Bulletin of the *Transilvania* University of Braşov • Vol 13(62), No. 1 - 2020
Series III: Mathematics, Informatics, Physics, 363-372
https://doi.org/10.31926/but.mif.2020.13.62.1.27

MAXIMUM FLOW IN BUFFER-LIMITED DELAY TOLERANT NETWORKS. THE STATIC APPROACH

Corina-Ştefania NĂNĂU[1]

Abstract

Delay Tolerant Networks (DTNs), such as Internet, ad hoc networks, satellite networks and sensor networks, have attracted considerable attention. The maximum flow problem has a vital importance for routing and service scheduling in networks. For delay tolerant networks there are no permanent end-to-end paths since the topology and links characteristics are time-varying. In these instances, to account properly for the evolution of the underlying system over time, we need to use dynamic network flow models. When time is considered as a discrete variable, these problems can be solved by constructing an equivalent static time expanded network. This is a static approach. In this paper we study the maximum flow in a buffer-limited delay tolerant network, with static approach.

2000 *Mathematics Subject Classification:* 93C35, 90B10, 68R10.
Key words: delay tolerant network, time varying graph, maximum flow.

1 Introduction

The graph theory has been widely utilized to study DTNs in many works. The maximum flow problem poses a crucial issue in network flow theory, which is also the basis for routing and service scheduling in networks. When time is considered as a variable discrete value, this problem can be solved by constructing an equivalent static time expanded network [1, 4].

Static time expanded network has more nodes and arcs than the dynamic network. Time Aggregated Graph (TAG) allows the properties of arcs to be modeled as time series [6, 9].

Since the model does not need to replicate the entire graph for each time interval, the algorithms for common operation are computationally more efficient than those for static time expanded network. In this paper we describe the solution of maximum flow in buffer-limited delay tolerant network, with static approach. In many cases this approach is preferable.

[1]Faculty of Mathematics and Informatics, *Transilvania* University of Braşov, Romania, e-mail: corina.nanau@unitbv.ro

The remainder of this paper is organized as follows. In Section 2, the maximum flow in static networks and dynamic networks are presented. Following it, Section 3 provides the maximum flow in buffer-limited delay tolerant network. In Section 4 there is an example.

2 Maximum flow in static networks and dynamic networks

In this section some notations and results used throughout the paper are discussed.

It is considered a connected, antisymmetric graph G = (N, A) without loops, with the set of nodes $N = \{1, ..., i, ..., j, ..., n\}$, the set of arcs $A = \{a_1, ..., a_k, ..., a_m\}, a_k = (i, j), i, j \in N$. Let S = (N, A, u) be a static network with the upper bound (capacity) function $u : A \to \mathbb{N}$, $\mathbb{N}$ the natural number set, 1 the source node and n the sink node.

For a given pair of subset X, Y of the nodes set N of a network S, the notation is:

$$(X, Y) = \{(i, j)|(i, j) \in A, i \in X, j \in Y\}$$

and for a given function f on arcs set A, the notation is:

$$f(x, y) = \sum_{(x,y)} f(i, j)$$

If $X = \{i\}$ or $Y = \{j\}$, the notation (i, Y) respectively (X, j) can be used.

A flow is a function $f : A \to \mathbb{N}$ satisfying the next conditions:

$$f(i, N) - f(N, i) = \begin{cases} v & \text{if } i = 1 \\ 0 & \text{if } i \neq 1, n \\ -v & \text{if } i = n \end{cases} \tag{1a}$$

$$0 \leq f(i, j) \leq u(i, j), (i, j) \in A \tag{1b}$$

for some $v \geq 0$. We refer to v as the value of the flow f.

The maximum flow problem is to determine a flow f for which v is maximum.

Many algorithms for the maximum flow problem are based on the concept of flow augmenting paths. A flow f is considered and the residual network $R = (N, \tilde{A}, r)$ is defined with nodes set N and

$$\tilde{A} = \tilde{A}^+ \cup \tilde{A}^- \tag{2a}$$

$$\tilde{A}^+ = \{(i, j)|(i, j) \in A \text{ and } f(i, j) \leq u(i, j)\} \tag{2b}$$

$$\tilde{A}^- = \{(i, j)|(i, j) \in A \text{ and } f(i, j) \geq 0\} \tag{2c}$$

The residual capacity $r : \tilde{A} \to \mathbb{N}$ with respect to f is

$$r(i, j) = \begin{cases} u(i, j) - f(i, j) & \text{for } (i, j) \in \tilde{A}^+ \\ f(j, i) & \text{for } (j, i) \in \tilde{A}^- \end{cases} \tag{3}$$

A flow augmenting path related to f is a path P in R from 1 to n.

