Exploring the Distributed Max Flow Algorithm for IoT Networks

Main Uddin, Md. Hasnat Riaz, Nishu Nath, Md. Auhidur Rahman

Abstract— For any kind of IoT network, we want maximum flow from source to destination. It is a classical optimization problem. In case of the static network, we can easily implement the classical max-flow algorithm. However, in the distributed environment, we can't implement classical max-flow algorithm directly. The main challenges are any node can join or leave at any time and node mobility. To get the max-flow in the distributed network with minimum overhead, we investigate the flooding method considering the three different types of distributed network environment based on the well-known classical max-flow algorithm such as Dinic's, Ford-Fulkerson and Edmond-Karp algorithm. However, our implementation is not restricted to any kind of particular max-flow algorithm.

Index Terms—IoT Network, Max-Flow, Node, Routing Protocol, Overhead, Dinic's Algorithm, Flooding

1. Introduction

The Internet of Things (IoT) is the most disruptive innovation in today's world that focuses at interconnecting every physical identifiable object (or, a thing) via a global networking infrastructure. The number of IoT device is increasing excessively. These IoT devices are highly distributed and have significantly less storage and processing capacity. In distributed IoT network, we want to send data from source to destination with maximum flow speed. The max-flow problem is defined as follows: for a directed graph *G* with edge capacities and given a source node s and a target node t, what is the maximum flow that can be pushed through from s to t through the edges of *G*?

We consider a directed graph with *n* nodes, *m* edges with positive capacity *c* and two distinguished nodes named source and sink. We can assume each edges in the network as a pipe which conduits water and the capacity of each edge is capacity-carrying of that pipe. Then the water flows in the network such that the amount of water in each pipe does not exceed the capacity of that edge. The maximum flow problem then responses to the question: how to maximize the quantity of water that could flow from the source to the sink or target [1].

Basically, to solve the maximum problem, there are two principal categories: the labelling Fork-Fulkerson's method [2] and the preflow-push method that was introduced by Golberg-Tarjan [3] who takes the original idea of preflow from Karzanov [4]. The labelling method increases flow along augmenting paths from the source node to the sink node. The idea of preflow-push algorithm is to seek-out the shortest paths as in the labelling method, but do not send flow along paths from the source to the sink. Instead, it processes excesses at nodes: nodes send flows on individuals arcs based on the knowledge it has about itself and its neighbors. This algorithm of Goldberg-Tarjan makes decisions locally and hence, is suitable for a distributed version in the asynchronous network.

In IoT network, each IoT device has certain properties such as processing, transmitting and receiving power and bandwidth. Another property is device mobility. To transmit data from source to destination in the network, we need at least a single path from source to destination. To get maximum flow from source to destination, we have to find all possible path from source to destination. In case of static, all node position in the network is fixed. Secondly, we can collect all node connectivity information between the nodes in the network. After that applying any classical max-flow algorithm like dinic's algorithm, we can get max-flow from source to destination easily.

However, we can't apply the classical max-flow algorithm to get max-flow in the dynamic IoT network. The first challenge is any node can join or leave at any time in the network. Secondly, the node has mobility property. Thirdly, each node has individual property. These are main obstacles in the dynamic or distributed environment to get max-flow by implementing the classical max-flow algorithm.

In this paper, we investigate how to implement classical max-flow algorithm such as dinic's or others algorithm to maximize data flow in distributed IoT network.

2. Materials and Methods

The distributed IoT Networks are comprised of mobile nodes that are communicating via either directed wireless links or multi-hop wireless links through a sequence of intermediate nodes. The multipath routing in IoT networks is difficult because the network topology may change constantly, and the

available alternative path is inherently unreliable. So far, much of the effort of multipath routing has been focused on using the predefined alternate path when a relay on the primary path has failed regardless of the availability of the alternate path. This reactive route handoff can increase the overhead for frequent route discoveries. A novel Entropy-based Long-life Multipath Routing algorithm in MANET (ELMR) can be used to resolve above problem [5].

