On the Min-Max-Delay Problem:
NP-completeness, Algorithm, and Integrality Gap

Qingyu Liu*, Lei Deng†, Haibo Zeng*, Minghua Chen†
*Department of Electrical and Computer Engineering, Virginia Tech, USA
†Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong

I. INTRODUCTION

Delay-sensitive network flows have strong applications in many domains, including communication networks, cyber-physical systems, transportation networks and evacuation planning [1]. In communication networks, video conferencing requires the video delivery delay to be no more than 250ms to ensure a good interactive conferencing experience [2], and packet delivery delay highly affects user experience and hence revenue in data center networks of cloud service providers like Amazon, Microsoft and Google [3], [4]. In cyber-physical systems, shorter delay for control messages can improve control quality [5]. In transportation networks, timely delivery is critical to deliver perishable goods [6]. In evacuation planning, it is important to move all people from hazardous areas to safe areas as soon as possible [7].

Theoretically, there are mainly two delay-sensitive flow models: the flow-amount model and the flow-rate model. The flow-amount model is suitable to the applications where the flow is generated once, while the flow-rate model is suitable to the applications where the flow is generated continuously. For both models, two different delay-sensitive network flow problems are important: maximizing the flow (amount or rate) subject to a maximum delay constraint, and minimizing the maximum delay subject to a flow (amount or rate) requirement. We summarize related studies in Tab. I.

TABLE I

<table>
<thead>
<tr>
<th>DELAY-SENSITIVE NETWORK FLOW PROBLEMS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow-amount Model</td>
</tr>
<tr>
<td>Flow-rate Model</td>
</tr>
<tr>
<td>Max-flow subject to delay constraint</td>
</tr>
<tr>
<td>Min-max delay subject to flow requirement</td>
</tr>
</tbody>
</table>

| Flow-amount Model                        |
| Flow-rate Model                          |
| [8], [12]                                |
| [10], [14]                               |
| [11], [15], our paper                   |

For the flow-rate model, the first problem, called delay-constrained max-flow problem [10], is to maximize the flow rate to be sent from a source to a sink while the end-to-end delay is bounded above by a given delay constraint. The problem has been proved to be NP-complete [10]. The second problem, called maximum latency problem [11], is to minimize the maximum end-to-end delay that flow units experience from a source to a sink while satisfying a given flow rate requirement and has been proved to be NP-complete [11].

This paper studies the Min-Max-Delay problem, which is similar to the maximum latency problem [11], [15] except for two key differences:

• In our Min-Max-Delay problem, each edge has an integer capacity such that the assigned flow rate cannot exceed the given capacity, while there is no capacity constraint in the maximum latency problem;
• Edge delay is an integer in Min-Max-Delay. But it is a flow-dependent function in the maximum latency problem.

These two differences can capture many applications that cannot be handled by the maximum latency problem. For example, in the communication networks, the flow-dependent queueing delay becomes negligible and thus the constant propagation delay dominates the edge delay in the light load scenario [16]. In the transportation networks, the ground vehicle speed (or equivalently the time to pass the road) remains nearly constant before reaching a certain flow rate [17]. Due to these two key differences, existing results on the maximum latency problem including the complexity analysis and approximation algorithms [11], [15] are not applicable to our problem.

In this paper, we make the following contributions:

• We prove that Min-Max-Delay is weakly NP-complete in Thm. 4 based on the results in Sec. III-A and Sec. IV.
• We propose a binary-search algorithm which can find the optimal solution to Min-Max-Delay in pseudo-polynomial time in Sec. IV. The time complexity of the algorithm is \( O(\log(N d_{\text{max}})(N^3 d^5_{\text{max}})(\log R + N^2 d_{\text{max}} \log(N^2 d_{\text{max}}))) \), where \( N \triangleq \max\{|V|, |E|\} \), \( R \) is the rate requirement, and \( d_{\text{max}} \) is the maximum edge delay. The complexity is pseudo-polynomial in the sense that it is polynomial in the numeric

