should be sufficient for emulation. □

7.1 Emulation Scheme E'

We will now describe the emulation scheme E', and prove that a speedup of 2 is sufficient to emulate $\mathcal{N}_0$ by $\mathcal{N}_1$. As mentioned before, bipartite graphs have a nice structure and it is easier to design a scheduling scheme for such graphs. Our approach will be to design a scheduling scheme for a bipartite graph and then translate to our network $\mathcal{N}_1$. For that, we will establish a correspondence between $\mathcal{N}_1$ and a bipartite graph. As we shall show shortly, in this correspondence, the constraints present in $\mathcal{N}_1$ will just translate to matching constraints in the bipartite graph; more specifically no node in the bipartite graph can be matched to more than one node.

In order to utilize bipartite graphs for scheduling, we will construct a $2N$ node bipartite graph from the given N node network $\mathcal{N}_1$. We shall call this bipartite graph the "equivalent bipartite graph of $\mathcal{N}_1$." Suppose that the network is represented by the graph $G(\mathcal{V},\mathcal{E})$. We construct the equivalent bipartite graph as follows: Take the N nodes of the original graph and call them transmitters. Thus, T_i corresponds to node i in the network. Make copies of these N nodes, introducing an additional N nodes, and call them receivers. Now, R_i corresponds to node i in the network. Complete the construction of the bipartite network by introducing an edge between nodes T_i and R_j if and only if $(i,j) \in \mathcal{E}$.

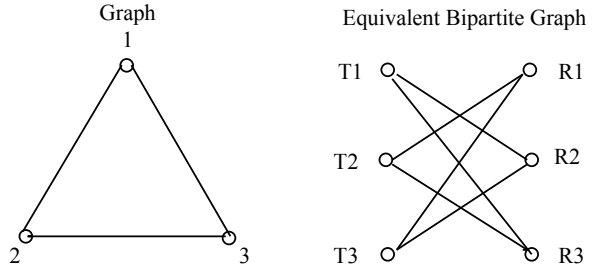

Figure 2: Construction of an equivalent bipartite graph

As we shall see shortly, the queuing mechanism used at each of the nodes plays a crucial role in the emulation scheme. In fact, intelligent queue management at each node is the reason why the emulation scheme works. Each node will maintain three queues of packets: *arrival queue, relay queue* and *departure queue.* In each scheduling step, every packet arriving to a node is pushed into the arrival queue. The only exceptions are packets that reach their destination node, where they are pushed into the departure queue. In each time slot, exactly one packet is moved from the arrival queue and pushed into the relay queue. Only packets present in the relay queue are scheduled, i.e., in each scheduling step the packet present at the head of the relay queue is moved across the active link.

It is now clear that our emulation scheme essentially comprises determining the set of active edges in each of the scheduling phases and determining which packet to move from the arrival queue and push into the relay queue in each time slot. Before we can describe the emulation scheme, we need to establish notation and definitions.

Notation: We shall denote the departure time of a packet p from node v in network $\mathcal{N}_0$ by $D(p,v,0)$ and the departure from the same node in $\mathcal{N}_1$ by $D(p,v,1)$. Similarly, $A(p,v,0)$ and $A(p,v,1)$ denote the arrival times of the packet p at node v in networks $\mathcal{N}_0$ and $\mathcal{N}_1$ respectively.

With this notation we shall formally define emulation as follows:

DEFINITION 4 (EMULATION). *Consider two networks* $\mathcal{N}$ *and* $\mathcal{N}'$ *operating under the same arrival process i.e., for any packet* p, $A(p,v_s,0) = A(p,v_s,1)$, *where* v_s *is the source node of packet* p. *Then, we say that* $\mathcal{N}'$ *emulates* $\mathcal{N}$ *if and only if* $D(p,v_d,1) = D(p,v_d,0)$, *where* v_d *is the destination node of the packet.*

The following definitions are crucial to the scheme that we are going to describe:

DEFINITION 5 (CUSHION). *Suppose a packet* p *is located in the relay queue of node* v. *Denote the next hop of* p *by* v'. *Then, the cushion of* p *is defined as the number of packets* p' *located in the arrival queue of* v' *such that* $D(p',v',0) < D(p,v',0)$.

DEFINITION 6 (PRIORITY). *Consider a packet* p *located in the relay queue of node* v. *Then, its priority is defined as*

the number of packets that are ahead of it in the relay queue of v.