A Network Coding-based on-demand Multipath Routing algorithm in MANET (NCMR) is typically proposed in order to increase the reliability of data transmission or to provide load balancing. The NCMR routing protocol provide an accurate and efficient method of estimating and evaluating the route stability in dynamic MANETs[6].

As the nodes in a wireless IoT network can be connected in a dynamic and arbitrary manner, the nodes themselves must behave as routers and take part in discovery and maintenance of routes to other nodes in the network. The goal of a routing algorithm is to devise a scheme for transferring a packet from one node to another. One challenge is to define/choose which criteria to base the routing decisions on. There are some well-known routing protocol those are basically used in Mobile Adhoc Network (Manet). DSDV (destination-sequenced distance vector) protocol [7] is proactive Table-Driven Routing Protocol that maintain the global topology information in the form of tables at every node. AODV (ad hoc on-demand distance vector) [8], [9] is a distance vector routing protocol that discovers routes only on demand and doesn't maintain routes from every node to every other.

In centralized approach, e.g. static network, the neighbor nodes are almost fixed for each node with certain transmission range. There are some well-known classical max-flow algorithm those are basically used in static network to calculate max-flow.

(a) Dinic's Algorithm:
This algorithm is a strongly polynomial algorithm for computing the maximum flow in a network[10], conceived in 1970 by Israeli (formerly Soviet) computer scientist Yefim Dinitz. The algorithm runs in time and is similar to the Edmonds–Karp algorithm, which runs in time, in that it uses shortest augmenting paths.

(b) Ford-Fulkerson Algorithm:
Ester R. Ford, Jr. and Delbert R. Fulkerson created the first known algorithm for computing max flow from source to sink in a network, called the FFA (Ford-Fulkerson algorithm). It is a "method" rather than an "algorithm" because it encompasses several implementations with differing running times. The Ford-Fulkerson method depends on three important ideas that transcend the method and are relevant to many flow algorithms and problems: residual networks, augmenting paths, and cuts.

For distributed IoT network, we investigate flooding method to calculate max flow from in-between two nodes.

2.1 Implementation

In this section, we try to address the above node failure and mobility challenges in detail. In wireless IoT network, any node can fail any time or after a certain period. Secondly, the node has mobility property that means node position change at any time or after a certain period. These are two main challenges for implementing classical max-flow algorithm such as dinic's, Ford-Fulkerson or Edmond-Karp algorithm in wireless IoT network.

To get max-flow from source to destination in wireless IoI network, we consider Flooding technique to get the neighbor node information and virtual network topology. In the network layer, we implement Destination-sequenced distance vector (DSDV) routing protocol.

2.2. Flooding

A wireless IoT network consists of wireless mobile nodes that form a temporary virtual network topology without the aid of established infrastructure or centralized administration. This type of network can be easily constructed with inexpensive cost because of no need to install wire line infrastructures. Like a wired network topology, a wireless IoT network can be modeled as a graph. We can consider every mobile node as a vertex and two nodes are connected by a link (represent the edge) when they are within the transmission range of each other. Due to mobility, the graph may change with a certain time period [11].

To get max-flow from source to destination with minimum overhead, we investigate the Complete Flooding method.

2.3. Complete Flooding

For a distributed network, Complete Flooding method is as follows. Firstly, we focus on creating a temporary virtual network topology through flooding method. Source node *s* start broadcasting *hello* message with a certain transmission range tr and bandwidth capacity c. Within this transmission range, those nodes are present, they will receive the hello message and make a neighbor list of the source node s. Then each node from the neighbor list starts broadcasting hello message with a certain transmission range tr and bandwidth capacity c and make their individual neighbor list. All nodes will send their neighbor list to the source node through unicast message. Such a way, network topology will be constructed. In this method, each node will broadcast only one time. The complete flooding method algorithm is given below:

Algorithm 1: Complete Flooding Method

Nodes periodically exchange **(1 hop broadcast)** its local information, i.e., node capacity

Source node periodically floods the following message:
Msg1: (SrcAddr, TTL, SeqNo, "GETLOCAL") ;

On receiving GETLOCAL:
Check SeqNo to determine if the packet seen before.
If not, then if (TTL != 0)
Decrement TTL and rebroadcast GETLOCAL
Send the local info to the source using following message
MSG2: (NodeID, NghID, LinkCapa, "MYLOCAL") ;

Source receive all MYLOCAL:
Create the global network graph.
Calculate Overhead
Find the MAX flow to the destination using Dinic's algorithm.
Start packet transmission

Secondly, after constructing virtual network topology, we implement classical max-flow algorithm like *dinic's algorithm* to get max-flow from source *s* to destination *t*. However, in a distributed network, devices are light-weighted and low power capacity in nature.