Abstract—We study a delay-sensitive information flow problem where a source streams information to a sink over a directed graph \( G \triangleq (V, E) \) possibly using multiple paths to minimize the maximum end-to-end delay, denoted as the Min-Max-Delay problem. We prove that Min-Max-Delay is weakly NP-complete, and demonstrate that it becomes strongly NP-complete if we require integer flow solution. We propose an optimal pseudo-polynomial time algorithm for Min-Max-Delay, with time complexity \( O(\log(N d_{\text{max}})(N^3 d^5_{\text{max}})(\log R + N^2 d_{\text{max}} \log(N^2 d_{\text{max}}))) \), where \( N \triangleq \max\{|V|, |E|\} \), \( R \) is the flow rate requirement and \( d_{\text{max}} \) is the maximum edge delay. Besides, we show that the integrality gap, which is defined as the ratio of the maximum delay of an optimal integer flow to the maximum delay of an optimal fractional flow, could be arbitrarily large.
value of the problem input $d_{\text{max}}$, but is exponential in the bit length of the problem input, i.e., $\log(d_{\text{max}})$ [18].

$\triangleright$ We prove Min-Max-Delay that becomes strongly NP-complete if each path can only have an integer flow in Thm. 2 of Sec. III-B.

$\triangleright$ In Sec. V, we further construct a network to show that the integrality gap, which is defined as the ratio of the maximum delay of an optimal integer flow to the maximum delay of an optimal fractional flow, could be arbitrarily large. This example illustrates additional intriguing difficulties for the integer version of Min-Max-Delay problem.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a network modeled as a directed graph $G = (V, E)$ with $|V|$ nodes and $|E|$ edges. We define $N = \max\{|V|, |E|\}$. Each edge $e \in E$ has a non-negative integer capacity $c_e$ and a non-negative integer delay $d_e$. We define $d_{\text{max}} = \max_{e \in E} d_e$. A source node $s \in V$ needs to send a positive integer rate $R$ to a sink node $t \in V \setminus \{s\}$.

We denote $P$ as the set of all paths from $s$ to $t$. For any path $p \in P$, we denote its path delay as $d_p \triangleq \sum_{e \in p} d_e$. A flow solution $f$ is defined as the assigned flow rate over $P$, i.e., $f \triangleq \{ f_p : f_p \geq 0, p \in P \}$. For a flow solution $f$, we define $f_e \triangleq \sum_{p \in E, e \in p} f_p$ as the flow rate on edge $e \in E$. We further define the maximum delay of a flow solution $f$ as

$$
D(f) \triangleq \max_{p \in P, f_p > 0} d_p,
$$

i.e., the maximum delay among paths that carry positive rates.

We consider the problem of finding a flow solution $f$ to minimize the maximum delay $D(f)$ while satisfying both the rate requirement and edge capacity constraints. We denote the problem as Min-Max-Delay. It is formulated as

$$
\begin{align*}
\min \quad & D \\
\text{s.t.} \quad & \sum_{p \in P} f_p = R, \tag{2a} \\
& f_e = \sum_{p \in P, e \in p} f_p \leq c_e, \quad \forall e \in E, \tag{2b} \\
& f_p (d_p - D) \leq 0, \quad \forall p \in P, \tag{2c} \\
& f_p \geq 0, \quad \forall p \in P, \tag{2d}
\end{align*}
$$

where (2a) together with (2d) define our objective to minimize the maximum path delay for $s$–$t$ paths that carry positive rates (called flow-carrying paths). Constraint (2b) restricts that the source $s$ sends $R$ rate to the sink $t$, and constraint (2c) requires that the flow rate on edge $e$ does not exceed its capacity $c_e$.

From formulation (2) we observe two difficulties to solve Min-Max-Delay: (i) The number of paths (number of variables) can exponentially increase w.r.t. the network size, and (ii) formulation (2) is non-convex due to constraint (2d).