DEFINITION 7 (SURPLUS). *The surplus of a packet p, denoted by $\eta(p)$, is defined as the difference between its cushion and priority, i.e., $\eta(p)$ = Cushion of p − Priority of p.*

DEFINITION 8 (SCHEDULE PRIORITY LIST). *Consider a packet p located in the relay queue of node v. Let v' denote its next hop. A packet p' is present in the Schedule Priority List of p if and only if $D(p', v', 0) < D(p, v', 0)$.*

This completes the definitions and notation that we require. We will now describe the emulation scheme E'.

Emulation Scheme E': One of the tasks of the emulation scheme is moving a packet from the arrival queue and pushing it into the relay queue. This comprises choosing a packet from the arrival queue and determining where to insert in the relay queue. We now describe a procedure for this. In each time slot t, the packet p in the arrival queue of node v, that satisfies $D(p, v, 0) = t$ will be chosen to be moved from the arrival queue to the relay queue. This packet will be inserted into the relay queue using the Urgent Cell First (UCF) Insertion policy. According to this policy, a packet being moved from an arrival queue will be inserted into the relay queue such that its surplus is non-negative. This is achieved by determining the packet's cushion just before insertion and then pushing it such that its priority ≤ cushion.

The only aspect of the emulation scheme left to be designed is the policy to determine the set of active links in each of the scheduling phases. As mentioned earlier, we will first describe a scheduling scheme for the "equivalent bipartite graph of $\mathcal{N}_1$". Using the correspondence described above, the schedule obtained for the bipartite graph will be mapped back to the original network.

The equivalent bipartite graph of $\mathcal{N}_1$ will be scheduled using stable marriage algorithms. In each of the scheduling phases, we use the stable marriage algorithm to come up with a stable matching of the bipartite graph. We define stable matching as follows: A matching of the transmitters and the receivers is called stable matching if for every packet p in one of the relay queues, one of the following hold:

1. Packet p is transferred to its next hop.
2. A packet that is ahead of p in its relay queue is scheduled.
3. A packet that is in the Schedule Priority List of p is transferred to the next hop of p.

This completes the description of the emulation scheme.

A stable matching can be obtained using Stable Marriage Algorithms. Gale and Shapely [4] gave an algorithm that finds a stable matching. We shall further discuss determination of stable matchings in Section 8.

7.2 Proof of Lemma 7

In each time slot t, the emulation scheme requires us to move a packet p that satisfies $D(p, v, 0) = t$, from the arrival queue to the relay queue of node v. This makes sense only if the packet p arrives at node v of network $\mathcal{N}_1$ before $D(p, v, 0)$ i.e., for every packet p we have $A(p, v, 1) \leq D(p, v, 0)$. If this condition is true for every node, then the packet p will arrive at its destination node v_d in $\mathcal{N}_1$ before its scheduled departure from $\mathcal{N}_0$. Then, we can send the packet p from $\mathcal{N}_1$ in time slot $D(p, v_d, 0)$, thus perfectly emulating $\mathcal{N}_0$. Hence, it is sufficient to prove that $A(p, v, 1) \leq D(p, v, 0)$, for every packet at every node, to prove Lemma 7.

Thus, Lemma 7 is a direct consequence of the following lemma:

LEMMA 8. *For every packet p of $\mathcal{N}_1$, operating under the emulation scheme E', we have*

$$A(p, v, 1) \leq D(p, v, 0)$$

for every node v on its path.

The result of the following lemma will be required for the proof of Lemma 8 and hence we shall prove it first.

LEMMA 9. *For every packet p in $\mathcal{N}_1$ operating under E', its surplus $\eta(p) \geq 0$.*

PROOF. Suppose that the packet p is present in the relay queue of node v. When the packet enters the relay queue of the node, UCF insertion policy ensures that its surplus is non-negative. Therefore, it is sufficient to prove that the surplus of the packet is non-decreasing with time. We will prove that surplus is non-decreasing by essentially inducting on time t.