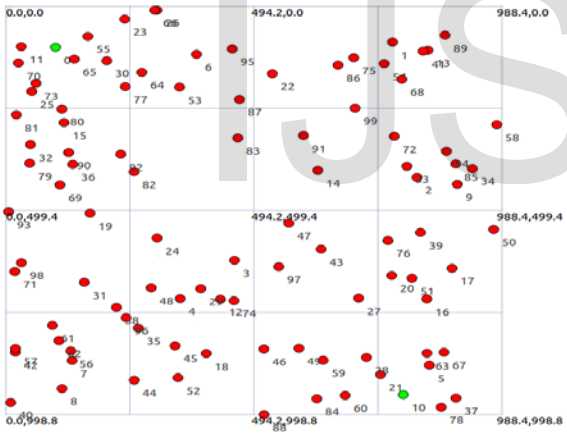

Figure 1. A distributed network containing nodes 100.

We consider three types of distributed IoT network.

1. All nodes position are fixed.
2. Every node has mobility except source and destination.
3. All nodes have mobility

2.3.1. *First Case*

In the first case, all node are uniformly distributed and have no node mobility property. To construct temporary virtual network topology, at first source node start flooding to get each node's neighbor information. Then, we have to check only node failure property in the network. To get maximum flow from source to the destination we can apply any classical max-flow algorithm. This is simple if we compare with other two implementations.

2.3.2. *Second Case*

In the second case, all node are randomly distributed and have node mobility property except for the source and destination node. That means all node will move with respect to time, however, source and destination node position are fixed. After creating virtual network topology, we have to take care of node failure as well as mobility property also.

2.3.3. *Third Case*

In the third case, this is similar to the second case. The only difference is each node has mobility property.

3. Results

In this section, for the three types of distributed network environments, we analyze and compare the result.

3.1. Flooding in Whole Network(Complete Flooding Method)

At first, we consider a distributed network(case 3) where node's failure and mobility property exists. In this network, the total number of the device is *N = 100* and each node has fixed transmission range *tr = 200* meters with different transmission capacity. The application starts at the time of 2 seconds and ends 60 seconds. That means, at time 2 seconds, source node starts flooding hello message, and it continues to the whole network.

Source Node	Sink Node	Time	Max-Flow
0	10	2	280
0	10	7.00174	300
0	10	12.0031	360
0	10	17.0077	100
0	10	22.0043	320
0	10	27.0052	410
0	10	32.0125	260
0	10	37.0063	660
0	10	42.0073	700
0	10	47.1525	500
0	10	52.0085	380
0	10	57.0087	375
Minimum			100
Average			387
Maximum			700

Table 1. Calculation of max-flow and message overhead for Case 3 at time 2 and 7 seconds

In table 1, for particular source 0 and destination 50, we calculate max-flow with the message overhead at the time of 2, 5, 12, 17, 22, 27, 32, 37, 42, 47, 52 and 57seconds. To check node failure and mobility flooding has been done every five seconds time interval. For each flooding, we get modified virtual temporary networks due node failure and mobility.

Node		Max-Flow			Message Overhead		
Source	Sink	Min	Avg.	Max	Min	Avg.	Max
0	10	100	387	700	679	1011	1375
0	50	100	230	320	679	1011	1375
25	75	0	157	280	2	919	1265
25	40	0	185	305	2	919	1265
45	90	105	435	735	688	1021	1241
45	70	100	398	690	688	1021	1241
45	60	245	442	625	688	1021	1241
80	40	85	190	310	689	1029	1315
80	55	80	172	290	689	1029	1315
38	65	85	216	335	674	1022	1314

Table 2. Max, Min and Avg. Flow From Source to Destination

Now, In table 2, we calculate the max, min and avg. max-flow with message overhead.