The integer version of problem Min-Max-Delay, denoted as Int-Min-Max-Delay, further requires that each path carries an integer flow rate, i.e., replacing (2e) by

$$
\text{vars.} \quad f_p \in \mathbb{Z}^+, \quad \forall p \in P. \tag{3}
$$

We let $f_{\text{MM}}$ (resp. $f_{\text{IMM}}$) be an optimal flow solution to problem Min-Max-Delay (resp. Int-Min-Max-Delay). Then, we define the integrality gap as

$$
\text{Int-Gap} \triangleq \frac{D(f_{\text{MM}})}{D(f_{\text{IMM}})},
$$

which is the maximum delay ratio of the integer flow solution $f_{\text{IMM}}$ to the possibly fractional flow solution $f_{\text{MM}}$.

III. NP-COMPLETENESS

In this section, we analyze the computational complexity of our two problems, Min-Max-Delay and Int-Min-Max-Delay. In Sec. III-A, we prove that Min-Max-Delay is NP-complete based on the polynomial reduction from the NP-complete partition problem [18]. In Sec. III-B, we further prove that Int-Min-Max-Delay is NP-complete in the strong sense based on the pseudo-polynomial transformation from the classic strongly NP-complete 3-partition problem [18].

A. NP-completeness for Min-Max-Delay

To analyze the computational complexity of Min-Max-Delay, we first define partition and the partition problem.

Definition 1 (Partition). Given a non-empty set $A$, its partition is a set of non-empty subsets such that each element in $A$ is in exactly one of these subsets.

Definition 2 (Partition Problem [18]). Given a set of $n$ positive integers $A = \{a_1, a_2, \ldots, a_n\}$ with sum $\sum_{i \in A} a_i = 2b$. Is there a partition $\{A_1, A_2\}$ of $A$ such that $\sum_{a_i \in A_1} a_i = \sum_{a_j \in A_2} a_j = b$?

The partition problem is known to be NP-complete [18] (in the weak sense). We now leverage it to prove that Min-Max-Delay is NP-complete.

Theorem 1. The decision version of Min-Max-Delay problem is NP-complete.

Proof: For any partition problem we construct a graph $G'$ with $(2n + 1)$ nodes and $3n$ edges as in Fig. 1. All edges have unit capacity. Each edge $(w_{i-1}, w_i)$ has a delay of $a_i$ for all $i = 1, \ldots, n$, while edges $(w_{i-1}, v_i)$ and $(v_i, w_i)$ have a delay of zero. Obviously it takes polynomial time to construct the graph $G'$. We then consider the following decision problem of Min-Max-Delay: for graph $G'$ with source $s = w_0$, sink $t = w_n$, and flow rate requirement $R = 2$, is there any feasible flow $f$ such that the maximum delay $D(f) \leq b$?

Now we prove the partition problem answers “Yes” if and only if the decision version of Min-Max-Delay answers “Yes”. If Part. If the decision problem of Min-Max-Delay answers “Yes”, then there exists a flow $f$ such that $D(f) \leq b$. Since $f$ is feasible, the total rate from $w_0$ to $w_n$ in $f$ is $R = 2$. Now
due to the capacity constraint and flow conservation, all edges must exactly have a flow rate 1 to satisfy the requirement \( R = 2 \). The total delay in flow \( f \) is

\[
\sum_{p \in P} f_p d_p = \sum_{e \in E} f_e d_e = \sum_{e \in E} 1 \cdot d_e = \sum_{i=1}^{n} a_i = 2b. \tag{5}
\]

Since \( D(f) \leq b \), we have

\[
d_p \leq D(f) \leq b, \forall p \in P \text{ with } f_p > 0. \tag{6}
\]

Also, because the total flow rate is equal to 2, we have

\[
2b = \sum_{p \in P} f_p d_p = \sum_{p \in P} f_p \leq b \sum_{p \in P} f_p = 2b. \tag{7}
\]

As both ends in (7) are the same, it must be

\[
d_p = b, \forall p \in P \text{ with } f_p > 0. \tag{8}
\]

Therefore, all flow-carrying paths have a path delay of \( b \). We choose an arbitrary flow-carrying path \( p \). Since all solid edges have a delay of 0, the path delay of \( p \) is the delay of all dashed edges that belong to \( p \). We consider the set \( A_1 \) that contains \( a_i \) if edge \((w_{i-1}, w_i) \in p \). Clearly, it holds that \( \sum_{a_i \in A_1} a_i = b \). We then define \( A_2 = A \setminus A_1 \). It shall be \( \sum_{a_i \in A_2} = \sum_{a_i \in A} - \sum_{a_i \in A_1} = 2b - b = b \). \( A_1 \) and \( A_2 \) are thus a partition of set \( A \) and meet the requirement of the partition problem. Hence, the partition problem answers “Yes”.