To establish this, we will consider the change in $\eta(p)$ during a time slot t. During each of the scheduling phases of $\mathcal{N}_1$, either the priority decreases by 1 or the cushion increases by 1 or both occur (because of stable matching). Thus, at the end of the two scheduling phases $\eta(p)$ would have increased at least by 2. When a packet is moved from the arrival queue and pushed into the relay queue, priority may increase by 1 or cushion may decrease by 1 or both an increase in priority and decrease in cushion may occur. Thus, during this $\eta(p)$ decreases at most by 2. Thus, by the end of the time slot t, $\eta(p)$ does not decrease and hence $\eta(p)$ is non-decreasing. This completes the proof of this lemma. □

PROOF OF LEMMA 8. We use induction on time t. As the induction hypothesis, we assume that $A(p, v, 1) \leq D(p, v, 0)$ for all packets p such that $D(p, v, 0) \leq t$. For $t = 1$, all packets p in the network $\mathcal{N}_1$ that satisfy $D(p, v, 0) \leq 1$ are at their respective source nodes at the beginning of time slot 1. Thus, the condition $A(p, v, 1) = A(p, v, 0) \leq D(p, v, 0)$ is satisfied. Thus, the induction hypothesis is true for the base case.

Now assume that it is true for some t. We want to prove that any packet p that satisfies $D(p, v, 0) = t+1$ also satisfies $A(p, v, 1) \leq D(p, v, 0)$.

When v is the source node, we have $A(p, v, 1) = A(p, v, 0) \leq D(p, v, 0)$ and hence the result is true. Now, suppose that v is the not the source node of the packet p. Let's suppose that the packet is located at node v' in network $\mathcal{N}_1$ at the beginning of time slot $t + 1$. If v' is either v or one of its later hops then $A(p, v, 1) < t + 1 = D(p, v, 0)$ and hence we are done. If this is not true, then v' has to be one of the previous hops along the path of p. Let v'' denote the hop of p just before arriving at node v. Since, $D(p, v, 0) = t + 1$, we should have $D(p, v'', 0) < t + 1$. Thus, it follows from induction hypothesis that $A(p, v'', 1) \leq D(p, v'', 0) < t + 1$. Thus, if v' is neither v nor one of its later hops, then it has to be v''.

Since $A(p, v'', 1) \leq D(p, v'', 0) < t + 1$, the packet p will be located in the relay queue of node v'' at the beginning of

time slot $t+1$. The cushion of packet p is clearly 0. Lemma 9 tells us that the surplus is always non-negative. Thus, the priority of p should also be 0 i.e., p is at the head of the relay queue of node v''. Every packet p' present in the Schedule Priority List of p is either already at v or moved beyond v. To see why, consider a packet p' in the list. Then, by definition it should satisfy $D(p', v, 0) < D(p, v, 0) = t+1$ and hence already would have been transferred to or beyond the arrival queue of node v, by induction hypothesis. Thus, the packet p is at the head of its relay queue and there is no packet p' in its Schedule Priority List that can be moved to v. Therefore, the stable marriage algorithm schedules this packet p in time slot $t+1$ and moves it from the relay queue of v'' to the arrival queue/departure queue of node v. Thus, $A(p, v, 1) = t+1 \leq D(p, v, 0)$.

The result of the lemma follows by induction. □

7.3 Optimality of the scheme

In this section, we will prove the necessity of a speedup of 4 for the emulation of $\mathcal{N}_0$ using $\mathcal{N}_2$. Taking into account the sufficiency of speedup 4 for graphs without odd cycles, this essentially implies the optimality of our result for $\mathcal{N}_2$. The necessity result will be proved through a carefully constructed counter-example that requires a speedup of at least 4 for *any* emulation scheme. We will first construct a counter-example for a bipartite graph with d-matching constraints, that requires a speedup of 2 for emulation of $\mathcal{N}_0$. This will prove the necessity of speedup 2 for emulation of $\mathcal{N}_0$ using $\mathcal{N}_1$. The counter example for the bipartite graph will then be extended to network $\mathcal{N}_2$.

The counter example we are going to construct is a non-trivial modification and extension of the counter example given in [2]. The counter example of [2] proves the necessity of a speedup of 2 for the special case of single-hop bipartite networks. In the counter-example to be constructed, we will use fractional speedups. In fact, we are going to prove the necessity of a speedup of $4 - \frac{1}{N}$ for $\mathcal{N}_2$. A fractional speedup of $4 - \frac{1}{N}$ means that there are $4N - 1$ scheduling steps for every N time slots.