Number of nodes	100
Terrain range	1000 m × 1000 m
Transmission range	200 m
Simulation time	60 seconds
Node's mobility speed	0-10 m/s
Mobility model	Random way point
Propagation model	Free space
Channel bandwidth	1-3 Mbps
Links delay	20-200ms
Data payload	512 bytes/packet
Node pause time	0-10 seconds

Table 3. Simulation parameters in NS-3(Ver. 3.27)

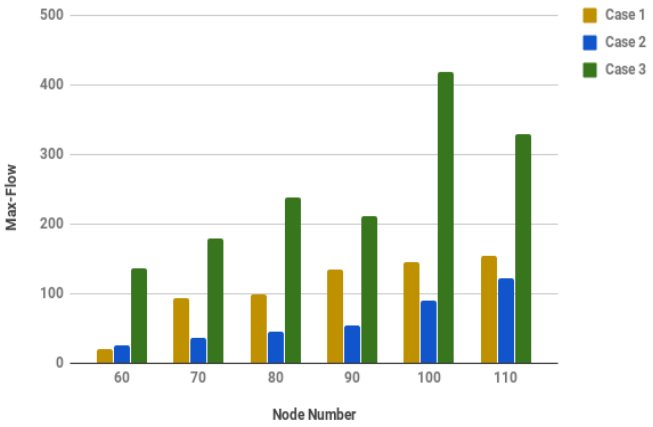

Figure 2. Comparison of avg. Max-flow for **case 1,2 and 3** for increasing node N.

Figure 2 depicts the comparison of max-flow with increasing node N for case 1,2,3. Here, we can conclude that if we increase node number in the same distributed network for case 1 and 2, the max-flow is increased linearly. However, For case 3, max-flow is increased randomly. In the figure for node number 100 max-flow is 420 but node 110 max flow is 360.

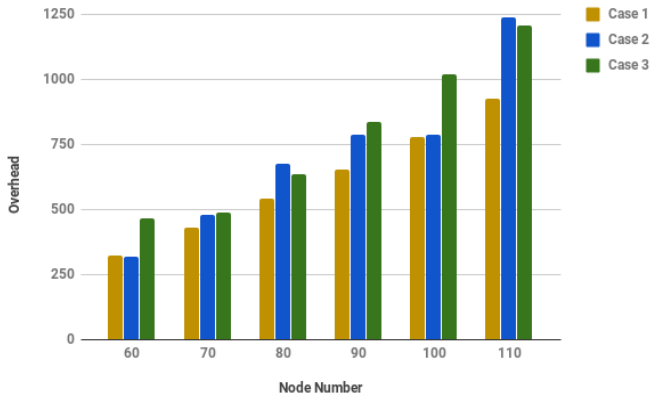

Figure 3. Comparison of avg. Message Overhead for **case 1, 2, 3** for increasing Node number N.

Figure 3 depicts the comparison of message overhead with increasing node N for case 1,2,3. Here, we can conclude that if we increase node number in the same distributed network for case 1, 2 and 3 the message overhead is increased linearly.

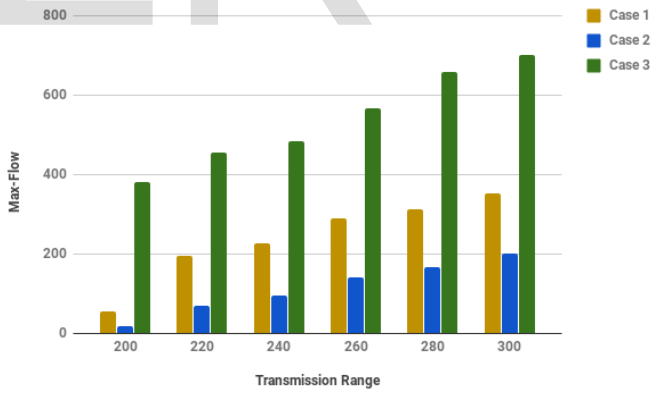

Figure 4. Comparison of avg. Max-flow for **case 1, 2, 3** for increasing transmission range tr..