Theorem 1. *[1] A flow f^* is a maximum flow if and only if the residual network R contains no augmenting path.*

Some additional notions and results are introduced. A distance function in the residual network R is a function $d : N \to \mathbb{N}$. We say that a distance function is valid if it satisfies the following two conditions:

$$d(n) = 0 \tag{4a}$$

$$d(i) \leq d(j) + 1, i \in N, (i, j) \in A \tag{4b}$$

The distance labels have the following two properties [1].

Property 1. If the distance labels are valid, the distance label $d(i)$ is a lower bound on the length of the shortest path from node i to node n in the residual network R.

Property 2. If $d(1) \geq n$, the residual network R contains no path from the source node 1 to the sink node n.

We say that the distance labels are exact for each node i if $d(i)$ equals the length of the shortest path from node i to node n in the residual network R. Also, we say that an arc (i,j) in the residual network is admissible if it satisfies the condition that $d(i) = d(j) + 1$ and we refer to all other arcs as inadmissible. A path from node 1 to node n consisting entirely of admissible arcs is an admissible path. An admissible path has the following property [1].

Property 3. An admissible path from the source node 1 to the sink node n is a shortest augmenting path.

An shortest augmenting path from the source node 1 to the sink node n in residual network R can be determined by performing a backward breath-first search of the network R, starting at the sink node n with $d(n) = 0$.

The shortest augmenting path algorithm is presented in Algorithm 1, which appeals three procedures presented in Algorithm 2, Algorithm 3 and Algorithm 4 [1].

Algorithm 1 Shortest Augmenting Path Algorithm.

```
1: f ← 0
2: obtain the exact distance labels d(i)
3: i ← 1
4: while d(1) < n do
5:     if i has an admissible arc then
6:         Advance(i)
7:         if i = n then
8:             Augment and set i = 1
9:         end if
10:    else
11:        Retreat(i)
12:    end if
13: end while
```

Algorithm 2 Advance Procedure.

```
1: procedure ADVANCE(i)
2:     let (i, j) be an admissible arc in Ã
3:     pred(j) ← i
4:     i ← j
5: end procedure
```

Algorithm 3 Retreat Procedure.

```
1: procedure RETREAT(i)
2:     d(i) ← min{d(j) + 1|(i, j) ∈ Ã}
3:     if i ≠ s then
4:         i ← pred(i)
5:     end if
6: end procedure
```

Algorithm 4 Augment Procedure.

```
1: procedure AUGMENT
2:     using the predecessor indices identify an augmenting path P from 1 to n
3:     mr ← min{r(i, j)|(i, j) ∈ P}
4:     augment mr units of flow along path P
5: end procedure
```

There are the following two theorems [1].

Theorem 2. *The shortest augmenting path algorithm correctly computes a maximum flow in network* $S = (N, A, u)$.

Theorem 3. *The shortest augmenting path algorithm runs in* $O(n^2m)$ *time.*

Further, there are some notations and notions for dynamic flow.

The static network models have many applications. In other applications, time is an essential ingredient. In these instances, we need to use dynamic network flow models.

Obviously, the problem of finding a maximum flow in dynamic network $D = (N, A, u, h)$, with time function $h : A \to \mathbb{N}$, is more complex than the problem of finding a maximum flow in static network $S = (N, A, u)$. Fortunately, this issue can be solved by rephrasing the problem in dynamic network D into a problem in the expanded static network $S_e = (N_e, A_e, u_e)$. This is a static approach.

