An Approach Based on Ford-Fulkerson Algorithm to Optimize Network Bandwidth Usage

Conference Paper · November 2015

DOI: 10.1109/SBESC.2015.21

CITATIONS
2

READS
2,266

2 authors, including:

Gustavo Rau de Almeida Callou
Universidade Federal Rural de Pernambuco
85 PUBLICATIONS **1,179** CITATIONS

SEE PROFILE

An Approach Based on Ford-Fulkerson Algorithm to Optimize Network Bandwidth Usage

Euclides Pinto Neto
Department of Statistics and Informatics
Federal Rural University of Pernambuco (UFRPE)
Recife, PE, Brazil
Email: ecpn@deinfo.ufrpe.br

Gustavo Callou
Department of Statistics and Informatics
Federal Rural University of Pernambuco (UFRPE)
Recife, PE, Brazil
Email: gustavo@deinfo.ufrpe.br

***Abstract*—Ford-Fulkerson algorithm is widely used to solve maximum Graph-Flow problems and it can be applied to a range of different areas, including networking. This paper proposes an approach based on Ford-Fulkerson algorithm to maximize the flow (bandwidth usage) of computer network. Such method mitigates congestion problems and increases network utilization. In order to show the applicability of the proposed approach, this paper presents the analysis of different network scenarios.**

***Keywords*-Networking, Ford-Fulkerson, maximum Graph-Flow, Bandwidth.**

I. Introduction

Internet is based on a single shortest path routing like Routing Information Protocol (RIP) [1], which advantage consists in its simplicity [2]. RIP was designed to work with moderate-size networks using reasonably homogeneous technology. This protocol may be suitable for many campuses and for regional networks where speed does not vary widely. However, such technique is not appropriate for complex situations where real-time load is used to dynamically define routes [4]. Although RIP is the simplest routing protocol, which is supported by almost all routers and many operating systems, it has several deficiencies, such as emerging routing loops and slow convergence [8]. Therefore, many extensions have been proposed as reroute mechanism [11], for instance.

Flow control allows sources to adapt their bandwidth usage to the network, increase network utilization, and mitigate congestion problems [6]. To optimize these parameters, a flow optimization application based on Ford-Fulkerson algorithm is proposed in this work. It builds directed graphs from network scenarios. In these graphs, edges represent links, and nodes represent devices. Every edge has its own capacity, which corresponds to the bandwidth, and flow usage.

This paper is organized as follows. Section II shows related works of this research. Section III presents the Ford-Fulkerson algorithm. Next, Section IV shows two case studies. Finally, Section V concludes the paper and presents future directions.

II. Related Work

Several works have proposed applications to optimize the flow in network scenarios. In [7], a class of flow control problems in a network which involves dynamic optimization is presented to maximize transmission rates. Mahlous et al. [5] propose an algorithm, based on a Ford-Fulkerson algorithm, to distribute load as well as improve network redundancy and security.

In [3], a real-time flow control algorithm is presented using metrics as feedback delay. However, the authors did not applied the approach into a well-known scenario considering similar links. Spasov et al. [8] propose an enhancing strategy to reduce RIP protocol deficiencies. However, even solving RIP's problems as routing loop, it does not considers real-time metrics as bandwidth usage. The authors in [9] propose an approach for routing packets. A standard routing protocol is used to find the optimal path between two nodes in a network. This work has not the focus on the quantification of the maximum amount of data that can be send through the network.

III. Ford-Fulkerson Algorithm

Ford-Fulkerson Algorithm is widely used in many problems [10] [12]. Assume that G is a finite directed graph where every edge (u, v) has a capacity $c(u,v) \geq 0$. The capacity means how much flow this link (edge) supports. Every link has a flow $f(u,v) \leq (u,v)$. Suppose G has two special vertices: source (s) and destination (t). The source (s) is the only node that can create flow in the network. The destination (s) is the only node which can use (destroy) the flow in the network. Any other node in the network cannot create or use flow.

A. Implementation

Our Ford-Fulkerson implementation is divided in functions, in which the first step is to generate the residual network. The residual network shows the amount of flow that can be added to each link. In order to accomplish that, the current flow is subtracted from the capacity of each link which results in the residual network(lines 2 and 3 of

2324-7894/15 $31.00 © 2015 IEEE
DOI 10.1109/SBESC.2015.21

Algorithm 1). Therefore, the residual network presents the actual state of the network showing the amount of flow that can be added for each link (represented as nodes of the G graph). The associated pseudo-code that represents such behavior is represented by Algorithm 1.