Figure 4 depicts the comparison of max-flow with increasing transmission range tr for case 1. Here, we can comclude that if we increase the transmission range tr in the same network, the max-flow is increased linearly.

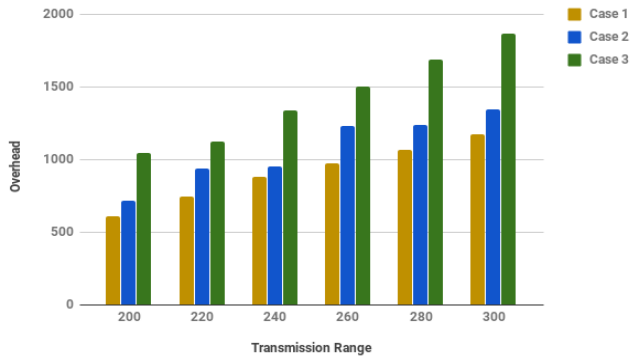

Figure 5. Comparison of message overhaed for **case 1,2 and** 3 with transmission range tr.

Figure 5 depicts the comparison of message overhead with increasing transmission range tr for case 1.Here also, we can conclude that if we increase the trasmisson range tr in the same network, the message overhead is increased linearly.

4. Discussion

In this project work, to calculate max-flow from source to destination, I have to choose any classical max-flow algorithm. In future, we will try to come up with a new algorithm or modification of conventional max-flow algorithm that will give max-flow with minimum message overhead for any distributed network. In our project implementation, we have considered some fixed parameter such as transmission range is same for all nodes. However, in practically, all nodes have different transmission range. We shall try to fix it.

5. Conclusions

In summary, this project work can be divided into two parts. Firstly, we focus on how we can implement the classical max-flow algorithm for any distributed networks considering node failure and mobility properties. In our implementation, we have chosen three different types of distributed environment named case 1, 2 and 3. To calculate max-flow from source to destination, we have used flooding technique. To check node failure and mobility, we send hello message after a particular period interval. Using this method, we can calculate max-flow for any distributed networks. However, in this method message overhead is more. We have found a trade-off relation between max-flow and message overhead. Secondly, we focus on how we can get max-flow with minimum message overhead.

References

1. T. L. Pham, I. Lavallee, M. Bui, and S. H. Do, "A distributed algorithm for the maximum flow problem," in *Parallel and Distributed Computing, 2005. ISPDC 2005. The 4th International Symposium on*. IEEE, 2005, pp. 131–138.
2. D. R. Ford and D. R. Fulke rson, *Flows in Networks*. Princeton, NJ, USA: Princeton University Press, 2010.
3. A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum- flow problem," *J. ACM*, vol. 35, no. 4, pp. 921–940, Oct. 1988. [Online]. Available: http://doi.acm.org/10.1145/48014.61051
4. A. Karzanov, "Determining the maximal flow in a network by the method of preflows," vol. 15, p. 434437, 02 1974.
5. Y. Dinitz, "Dinitz algorithm: The original version and evens version," in *Theoretical computer science*. Springer, 2006, pp. 218–240.
6. C. E. Perkins and P. Bhagwat, "Highly dynamic destination- sequenced distance-vector routing (dsdv) for mobile computers," in *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, ser. SIGCOMM '94. New York, NY, USA: ACM, 1994, pp. 234–244. [Online]. Available: http://doi.acm.org/10.1145/190314.190336
7. H. Narra, Y. Cheng, E. K. Cetinkaya, J.P. Rohrer, and J. P. Ster- benz, "Destination-sequenced distance vector (dsdv) routing protocol implementation in ns-3," in *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011, pp. 439–446.
8. C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, ser. WMCSA '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 90–. [Online]. Available: http://dl.acm.org/citation.cfm?id=520551.837511
9. C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," Tech. Rep., 2003.
10. T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," Tech. Rep., 2003.
11. H. Lim and C. Kim, "Flooding in wireless ad hoc networks," *Computer Communications*, vol. 24, no. 3-4, pp. 353–363, 2001