Let $H = \{0, 1, ..., T\}$ be the set of time periods. There is $N_e^{'} = \{i_t | i \in N, t \in H\}, A_e^{'} = \{(i_t, j_\theta) | (i, j) \in A; t, \theta \in H\}, u_e^{'}(i_t, j_\theta) = u(i, j), (i_t, j_\theta) \in A_e^{'}$. Thus the expanded static network $S_e^{'} = (N_e^{'}, A_e^{'}, u_e^{'})$ is obtained with multiple sources $1_1, ..., 1_T$ and multiple sinks $n_1, ...n_T$. It can further be reduced the multiple sources, multiple sink problems in the time expanded network $S_e^{'} = (N_e^{'}, A_e^{'}, u_e^{'})$ to the single source, single sink problem by introducing a super source node 0 and a super sink node $n + 1$. Thus the time expanded network $S_e = (N_e, A_e, u_e)$, is obtained with $N_e = N_e^{'} \cup \{0, n + 1\}, A_e = A_e^{'} \cup \{(0, 1_t) | t \in H\} \cup \{(n_t, n + 1) | t \in H\}, u_e(0, 1_t) = u_e(n_t, n + 1) = \infty, t \in H$.

For more details see [1, 2, 4, 8]. Another dynamic networks were studied and detailed in [3, 7].

3 Maximum flow in buffer-limited delay tolerant networks

The DTN network is a special case of dynamic network [6], [9]. Assuming a time period $H = [t_0, T)$, where t_0 and T represent the start time and respectively the terminal time. H is partitioned into q small time intervals $\tau_k = [t_{k-1}, t_k), k = 1, 2, ..., q$.

Let $D = (N, A, H, u_\tau, c_\tau, b_\tau)$ be the dynamic network with the set of nodes $N = \{1, ..., n\}$, the set of arcs $A = \{a_1, ..., a_m\}$, the time period $H = [t_0, T)$, the upper bound (capacity) $u_\tau(i, j) = (u_{\tau_1}(i, j), ..., u_{\tau_q}(i, j)), (i, j) \in A$, the cache transfer series $c_\tau = (c_{\tau_1,\tau_2}(i), ..., c_{\tau_{q-1},\tau_q}(i)), i \in N$, where $c_{\tau_{k-1},\tau_k}(i)$ describes the data transferred from τ_{k-1} to τ_k time interval in node i. The initial values of $c_\tau(i)$ for all $i \in N$ are initialized to zero and the node buffers $b_\tau(i) = (b_{\tau_1,\tau_2}(i), ..., b_{\tau_{q-1},\tau_q}(i)), i \in N - \{1, n\}, b_{\tau_{k-1},\tau_k}(1) = b_{\tau_{k-1},\tau_k}(n) = \infty, k = 1, ..., q$.

The maximum flow problem of buffer-limited DTN is to send as much flow as possible from source node 1 to sink node n, without violating the arcs capacity and the nodes constraints. In this paper the static approach is used.

For maximum flow in buffer-limited DTN problem the time extended network $S_e = (N_e, A_e, u_e)$ is used here with the following modifications and completions: $N_e^{'} = \{i_k | i \in N, k = 1, ..., q\}, A_e^{'} = \{(i_k, j_k) | (i, j) \in A, k = 1, ..., q\} \cup \{(i_k, i_{k+1}) | k = 1, ..., q - 1\}, u_e(i_k, j_k) = u_{\tau_k}(i, j), k = 1, ..., q, u_e(i_k, i_{k+1}) = b_{\tau_k,\tau_{k+1}}(i, j), k = 1, ..., q - 1$.

In this network the shortest augmenting path algorithm is used, which has been presented in Section 2, to obtain the maximum flow in static extended network $S_e = (N_e, A_e, u_e)$, which is equivalent with a maximal flow in dynamic network $D = (N, A, H, u_\tau, c_\tau, b_\tau)$. The residual network $R_e = (N_e, \tilde{A}_e, r_e)$ is built as the residual network $R = (N, \tilde{A}, r)$ which has been described in Section 2.

The following two theorems are presented.