We state the result of this section as the following two lemmas:

LEMMA 10. *An $8N$ node network $\mathcal{N}_1$ requires a speedup of at least $2 - \frac{1}{4N}$ for the emulation of a FIFO $\mathcal{N}_0$.*

LEMMA 11. *A $12N$ node network $\mathcal{N}_2$ requires a speedup of at least $4 - \frac{1}{N}$ to exactly emulate a FIFO $\mathcal{N}_0$.*

PROOF OF LEMMA 10. Consider a $4N \times 4N$ complete bipartite graph. As before, call one set of $4N$ nodes transmitters and the other set of $4N$ nodes receivers. T_i and R_i, for $i = 1, \ldots, 4N$, denote transmitter and receiver nodes respectively. Packets arrive to the network only at transmitter nodes and leave the network only at receiver nodes. This is a single-hop network. We assume that the network is operating under d-matching constraints, i.e., in each scheduling phase, each transmitter can transmit at most one packet and each receiver can receive at most one packet. Thus, the set of active edges should form either a partial or complete matching. In the corresponding "less constrained" network, $\mathcal{N}_0$, of this bipartite graph, each receiver can receive more than one packet in a scheduling phase. Thus, all the packets arriving to the network are immediately moved from transmitters and queued at the receivers.

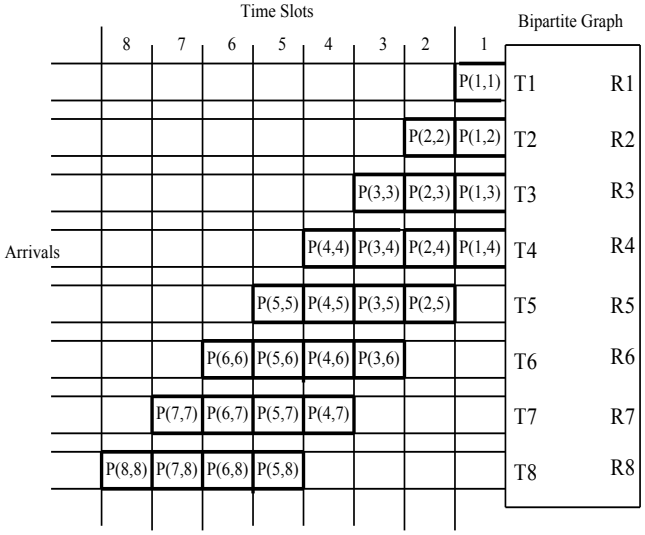

Figure 3: Arrival traffic pattern for the case $N = 2$

We will now describe an arrival traffic pattern for the bipartite graph which results in a speedup of $2 - \frac{1}{4N}$. This non-integral speedup corresponds to having one truncated time slot out of every $4N$ time slots. The truncated time slot contains only one scheduling phase, whereas the $4N - 1$ non-truncated time slots contain two scheduling phases. We also assume that the scheduling algorithm does not know in advance whether a time slot is truncated or not.

The arrival traffic pattern that gives the lower bound is as follows: The traffic pattern spans $4N$ time slots, the last of which is truncated. Each packet is labeled as $\mathrm{P}(i, j)$, where i denotes the receiver node to which it is destined and j denotes the time of departure from the corresponding "less constrained" network.

In the i^{th} time slot, where $i = 1, 2, \ldots, 4N - 3$, the transmitter nodes T_i to $\mathrm{T}_{(i+3)}$ receive packets, all of them destined to the receiver node R_i. In the corresponding less constrained network $\mathcal{N}_0$, these packets are immediately moved and queued at the node R_i. We assume that the packet arriving at T_j in time slot i, $j = i, i+1, i+2, i+3$ leaves the network $\mathcal{N}_0$ in time slot j. Thus, according to our notation, this packet will be denoted by $\mathrm{P}(i, j)$. Also, in time slot i, node T_i has the packet of lowest time to leave.

For $i = 4N - 2, 4N - 1$ and $4N$, the transmitter nodes with the lowest time to leave in each of the previous $i - 1$ time slots do not receive any more packets. In addition, the rest of the transmitter nodes receive packets, all destined for the same receiver node, R_i. The traffic pattern is depicted in Fig. 3 for the case $N = 2$.

This traffic pattern can be repeated as many times as required to create arbitrarily long traffic patterns. We now look at the scheduling order of the packets for this traffic pattern.