Algorithm 1: buildResidualNetwork(network)

```
1 residualNetwork = copy(network);
2 for edge in network do
3     edgeValue = edgeCapacity-edgeCurrentFlow;
4 return residualNetwork;
```

Secondly, in order to find an augmenting path, a set of links (edges) with value greater than zero that connects the source (s) to the destination (t) is found in the residual network. It is the only way to increase the flow's value. It is illustrated in Algorithm 2.

Algorithm 2: discoverAugmentingPath(residualNetwork, vertexOrigin, visitedVertices)

```
1  visitedVertices.add(vertexOrigin);
2  result = [];
3  if vertexOrigin==destination then
4      return 't';
5  for exitLink in vertexOrigin do
6      if residualNetorwkValueExitLink=0 then
7          continue;
8      else if vertexOrigin not in result then
9          result.add(discoverAugmentingPath(residualNetwork,
           vertexOrigin,(visitedVertices+vertexOrigin)));
10 return result;
```

Thirdly, in order to discover the smallest augmenting edge in the selected augmenting path, every single link is compared and the smallest is chosen, as illustrated in lines 3, 4 and 5 in Algorithm 3. Note that these augmenting paths are based on the residual network, which presents augmenting values in each link. The smallest value is important because that path cannot support a flow bigger than the smallest value, even if other edges can support.

Algorithm 3: getSmallestAugment(augmentingPath)

```
1 newNetwork = copy(network);
2 list = [];
3 for edge in augmentingPath do
4     lista.add(edge);
5 return min(lista);
```

Finally, the maximum flow, illustrated in Algorithm 4, is obtained by repeating all the previous steps. Line 2 builds the residual network. In line 3, an augmenting path is discovered and this process is repeat from lines 4 to 8, but the augmenting path is filled in line 6. The function to fill the augmenting path is illustrated in Algorithm 5. When augmenting paths are not available, the maximum flow is reached and the network with its maximum flow is returned (line 9). Note that a call to findMaximumFlow function accesses all the features presented in Algorithms 1, 2 and 3.

Algorithm 4: findMaximumFlow(network)

```
1 newNetwork = copy(network);
2 residualNetwork = buildResidualNetwork(newNetwork);
3 augmentingPath = discoverAugmentingPath(residualNetwork);
4 while augmentingPath not empty do
5     smallestAugment=getSmallestAugment(augmentingPath);
6     newNetwork =
      fillAugmentingPath(newNetwork,augmentingPath,smallestAugment);
7     newResidualNetwork = buildResidualNetwork(newNetwork);
8     augmentingPath= discoverAugmentingPath(newResidualNetwork);
9 return network;
```

Algorithm 5: fillAugmentingPath(network, augmentingPath, smallestAugment)

```
1 for edge in range(len(augmentingPath)) do
2     for k in network[augmentingPath[edge][0]])) do
3         if k[0]==augmentingPath[edge][0] then
4             k[2]+=smallestAugment
5 return network;
```

B. Example

Figure 1 shows the considered network scenario. This scenario is a directed graph, in which each edge has its capacity and the current flow. The first is depicted in red color and the current flow in green color. In addition, each source (s) and destination (t) nodes are represented as one node in the graph. Figure 2 shows the infrastructure presented in Figure 1 holding its maximum flow, which represents the output of our algorithm.

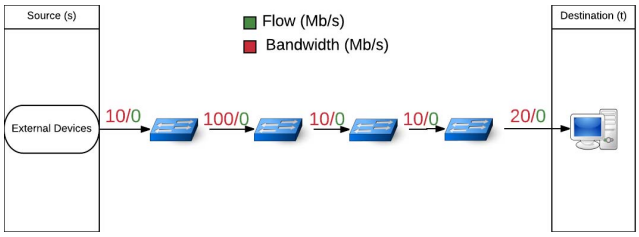

Figure 1. Network scenario representation.

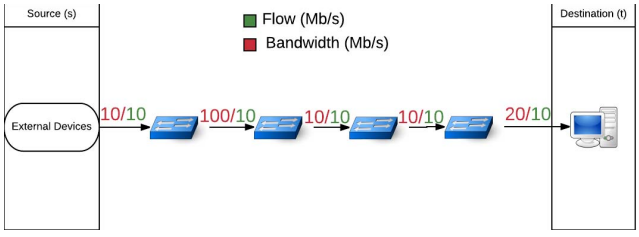

Figure 2. Maximum Flow for the network infrastructure of Figure 1.

IV. Case Studies

This section presents two case studies. The main goal of the first one is to analyze and maximize the bandwidth usage of a network that is composed of switches and host connected via Ethernet. The second case study has the same aim but considering a Fiber-optic communication network.