Theorem 4. *The shortest augmenting path algorithm correctly computes a maximum flow in static extended network $S_e = (N_e, A_e, u_e)$.*

Proof. This theorem results from Theorem 2. □

Theorem 5. *The shortest augmenting path algorithm applied in static extended network $S_e = (N_e, A_e, u_e)$ runs in $O(T^3n^2(n+m))$ time.*

Proof. From Theorem 3 results that the shortest augmenting path algorithm applies in static extended network $S_e = (N_e, A_e, u_e)$ runs in $O(n_e^2 m_e)$ time. There is $n_e = Tn + 2$ and $O(Tn + 2) = O(Tn)$, $m_e = (T-1)(n-2) + Tm + 2T = T(n+m) - n + 2 < T(n+m)$. Thus, the algorithm runs in $O(T^3n^2(n+m))$ time. □

4 Example

In the example from paper [9] a maximum flow with dynamic approach is obtained. In this paper, that example is taken again in order to obtain a maximum flow with static approach.

The support graph $G = (N, A)$ is presented in Figure 1, where the source node is 1, the sink node is 4 and the given time period is $H = [t_0, T) = [0, 5)$. Note here that T is partitioned into 5 time intervals, for instance $\tau_1 = [0, 1), \tau_2 = [1, 2), \tau_3 = [2, 3), \tau_4 = [3, 4), \tau_5 = [4, 5)$. Assume that the initial feasible flow is equal to zero.

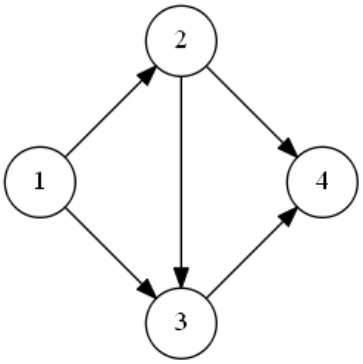

Figure 1: The support graph G=(N,A)

The capacities $u_{\tau_k}(i, j), k = 1, 2, 3, 4, 5$ and $(i, j) \in A$ are indicated in Table 1.

$(i,j)/u_\tau(i,j)$	$u_{\tau_1}(i,j)$	$u_{\tau_2}(i,j)$	$u_{\tau_3}(i,j)$	$u_{\tau_4}(i,j)$	$u_{\tau_5}(i,j)$
(1,2)	7	0	6	0	0
(1,3)	0	3	2	0	0
(2,3)	6	0	1	0	0
(2,4)	0	7	0	4	2
(3,4)	1	0	0	5	1

Table 1: The capacities $u_{\tau_k}(i,j), k = 1,2,3,4,5$ and $(i,j) \in A$

The nodes buffers $b_\tau(i) = (b_{\tau_1,\tau_2}, b_{\tau_2,\tau_3}, b_{\tau_3,\tau_4}, b_{\tau_4,\tau_5}), i \in N$ are indicated in Table 2.

$i/b_\tau(i)$	$b_{\tau_1,\tau_2}(i)$	$b_{\tau_2,\tau_3}(i)$	$b_{\tau_3,\tau_4}(i)$	$b_{\tau_4,\tau_5}(i)$
1	∞	∞	∞	∞
2	5	5	5	5
3	5	5	5	5
4	∞	∞	∞	∞

Table 2: The node buffers $b_\tau(i), i \in N$

The time extended network $S_e = (N_e, A_e, u_e)$ is presented in Figure 2 and on each arc the capacity is passed. In this network the shortest augmenting path algorithm is used to obtain the path $P_1 = (0, 1_1, 2_1, 3_1, 4_1, 5)$ is obtained with path flow $f(P_1) = 1$. After flow enlarging the residual network $R_e = (N_e, \tilde{A}_e, r_e)$ which is presented in Figure 3. Further the augmenting paths is determined:

- $P_2 = (0, 1_1, 2_1, 2_2, 4_2, 5)$ with path flow $f(P_2) = 5$
- $P_3 = (0, 1_3, 2_3, 2_4, 4_4, 5)$ with path flow $f(P_3) = 4$
- $P_4 = (0, 1_3, 3_3, 3_4, 4_4, 5)$ with path flow $f(P_4) = 2$
- $P_5 = (0, 1_2, 3_2, 3_3, 3_4, 4_4, 5)$ with path flow $f(P_5) = 1$
- $P_6 = (0, 1_3, 2_3, 2_4, 2_5, 4_5, 5)$ with path flow $f(P_6) = 1$
- $P_7 = (0, 1_3, 2_3, 3_3, 3_4, 4_4, 5)$ with path flow $f(P_7) = 1$
- $P_8 = (0, 1_1, 2_1, 3_1, 3_2, 3_3, 3_4, 4_5, 5)$ with path flow $f(P_8) = 1$

In residual network $R_e = (N_e, \tilde{A}_e, r_e)$ there is not an augmenting path and from Theorem 1 results that the flow is a maximum flow. The value of maximum flow is $v_e = f(P_1) + f(P_2) + f(P_3) + f(P_4) + f(P_5) + f(P_6) + f(P_7) + f(P_8) = 1 + 5 + 4 + 2 + 1 + 1 + 1 + 1 = 16$.