Let's first consider the case of $N = 1$. The $N = 1$ case is exactly similar to the 4×4 switch counter example described in Appendix A of [2]. Thus, as described in that paper, there is only one possible scheduling order as shown in Fig. 4. The speedup required in this case is equal to $\frac{7}{4}$ which is equal to $2 - \frac{1}{4}$.

We now consider the $N = 2$ case. Using arguments similar

Phase	R1	R2	R3	R4
1	P(1,1)			
2	P(1,2)			
3	P(1,3)	P(2,2)		
4	P(1,4)	P(2,3)		
5		P(2,4)	P(3,3)	
6			P(3,4)	
7				P(4,4)

Figure 4: Scheduling order for the given arrival traffic pattern for $N = 1$

Phase	R1	R2	R3	R4	R5	R6	R7	R8
1	P(1,1)							
2	P(1,2)							
3	P(1,3)	P(2,2)						
4	P(1,4)	P(2,3)						
5		P(2,4)	P(3,3)					
6		P(2,5)	P(3,4)					
7			P(3,5)	P(4,4)				
8			P(3,6)	P(4,5)				
9				P(4,6)	P(5,5)			
10				P(4,7)	P(5,6)			
11					P(5,7)	P(6,6)		
12					P(5,8)	P(6,7)		
13						P(6,8)	P(7,7)	
14							P(7,8)	
15								P(8,8)

Figure 5: Scheduling order for the given arrival traffic pattern for $N = 2$

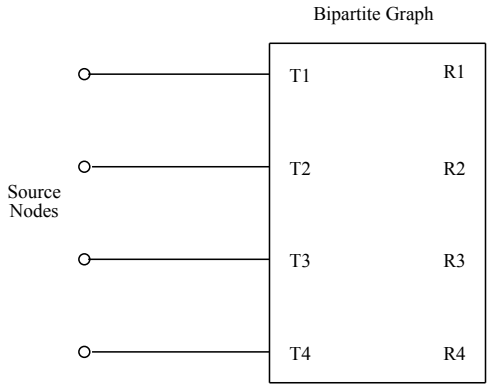

Figure 6: A 4×4 bipartite graph with transmitter nodes connected to source nodes

to the $N = 1$ case, we conclude that there is only one possible scheduling order as shown in Fig. 5. Thus, a speedup of $\frac{15}{8}$ or $2 - \frac{1}{8}$ is required for exact emulation.

This procedure can be repeated for an arbitrary N to obtain a minimum speedup of $2 - \frac{1}{4N}$, thus proving the lemma. □

Proof of Lemma 11. We now extend the previous counter example to a network with $\mathcal{N}_2$ constraints. For that, consider a $12N$ node network obtained by connecting each of the transmitter nodes of the $4N \times 4N$ bipartite graph to an isolated node using a single edge. This is shown in Fig. 6

Phase	From Source Nodes	R1	R2	R3	R4
1	P(1,1)P(1,2) P(1,3)P(1,4)				
2	P(2,2)P(2,3)	P(1,1)			
3	P(2,4)	P(1,2)			
4	P(3,3)P(3,4)	P(1,3)	P(2,2)		
5		P(1,4)	P(2,3)		
6	P(4,4)		P(2,4)	P(3,3)	
7				P(3,4)	
8					P(4,4)

Figure 7: Scheduling order for the given arrival traffic pattern for a bipartite graph with source nodes and operating under d-matching constraints

for $N = 1$. Call the newly added nodes source nodes. We assume that this is a two-hop network, in which, packets arrive only at source nodes and depart the network only from receiver nodes. We also assume that the network is operating under d-matching constraints, i.e., each node can either transfer or receive or simultaneously transfer and receive at most one packet during each scheduling phase.

Suppose that the arrival traffic pattern applied to this network is same as that described above for the bipartite graph, the only difference being that now the packets arrive at the source nodes instead of arriving at the corresponding transmitter nodes. The output traffic pattern for the corresponding less constrained network $\mathcal{N}_0$, will be identical to the previous case except for a delay of one time slot.