A. Case Study I

Figure 3 shows the network scenario with the bandwidth of each link. The bandwidth value of each link varies between 10Mb/s, 100Mb/s and 1024Mb/s. In this scenario, the source (s) is represented by external devices and the destination (t) is represented by the set of available hosts.

To find the maximum bandwidth usage (flow), the bandwidth of every link is compared with its current bandwidth usage and checked to improve or not by increasing the amount of data flowing through this communication link. However, the bandwidth limitation of every link must be respected in order to keep the network without congestion.

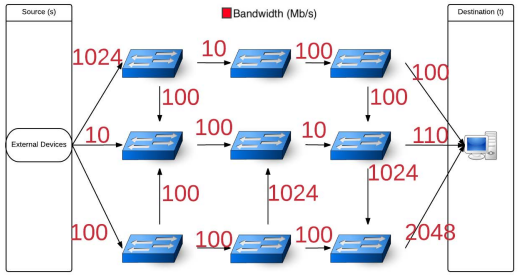

Figure 3. Network infrastructure and the correspondent bandwidth for each device.

Results. Figure 3 shows the bandwidth of links and Figure 4 shows how much flow the network can support. Note that although some links have capacity greater than its flow, it cannot be increased to respect the limitations of the network. For example, links have bandwidth 100Mb/s and bandwidth usage 10Mb/s even applying the maximum flow.

B. Case Study II

This second case study adopts considers a fiber-optic network as shown in Figure 4. The bandwidth value of each link varies between 1024 to 102400Mb/s. In this scenario, the source (s) is represented by external devices and the destination (t) is represented by a set of available hosts. To find the maximum bandwidth usage (flow), the bandwidth of every link is compared with its current bandwidth usage. After that, it is checked the possibility to improve the bandwidth by increasing the amount of data flowing through the communication link. The bandwidth limitation of every link must be respected in order to keep the network without congestion.

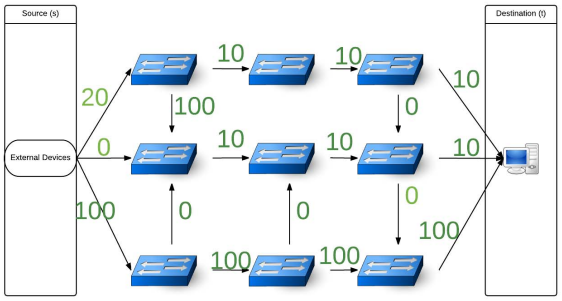

Figure 4. Maximum Flow for the network infrastructure of Figure 3.

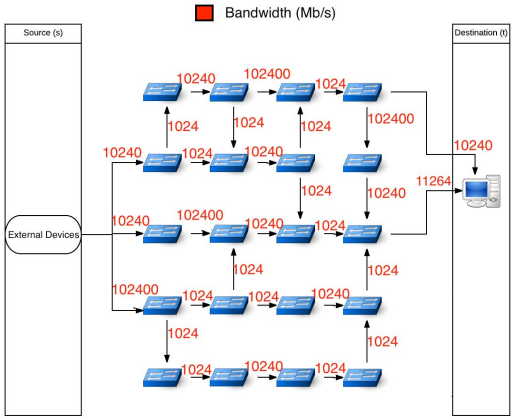

Figure 5. Network infrastructure and the correspondent bandwidth for each device.

Results. Figure 5 depicts each link capacity and Figure 6 shows its maximum flow (bandwidth usage). it is possible to note that some links have bandwidth 1024Mb/s and bandwidth usage 0Mb/s even considering the maximum flow of the network.

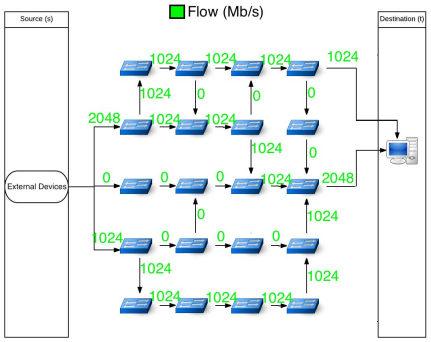

Figure 6. Maximum Flow for the network infrastructure present on Figure 5.

C. Discussion

The analysis of the FF algorithm applied on networking can provide support for IT designers. For instance, let us suppose a scenario in which a network does not treat the load balance problem. Therefore, adopting our approach, by running the FF based algorithm, IT designers will be able to see the bandwidth usage improvement that can be applied on that networking system. In networks without treatment, uncontrolled growth of flow may occur. Consequently, packets can be dropped.