**Only If Part.** If the partition problem answers “Yes”, then there exists a partition \( A_1 \) and \( A_2 \) such that \( \sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i = b \). We now construct two paths \( p_1 \) and \( p_2 \).

- \( \forall i \in [1, n] \), if \( a_i \in A_1 \), we put edge \((w_{i-1}, w_i) \) into path \( p_1 \); otherwise, we put \((w_{i-1}, v_i)\) and \((v_i, w_i)\) into \( p_1 \).
- \( \forall i \in [1, n] \), if \( a_i \in A_2 \), we put \((w_{i-1}, w_i) \) into path \( p_2 \); otherwise, we put \((w_{i-1}, v_i)\) and \((v_i, w_i)\) into \( p_2 \).

Due to the definition of a partition (see Definition 1), \( A_1 \) and \( A_2 \) are two disjoint sets, i.e., \( A_1 \cap A_2 = \emptyset \). Thus, we can easily see that \( p_1 \) and \( p_2 \) are two disjoint \( s-t \) path, i.e., \( p_1 \) and \( p_2 \) do not share any common edge. We can see that \( d_{p_1} = \sum_{a_i \in A_1} \) and \( d_{p_2} = \sum_{a_i \in A_2} \). We then construct the flow \( f \) with only two flow-carrying paths \( p_1 \) and \( p_2 \) and set \( f_{p_1} = f_{p_2} = 1 \). Since \( p_1 \) and \( p_2 \) are disjoint, the capacity constraint is satisfied. Also, since \( f_{p_1} + f_{p_2} = R = 2 \), the rate requirement is satisfied. Thus \( f \) is a feasible flow with maximum delay \( D(f) = b \). Therefore, the decision problem of Min-Max-Delay answers “Yes”.

Since the partition problem is NP-complete [18] and the reduction can be done in polynomial time, the decision problem of Min-Max-Delay is also NP-complete.

Due to the NP-completeness, it is impossible to solve Min-Max-Delay optimally in polynomial time unless \( P = NP \). Later in Sec. IV, a pseudo-polynomial time algorithm is proposed to solve Min-Max-Delay optimally, which further proves that Min-Max-Delay is NP-complete in the weak sense.

**B. Strong NP-completeness for Int-Min-Max-Delay**

We now analyze the computational complexity of Int-Min-Max-Delay where only integer flows are allowed. We begin with the definition of the 3-Partition Problem [18].

![Fig. 2. Reduced network graph from 3-partition problem.](image)

**Definition 3 (3-Partition Problem [18]).** Given a set of \( n = 3k \) \((k > 0)\) positive integers \( A = \{a_1, a_2, ..., a_n\} \) with sum \( \sum_{a_i \in A} a_i = kb \) and \( b/4 < a_i < b/2 \) for each element \( a_i \), is there a partition \( \{A_1, A_2, ..., A_k\} \) of set \( A \) such that for each subset \( A_j \), \( \sum_{a_i \in A_j} a_i = b \).

The 3-3-partition problem has been proved to be NP-complete in the strong sense [18], which can be utilized to prove the strong NP-completeness of our problem Int-Min-Max-Delay.

**Theorem 2.** The decision version of Int-Min-Max-Delay problem is NP-complete in the strong sense.

**Proof:** For any 3-partition problem, we construct a network following Fig. 2. Edge capacity and delay are similarly defined as in the proof to Thm. 1. We set the flow requirement \( R = k \) and thus get the problem Int-Min-Max-Delay. Such a reduction is a pseudo-polynomial transformation [18]. Similar to the proof in Thm. 1, we show that the 3-partition problem answers “Yes” if and only if the decision problem of Int-Min-Max-Delay answers “Yes”. Due to the space limit, we put the detailed proof in our technical report [19].