We claim that a minimum speedup of $2 - \frac{1}{4N}$ is required for emulation of $\mathcal{N}_0$. For that, we assume that the first time slot is truncated i.e., has only one scheduling phase. The subsequent distribution of scheduling phases is the same as before i.e., one truncated time slot for every $4N$ time slots. In the first time slot, all the packets at the source nodes are transferred to their corresponding transmitter nodes. In fact, in every time slot, the packets arriving at the source nodes can be transferred to the transmitter nodes, in one of the scheduling phases, independently of the scheduling order. Thus the state of the bipartite graph in the network, from the second time slot onwards, is exactly the same as that of the bipartite graph described above. Hence, there can only be one scheduling order as described above, with everything delayed by one time slot. This is depicted in Fig. 7. Thus, a minimum speedup of $2 - \frac{1}{4N}$ is required for precise emulation of $\mathcal{N}_0$. We note that, the additional time slot required in the beginning will lead to arbitrarily small change in the speedup for arbitrarily long traffic patterns, and hence we neglect it.

We now carry out another modification to the above $12N$ node network to obtain another $12N$ node network as follows: The source nodes and the edges going out of them are kept intact. The $4N \times 4N$ bipartite graph in the network is replaced by a $4N$ node complete graph. The edges coming out of each of the source nodes are connected to each of the $4N$ nodes of the network. We then add $4N$ additional nodes and connect each of them to each of the $4N$ nodes of the complete graph. Fig. 8 shows the network for $N = 1$.

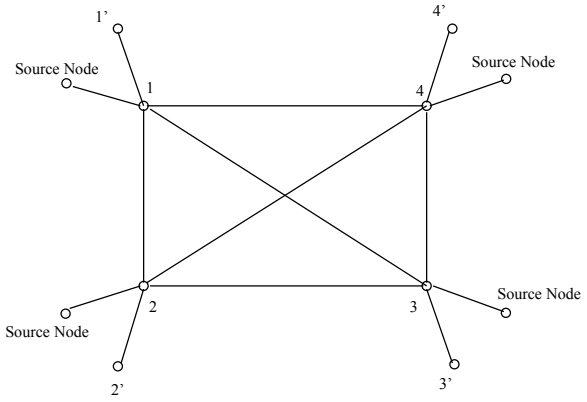

Figure 8: Topology of the $12N$ node network to which arrival traffic pattern is applied

We will consider the scheduling of this network by establishing a one-one correspondence between this network and the $12N$ node network with a bipartite graph we've just described. The source nodes and the edges coming out of them, of both the networks are mapped to each other. The $4N$ transmitter nodes, $\mathrm{T}_1, \mathrm{T}_2, \ldots, \mathrm{T}_{4N}$, of the bipartite graph are mapped to the $4N$ nodes of the complete graph. The receiver nodes $\mathrm{R}_1, \mathrm{R}_2, \ldots, \mathrm{R}_{4N}$ are also mapped to the nodes of the complete graph as follows: for $i = 1, \ldots, 4N-1$, $\mathrm{R}_i \rightarrow (i+1)$ and $\mathrm{R}_{4N} \rightarrow 1$. The additional $4N$ nodes will be labeled $1', 2', \ldots, 4N'$, with the node i of complete graph connected to node i', for $i = 1, 2, \ldots, 4N$. We shall call these additional nodes primed nodes. None of the nodes of the network with bipartite graph will be mapped to these nodes. Since we have mapped both the transmitter and receiver nodes to the same set of $4N$ nodes of the complete graph, translation of an arrival traffic pattern from the network with bipartite graph to the network with complete graph might result in packets to be transferred from a node to itself. In such a scenario, we assume that the packet is transferred from node i to node i'.

We assume that the arrival traffic pattern described for the bipartite graph is applied to this network. It is easy to see that, with $\mathcal{N}_0$ constraints, this network will have the same output traffic pattern. Therefore, from the argument given for the bipartite graph and the one-one correspondence, it follows that a speedup of at least $2-\frac{1}{4N}$ is necessary for this network operating under d-matching constraints to emulate its corresponding less constrained network. Since the transmitter and receiver nodes are mapped to the same set of nodes of the complete graph, the schedules obtained may not be directly implementable for this network with $\mathcal{N}_2$ constraints. We now argue that this network with $\mathcal{N}_2$ constraints will require a speedup of at least $4 - \frac{1}{N}$ for exact emulation of $\mathcal{N}_0$.