Figures 7 shows a comparison of two scenarios considering the network architecture present on the second case study. The first scenario considers no one flow treatment using RIP (blue line). On the other hand, the second scenario adopts our proposed approach to control the network flow (green line). Assuming that a big amount of data (100 Petabytes) must be sent through that network. In this case, the maximum supported flow, respecting the network limitation, coming from the source is 3072Mb/s. Our algorithm increases the flow when it finds augmenting paths and keep the flow constant when there is no augmenting paths. In this case, when the flow reaches 3072Mb/s, which is the maximum flow supported, it remains constant. The network scenario using RIP increases the flow regardless how much flow the network supports. Therefore, values grater than 3072Mb/s can be reached and may result in congestion and packet dropping issues.

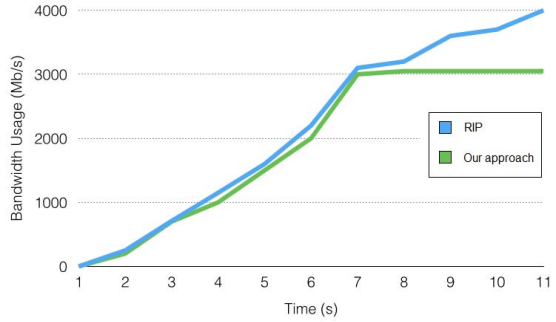

Figure 7. Source bandwidth usage (Mb/s) per Second (s) using RIP and our algorithm.

V. Conclusion

This paper presented a flow optimization algorithm based on the Ford-Fulkerson algorithm to improve network bandwidth usage. In order to accomplish this, the algorithm determines all augmenting paths in a network and increases each flow to the maximum value supported. Our solution is best indicated to optimize the bandwidth usage in networks where similar links are present on the network infrastructure. Through the experiments conducted, we realized that better improvements were detected once big amount of traffic is present.

As a future direction, we intend to extend our algorithm to work with other metrics (e.g., delay, data loss), set multiples sources (s) and destinations (t).

Acknowledgment

The authors would like to thank FACEPE and CNPq for the financial support of this project.

References

[1] G. Malkin,*"RIP version 2 protocol analysis"*, IETF Internet RFC 1721, November 1994.

[2] Fitigau, I., Toderean, G. , *"Network performance evaluation for RIP, OSPF and EIGRP routing protocols"*. Electronics, Computers and Artificial Intelligence (ECAI), 2013 International Conference on , vol., no., pp.1,4, 27-29 June 2013

[3] Steven H. Low and David E. Lapsley. , *"Optimization flow control, I: basic algorithm and convergence"*. IEEE/ACM Pansactions on Networking, 7(6):861-874, December 1999.

[4] Hedrick, C.,*"Routing Information Protocol"*, STD 34, RFC 1058,
Rutgers University, June 1988.

[5] A.R. Mahlous, R.J. Fretwell, B. Chaouran, *"MFMP: Max Flow Multipath Routing Algorithm"*, Second UKSIM European Symposium on Computer Modeling and Simulation 2008, EMS '08, 8-10 Sept. 2008.

[6] D. E. Lapsley and S. H. Low. , *"An IP Implementation of Optimization Flow Control"*, In Proceedings Globecom '98, November 1998.

[7] Imer, O.C.; Basar, T. , *"Dynamic optimization flow control"*, Decision and Control, 2003. Proceedings. 42nd IEEE Conference on , vol.3, no., pp.2082,2087 Vol.3, 9-12 Dec. 2003

[8] Spasov, D., Ristov, S., Gusev, M. , *"Enhancing and simulating the RIP routing protocol in cloud"*. Telecommunications Forum Telfor (TELFOR), 2014 22nd , vol., no., pp.372,375, 25-27 Nov. 2014

[9] Wojcik, R., Domzal, J., Dulinski, Z. , *"Flow-Aware Multi-Topology Adaptive Routing"*. Communications Letters, IEEE , vol.18, no.9, pp.1539,1542, Sept. 2014

[10] Dwivedi, A., Xinghuo Yu , *"A Maximum-Flow-Based Complex Network Approach for Power System Vulnerability Analysis"*. Industrial Informatics, IEEE Transactions on , vol.9, no.1, pp.81,88, Feb. 2013

[11] Bin Wang, Jian-hui Zhang, Wen-ping Chen, Yun-fei Guo , *"A Fast Reroute Mechanism for RIP Protocol"*. Circuits, Communications and Systems, 2009. PACCS '09. Pacific-Asia Conference on , vol., no., pp.74,77, 16-17 May 2009

[12] Ferreira, J.; Callou, G.; Maciel, P. , *"A Power Load Distribution Algorithm to Optimize Data Center Electrical Flow"*. Energies 2013, 6, 3422-3443.