As we will shown later in Thm. 4, Min-Max-Delay is weakly NP-complete. Thus, Thm. 2 shows that Int-Min-Max-Delay is more difficult than Min-Max-Delay. Later in Sec. V we show that the integrality gap defined in (4) could be arbitrarily large, which further illustrates additional intriguing difficulties for the Int-Min-Max-Delay problem.

**IV. OPTIMAL PSEUDO-POLYNOMIAL TIME ALGORITHM**

In this section we propose a pseudo-polynomial time algorithm to solve Min-Max-Delay optimally. This, combined with the NP-completeness result in Thm. 1, shows that Min-Max-Delay is NP-complete in the weak sense.

A closely-related problem to Min-Max-Delay is the Delay-Constrained Maximum Flow problem [14], denoted as DC-Max-Flow: for the same graph \( G \) and a given deadline constraint \( T \), finding the maximum flow (rate) such that the delay of any flow-carrying path does not exceed \( T \). Let us denote \( PT \) as the set of all \( s-t \) paths whose path delay does not exceed \( T \). Then DC-Max-Flow can be formulated as

\[
\begin{align*}
\max & \quad \sum_{p \in PT} f_p, \tag{9a} \\
\text{s.t.} & \quad f_e = \sum_{p \in PT: e \in p} f_p \leq c_e, \ \forall e \in E \tag{9b} \\
\text{vars.} & \quad f_p \geq 0, \ \forall p \in PT. \tag{9c}
\end{align*}
\]
It is shown that DC-Max-Flow can be solved with an edge-based flow formulation in pseudo-polynomial time [14]. They implicitly use the idea of time-expanded graph [20] by converting a delay-constrained max-flow problem in the original graph into a delay-unconstrained max-flow problem in the expanded graph. In [14], the authors only consider unit-delay edges, but it is easy to generalize their result to integer-delay edges. Due to the space limit, we omit the procedure of constructing the expanded graph and directly give the equivalent edge-based flow formulation for DC-Max-Flow as follows [14, Proposition 1],

\[
\begin{align*}
\max & \sum_{e \in \ln(t)} \sum_{d=0}^{T} f_e(d) \\
\text{s.t.} & \sum_{e \in \Out(s)} f_e(d_e) = \sum_{e \in \ln(t)} \sum_{d=0}^{T} f_e(d), \\
& \sum_{e \in \ln(v)} f_e(d) = \sum_{e \in \Out(v)} f_e(d_e), \\
& \forall v \in V \setminus \{s, t\}, d \in [0, T] \\
& \sum_{d=0}^{T} f_e(d_e) \leq c_e, \forall e \in E \\
\text{vars.} & f_e(d_e) \geq 0, \forall e \in E, d \in [0, T]
\end{align*}
\]

where \( \ln(v) \triangleq \{ e = (w, v) : e \in E, w \in V \} \) is the set of incoming edges of node \( v \), \( \Out(v) \triangleq \{ e = (v, w) : e \in E, w \in V \} \) is the set of outgoing edges of node \( v \), and \( f_e(d) \) is the total flow rate that experiences a delay of \( d \) after passing edge \( e \) from the source \( s \). The objective (10a) is the total flow rate that arrives at the sink \( t \) within the delay bound \( T \). Eqs. (10b) requires the rate entering the network should equal to the rate leaving the network. Eqs. (10c) are the flow conservation constraints in the expanded graph. Note that by convention, for any edge \( e \in E \), we set \( f_e(0) = 0 \) for \( d < 0 \) and \( d > T \). Eqs. (10d) are the edge capacity constraints.