We prove the claim for the case $N = 1$. Proof of the general case is very similar. The scheduling order for this network with d-matching constraints is shown in Fig. 7. Fig. 9, Fig. 10 and Fig. 11 depict the active edges during time slots 2, 3 and 4 respectively. It is easy to see from the figures that this network with d-matching constraints will require a speedup of 2 in time slots 2, 3 and 4, whereas network with $\mathcal{N}_2$ constraints requires a speedup of 3 in time slot 2 and a speedup of 4 in time slots 3 and 4. This makes

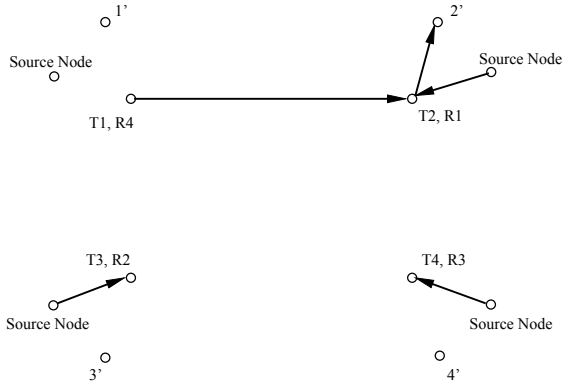

Figure 9: Network of active edges during time slot 2

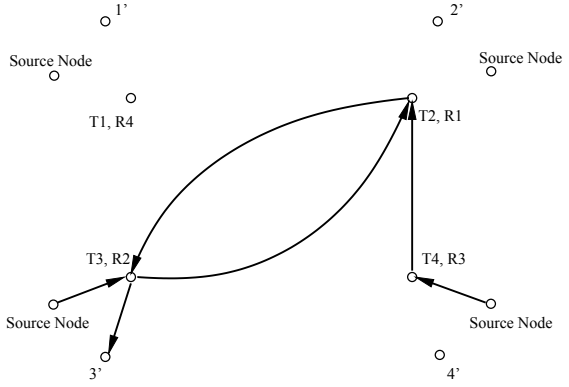

Figure 10: Network of active edges during time slot 3

the net speedup $\frac{7}{4}$ or $2 - \frac{1}{4}$ for $\mathcal{N}_1$ and 3 or $4 - \frac{1}{1}$ for $\mathcal{N}_2$. Since the scheduling order of Fig. 7 is the only possible scheduling order implementable in network with d-matching constraints, it follows that the same scheduling order is the only possible scheduling order for $\mathcal{N}_2$ and hence a speedup of 3 is necessary.

This argument can be easily extended to a general case and thus we have proved that a speedup of $4-\frac{1}{N}$ is necessary for the precise emulation of a FIFO output queued network.

□

8. IMPLEMENTATION

We now discuss some of the implementation issues of our scheme. The ensuing discussion will demonstrate the feasibility of implementation. It should be kept in mind that our result is a proof of concept, and by no means we are suggesting that it is practically implementable in its current form.

Recall that there are two main 'sub-routines' that are utilized by our algorithm:(a) Simulation of $\mathcal{N}_C$ to obtain the arrival times for scheduling $\mathcal{N}_0$; (b) Stable marriage algorithm for determining a stable matching. It is easy to verify that the computation complexity of these dominate the overall complexity of the algorithm.

We assume that simulation of $\mathcal{N}_C$ is being carried out by a control layer in the network. Preemptive last-in-first-out

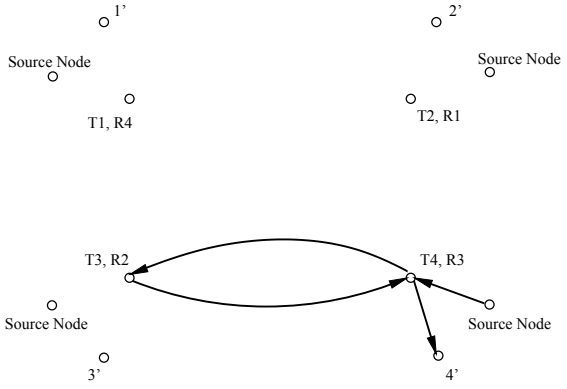

Figure 11: Network of active edges during time slot 4

scheme is carried out in this control layer, but without any exchange of actual packets. This can be implemented in a distributed fashion. This exchange of control information would require $O(N)$, where N is the number of nodes in the network, operations per time slot.