The time extended network $S_e = (N_e, A_e, u_e)$, is presented in Figure 4, where on each arc the capacity and the flow are passed.

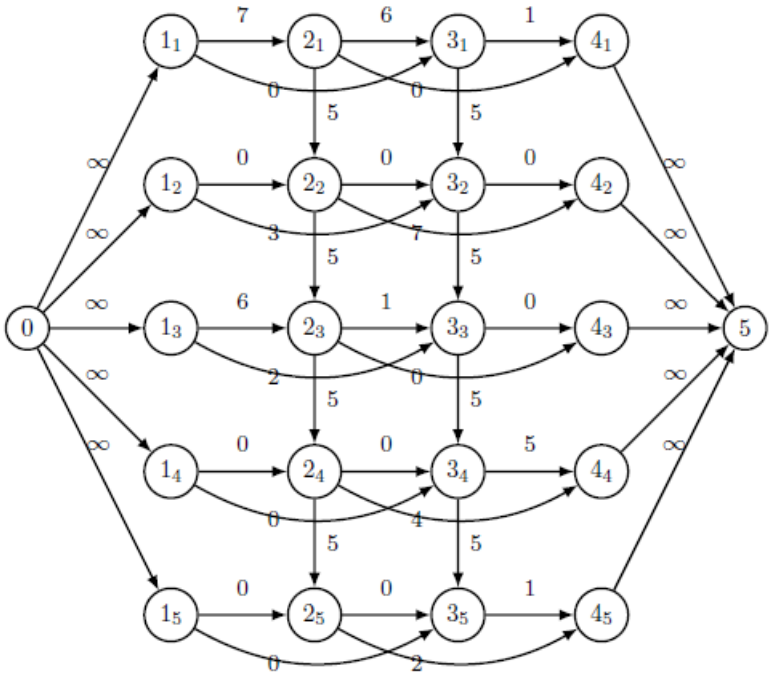

Figure 2: The time extended network $S_e = (N_e, A_e, u_e)$

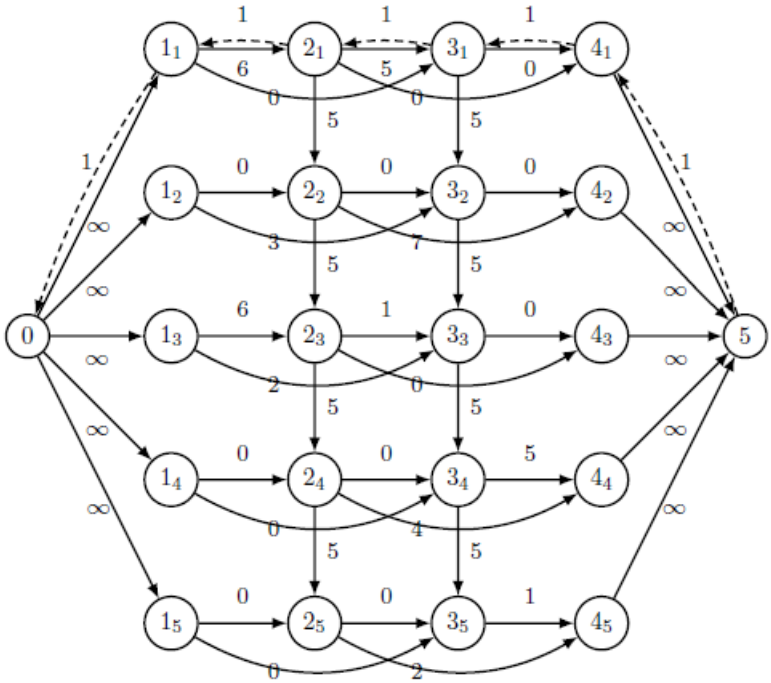

Figure 3: The residual network $R_e = (N_e, \tilde{A}_e, r_e)$

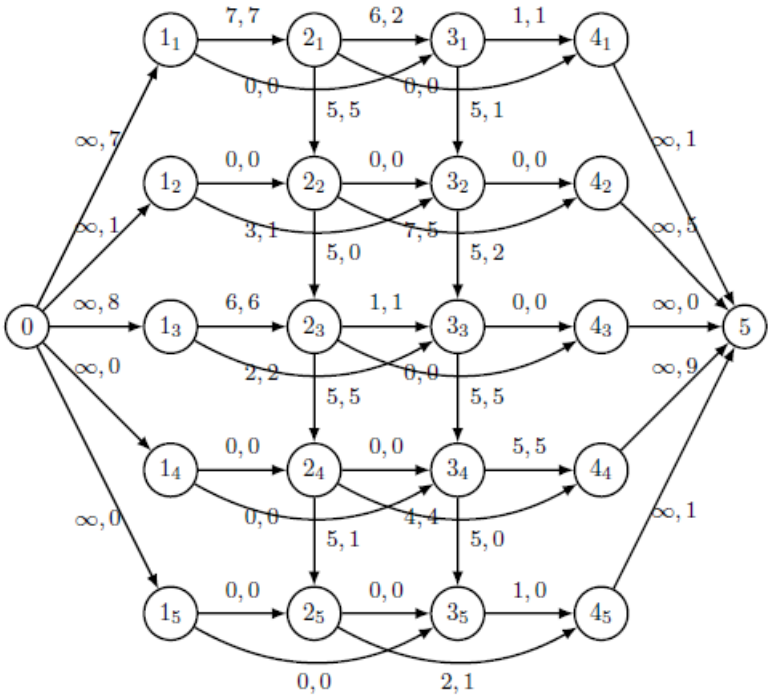

Figure 4: The network $S_e = (N_e, A_e, u_e)$ and the maximum flow

5 Conclusions

In this paper, we have presented a solution of maximum flow in buffer-limited delay tolerant networks with static approach. In many cases this approach is preferable. We have succeed in adapting the time expanded network $S_e = (N_e, A_e, u_e)$ for the classic dynamic network $D = (N, A, H, u)$ to the time expanded network $S_e = (N_e, A_e, u_e)$ for the buffer-limited delay tolerant network $D = (N, A, H, u_\tau, c_\tau, b_\tau)$.

We have studied only the maximum flow problem from single source to single sink node in a network. The multiple sources and multiple sinks flow problems have many applications in practice, and will be further studied.

References

[1] Ahuja, R., Magnanti, T. and Orlin, J., *Network flows. Theory, algorithms and applications*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1993.