We now show the relationship between our problem Min-Max-Delay and problem DC-Max-Flow. For a graph \( G \), denote \( d^*(R) \) as the minimum maximum delay with rate requirement \( R \) (the optimal value of Min-Max-Delay). For the same graph \( G \) and a non-negative integer \( T \), denote \( r^*(T) \) as the maximum flow subject to a delay constraint \( T \) (the optimal value of DC-Max-Flow). We have the following lemma.

**Lemma 1.** \( d^*(R) \leq T \) if and only if \( r^*(T) \geq R \).

**Proof:** If Part. If \( r^*(T) \geq R \), then there exists a flow solution over \( P^T \) such that \( \sum_{e \in P^T} f^P \geq R \). We can thus decrease the flow solution \( f \) to construct another flow solution \( \tilde{f} \) such that \( \sum_{e \in P^T} \tilde{f} = R \). Since \( f \) satisfies the capacity constraints, \( \tilde{f} \) must also satisfy the capacity constraints. Thus, \( \tilde{f} \) is a feasible solution to Min-Max-Delay with rate requirement \( R \). In addition, since all flow-carrying paths in \( \tilde{f} \) belong to the set \( P^T \), we have \( d^*(R) \leq D(\tilde{f}) \leq T \).

Only If Part. If \( d^*(R) \leq T \), then there exists a flow solution \( f \) where the path delay of any flow-carrying path does not exceed \( T \). Thus all flow-carrying paths belong to \( P^T \) and \( f \) is also a feasible solution to DC-Max-Flow with a delay bound \( T \). Thus, \( r^*(T) \geq \sum_{e \in P^T} \tilde{f} = R \).

**Theorem 3.** The binary search scheme solves Min-Max-Delay problem optimally and has a pseudo-polynomial time complexity \( O(\log(N^{d_{max}})N^2d_{max}^2 \log(\log(N^{d_{max}}))) \).

**Proof:** The optimality of the binary search scheme directly follows from Lem. 1. For the time complexity analysis, please see our technical report in [19].

The time complexity shown above is pseudo-polynomial in the sense that it is polynomial in the numeric value of the input \( d_{max} \), but it is exponential in the bit length of the \( d_{max} \). Thus, our binary search algorithm could still have high complexity for the graph with large \( d_{max} \). To further reduce the complexity, we adapt the rounding and scaling approach [21] and design a fully polynomial time approximation scheme (FPTAS) for our problem Min-Max-Delay. For any \( \epsilon > 0 \), our proposed FPTAS can find a \((1 + \epsilon)\)-approximate solution and the time complexity is polynomial in both problem size and \( 1/\epsilon \). Due to the space limit, the detailed FPTAS design is shown in our technical report [19].

A direct result of Thm. 3 is as follows.

**Theorem 4.** The decision version of Min-Max-Delay is NP-complete in the weak sense.

**Proof:** It follows from Thm. 1 and Thm. 3.

V. INTEGRALITY GAP

We have proved that Min-Max-Delay allowing fractional flow is NP-complete in the weak sense, but Int-Min-Max-Delay only allowing integer flow solution is NP-complete in the strong sense. Thus, there are some fundamental differences between these two problems. To elaborate their differences, in this section, we construct a network to show that the integrality gap defined in (4) could be arbitrarily large.

We will use the network in Fig. 3 as a building block to construct the network with arbitrarily large integrality gap. Towards that end, for the building-block graph in Fig. 3 with source node \( a_1 \) and sink node \( a_n \), we first analyze the maximum delay of problem Int-Min-Max-Delay and problem Min-Max-Delay in Lem. 2 and Lem. 3, respectively.

**Lemma 2.** The Int-Min-Max-Delay flow in Fig. 3 gives \( R = 2 \) has a maximum delay of \( \lceil \frac{n-1}{2} \rceil \).

![Fig. 3. A building-block graph with \( n \geq 3 \) to show that the integrality gap could be arbitrarily large. All edges have unit capacity. Each upper dashed edge has a unit delay while each lower solid edge has a delay of zero.](image-url)
Proof: Because of the unit capacity constraint for each edge, any flow-carrying path will be assigned a unit flow rate in the integer flow from $a_1$ to $a_n$.