Now, the stable marriage algorithm. As mentioned earlier, [4] describe a simple, iterative algorithm to determine a stable matching in $O(N)$ iterations. Each node requires $O(1)$ operations to determine preferences (from corresponding less constrained network). Thus, a total of $O(N^2)$ operations are required per scheduling phase.

Putting everything together, our emulation scheme requires $O(N)$ iterations and $O(N^2)$ operations per scheduling phase.

9. POISSONIZATION

Theorem 2 assumed that the arrival process is Poisson. This is not restrictive because any arrival process can be converted into a Poisson process through "Poissonization." Poissonization can be carried out as follows:

Poissonization. : Consider an arrival process of rate λ to a node. The arrivals are queued in a separate "preprocessing" buffer and the departures from the buffer are fed into the network. The buffer is serviced according to a Poisson process of rate μ. A packet is sent out for every service token generated, when the buffer is non-empty. When the buffer is empty, a dummy packet is sent out for every service token, thus ensuring a departure process that is Poisson with rate μ.

Clearly, preprocessing buffer is a $G/M/1$ queue and has a finite expected queue length for $\mu > \lambda$. Kingman's bound implies that the delay of the $G/M/1$ queue scales like $\Theta(\frac{1}{\rho_j(\underline{\mu})-\rho_j(\underline{\lambda})})$ because the mean and variance of the arrival process are finite. Due to convexity of the region $\Lambda(\mathcal{S})$, it is possible to choose a $\underline{\mu}$ such that $\underline{\mu} > \underline{\lambda}$, $1-\rho_j(\underline{\mu}) = \Theta(1-\rho_j(\underline{\lambda}))$ and $\rho_j(\underline{\mu}) - \rho_j(\underline{\lambda}) = \Theta(1-\rho_j(\underline{\lambda}))$. Therefore, our delay bound changes by only by an additive term of $O(\frac{1}{1-\rho_j(\underline{\lambda})})$.

10. CONCLUSION

In this paper we have provided a simple scheduling scheme, through the concept of emulation, that guarantees a per-flow end-to-end packet delay of $\frac{5d_j}{1-\rho_j(5\underline{\lambda})}$ at factor 5 throughput loss, for wireless networks with primary interference constraints. Thus, it settles the much debated recent question of achieving a delay that is of the order of the number of hops, with maybe some loss of throughput.

Our approach extends to a network operating under arbitrary scheduling constraints. For a general network, our scheme achieves the same delay bound (up to constant factors), with a loss of throughput that depends on the scheduling constraints through an intriguing "sub-graph covering" property. For the primary interference constraints, Vizing's theorem allows us to determine this constant as 5 for a general graph. However, understanding this for an arbitrary constraint set is of general interest.

Our result requires a constant factor loss of throughput to achieve the desired delay bound. Establishing the optimality of our approach for a network with arbitrary set of constraints is a natural open problem.

11. REFERENCES

[1] M. Andrews, A. Fernandez, M. Harchol-Balter, T. Leighton, and L. Zhang. General dynamic routing with per-packet delay guarantees of o(distance + 1/session rate). *SIAM Journal on Computing*, 30(5):1594–1623, 2000.

[2] S. Chuang, A. Goel, N. McKeown, and B. Prabhakar. Matching output queueing with a combined input/output-queued switch. *Selected Areas in Communications, IEEE Journal on*, 17(6):1030–1039, 1999.

[3] F.P.Kelly. *Reversibility and Stochastic Networks*. John Wiley and Sons Ltd., New York, 1979.

[4] D. Gale and L. Shapley. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[5] A. E. Gamal, J. Mammen, B. Prabhakar, and D. Shah. Optimal Throughput-Delay Scaling in Wireless Networks - Part II: Constant-Size Packets. *IEEE Transactions on Informaition theory*, 52(11):5111–5116, 2006.

[6] L. Georgiadis, M. Neely, and L. Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking*, 1(1):1–144, 2006.

[7] P. Gupta and P. Javidi. Towards delay-optimal routing in ad-hoc networks. In *Proc. Asilomar Conference on Signals Systems and Computers*, Pacific Grove CA USA, Nov.4–7 2007.

[8] F. Leighton, B. Maggs, and S. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–186, 1994.

[9] B. Prabhakar and N. McKeown. On the speedup required for combined input and output-queued switching. *Automatica*, 35(12):1909–1920, 1999.

[10] R.W.Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice-Hall, 1988.

[11] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1949, 1992.