[2] Cai, X., Sha, D. and Wong, C., *Time-varying Network Optimization*, Springer, 2007.

[3] Ciurea, E., *Counterexamples in maximal dynamic flows*, Annual Congress of

the American Romanian Academy of Arts and Sciences, Libertas Mathematica **17** (1997), 77-97.

[4] Ford, L.R. and Fulkerson, D.R., *Flows in Networks*, Princeton University Press, Princeton New Jersey, 1962.

[5] Jungnickel, D., *Graphs, networks and algorithms*, Springer, Berlin, 1999.

[6] Li, H., Zhang, T., Zhang, Y., Wang, K. and Li, J., *A maximum flow algorithm based on storage time aggregated graph for Delay Tolerant Networks*, Ad Hoc Networks **59** (2017), 63-70.

[7] Schiopu, C. and Ciurea, E., *Maximum flows in planar dynamic networks with lower bounds*, Fundamenta Informaticae, **163** (2018), no. 2, 189204.

[8] Skutella, M.G., *An Introduction to network flows over time. Research trends in combinatorial optimization*, Springer, (2008), pp. 451-482.

[9] Zhang, T., Deng, S., Li, H., Hou, R. and Zhang, H., *A maximum flow algorithm for buffer-limited Delay Tolerant Networks*, Journal of Communications and Information Networks, **2** (2017), no. 3, 52-60.