Because of the rate requirement $R = 2$, there are exactly two flow-carrying paths in the Int-Min-Max-Delay flow and each of them carries a unit flow rate. Moreover, because of the capacity constraint, the two flow-carrying paths must be disjoint, namely they share no edges.

Therefore, since the Int-Min-Max-Delay flow minimizes the maximum flow-carrying path delay among all integer flows, the two flow-carrying paths will have path delay of $\lceil (n-1)/2 \rceil$ and $\lceil (n-1)/2 \rceil$ respectively in the Int-Min-Max-Delay flow, leading to a maximum delay of $\lceil (n-1)/2 \rceil$.

Lemma 3. The Min-Max-Delay flow in Fig. 3 given $R = \frac{n-1}{n-2}$ has a maximum delay of 1.

Proof: First it is straightforward that the Min-Max-Delay flow has a maximum delay no smaller than 1 since $R > 1$.

Next, we will explicitly construct a feasible flow with unit maximum delay: Note that there are $(n - 1)$ different $a_1 - a_n$ paths containing exactly one dashed edge. We then place $1/(n - 2)$ flow rate on each of these $(n - 1)$ paths with the rate requirement $R = (n - 1)/(n - 2)$ satisfied. It is easy to verify that edge capacity constraints are satisfied, too.

Note that even though our system model in Sec. II requires an integer rate $R$, problem Min-Max-Delay is also well defined for fractional rate requirement $R$, which is the case in Lem. 3. However, later in our constructed graph to show that the integrality gap can be arbitrarily large (see the proof in Thm. 5), we set integer flow rate requirement $R$, which is in line with our system model in Sec. II.

Based on Lem. 2 and Lem. 3, we use Fig. 3 as a building-block to construct a network with integer edge delay, integer capacity, integer rate $R$ and arbitrarily large Int-Gap.

Theorem 5. There exists a Min-Max-Delay problem instance such that Int-Gap could be arbitrarily large.

Proof: We place $(n-2)$ subnetworks with the topology of Fig. 3 side by side disjointly and then connect each of them to both a single source $s$ and a single sink $t$. All the outgoing edges of $s$ and the incoming edges of $t$ have a capacity of 2 and a delay of 0. We set the rate requirement $R = n - 1$.

First, considering Lem. 3, the Min-Max-Delay flow has a maximum delay of 1 because we can route $(n - 1)/(n - 2)$ flow rate to each of the $(n - 2)$ subnetwork.

Next since the rate requirement $R = n - 1$ is strictly larger than $n - 2$ which is the number of subnetworks, for any feasible integer flow, there will be at least one subnetwork who is assigned a flow rate of 2. According to Lem. 2, corresponding minimal maximum delay to pass the subnetwork is $\lceil (n-1)/2 \rceil$. Thus, the Int-Min-Max-Delay flow will have a maximum delay of $\lceil (n-1)/2 \rceil$.

Overall, the Int-Gap is $\lceil (n-1)/2 \rceil$. Since $n$ can be arbitrarily large, the gap can be arbitrarily large.

Thm. 5 shows that indeed Int-Min-Max-Delay are different from Min-Max-Delay. Though we have designed an optimal pseudo-polynomial time algorithm and an FPTAS for problem Min-Max-Delay, we need to tackle some additional intriguing difficulties to design efficient approximate algorithms for the strongly NP-complete problem Int-Min-Max-Delay. In fact, our report [19] further introduces a near-tight bi-criteria $(1 - \epsilon, 1/\epsilon)$ upper bound for Int-Gap.

VI. CONCLUSION

In the paper we study the Min-Max-Delay single-unicast flow problem which minimizes the maximum end-to-end delay under a rate requirement. We prove it is NP-complete in the weak sense, and propose a pseudo-polynomial time algorithm to find the optimal solution. We also show that if integer flow solution is required, the problem becomes NP-complete in the strong sense. We further construct a network to show that the integrality gap, which is the maximum delay ratio of the optimal integer flow to the optimal fractional flow, could be arbitrarily large.

REFERENCES


