Proposed Approach of Extended Johnson Algorithm Analysis for Load Balancing and Congestion Control in Software Defined Networking

M. Bindhu and G. P. Ramesh

Department of ECE, St. Peter's University, Chennai, India

Copyright © 2016 M. Bindhu and G. P. Ramesh. This article is distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

Software-Defined Networking (SDN), display cases of a paradigm shift from traditional network towards a new era of Internet. This paper offers different attempts to integrate congestion control (cc) and load balancing by combining SDN approach in conjunction with Extended Johnson Algorithm applied for sparse graphs. This proposed algorithm takes into consideration not only edge weights but also node weights with negative cycles. This algorithm produces efficient utilization of network resources, high throughput, jitter of all pairs with shortest paths in a sparse weighted, dense and directed graph. Implementation of this proposed algorithm is studied with OMNET++ emulation tool under the Abilene network topology. Various parameters like response time, throughput, latency, server load variation, and jitter are analysed for load balancing and congestion control in SDN. The results obtained are compared with other existing algorithms and performances of these parameters are discussed.

Keywords: Software-defined networking (SDN), Congestion Control (CC), Load Balancing, Throughput
1. Introduction

Software Defined Networking (SDN) technology separates the control plane and data plane of network devices [1] where a logically centralized controller configures the forwarding tables, which are responsible for forwarding the packets of communication flows. At present, network traffic is growing fast and complex as enterprises need to purchase more equipment to handle this complex network. Network congestion and server overload are the serious problems faced by this enterprises network. The traditional load balancing products cannot be used in data centre with virtualisation environment. Many business are transforming into cloud environment where the virtualisation technology help companies like cloud computing, big data in integration of servers with maximum utility [2], thereby saving money. Open Flow protocol [3] uses a Controller which can manage the traffic across the network for better load balance trying to reduce the channel congestion. This Extended Johnson Algorithm [4] can be applied to derive all pair’s shortest paths, in a sparse weighted, directed graph with the negative edge weights and node weights routing path for some specific SDN environment. This proposed algorithm is faster than Floyd–Warshall on sparse graphs which uses as subroutines both the Bellman-Ford and Dijkstra’s algorithm to find the shortest paths from each node to other vertex in the reweighted graph. Various parameters like response time, throughput, latency, server load variation, jitter are analysed for load balancing and congestion control in SDN.

This paper says about load balancing algorithm which helps in finding the shortest path using Extended Johnson algorithm for SDN based networks under OMNET++ emulation tool[4] under the Abilene network topology[5]. This paper is organized as follows: Section 2 briefs about the related works in SDN Load balancing, Section 3 describes proposed work algorithm, Section 4 discusses about the simulated results, Section 5,6 says about Comparison analysis of various parameters tabulated and graphs, finally section 7 Concludes with possible improvements in the work.

2. Related work

SDN based Load Balancing

The online services like e-commerce, websites, and social networks frequently use multiple servers to get high reliability and accessibility. The paper [6] proposed a load balancing algorithm, named LABERIO (LoAdBalancEd Routing wIth OpenFlow), minimizes latency and response time increases the network throughput. It uses ToR (Top of Rack) Switch-to-ToR Switch Paths Table (S2SPT) and Load Allocation Table (LAT). So, LABERIO is not suitable in wide area network because in the wide area we cannot predict the topology changes. The paper [7] proposed the Plug-n-Serve system implementing a load balancing algorithm, called LOBUS (LOad-Balancing over UnStructure networks) [8], using OpenFlow for unstructured networks yields the lowest total response time for each
newly arriving request. IP services uses Open Shortest Path First (OSPF), the computation is based on Dijkstra’s algorithm which calculates the shortest path within the network with disadvantage is ,a blind search is done which involves in time consumption and it cannot handle negative edges. The design [9] of an intra-data centre switch architecture encompassing the proposed transmission scheme is then presented with help of LDR optical signal inputs which are multiplexed is transmitted and received in intra data centre under open flow principles.

3. Proposed Extended Johnson Algorithm

The algorithm solves all pair’s shortest paths, in a sparse weighted, directed graph with negative numbers. This algorithm is faster than Floyd–Warshall on sparse graphs which uses as subroutines both Bellman-Ford and Dijkstra's algorithm to find the shortest paths from each node to other vertex in the reweighted graph. The idea of Extended Johnson’s algorithm is to re-weight all edges and make them all positive, then apply Dijkstra’s algorithm for every vertex. The great thing about this reweighting is, all position of paths linking any two vertices is increased by same amount and all negative weights become non-negative. The network congestion is prevented by sending the request to nearest by server with threshold value which normally is lower than specified one.

Time Analysis:

The time difficulty of this algorithm, using Fibonacci pile Dijkstra's algorithm [10] has $O(V^2 \log V + VE)$ and $O(VE)$ time for the Bellman–Ford stage. Thus, once the graph is sparse, the total time can be faster than the Floyd–Warshall algorithm, which cracks the similar problem in time $O(V^3)$[11] [12].

The node weight $n[w[v]]$ of $v$ is named according to Eq. (1),

$$nw[v] = \sum_{f \in flow\ bits(v)} \frac{\text{Capacity}(v)}$$

and the edge weight $e[w[e]]$ of $e$ is named according to Eq. (2).

$$ew[e] = \sum_{f \in flow\ bits(e)} \frac{\text{Capacity}(e)}$$

The technique of reweighting Assign new weights $\hat{w}$ to each edge as follows:

$$\hat{w}(u, v) = w(u, v) + d(s, u) - d(s, v)$$

$$w(u, v) + d(s, u) - d(s, v) \geq 0$$


**Extended Johnson's algorithm**

1. Compute $G'$, which consists of $G$ supplemented with $s$ and a zero-weight edge from $s$ to every vertex in $G$.
2. Run Bellman-Ford($G'$, $w$, $s$) to obtain the $d(s,v)$'s.
3. Reweight by computing $\tilde{w}$ for each edge.
4. Run on each vertex to compute, The extended Dijkstra algorithm uses $d[u]$ to store the distance of the current shortest path from the source node $s$ to the destination node $u$, and uses $p[u]$ to store the previous node preceding $u$ on the current shortest path.
5. Undo reweighting factors to compute $d$. 
8. $p[v] \leftarrow d[u]$.
9. if $p=\emptyset$ then
10. $s \leftarrow \text{min}(P).server$
11. else
12. $s \leftarrow \text{min}(Q).server$
13. return $s$

**4. Simulation and Results Analysis**

For Dijkstra’s algorithm under the Abilene network uses Pyretic to implement the proposed algorithms and compare it with related basic algorithms, i.e., the round-robin load-balancing algorithm and the randomized load-balancing algorithm we refer to [13].

**5. Comparison Analysis of various load balancing parameters**

1. **Response time:**

   Time taken for distributing the packets from the server to the host in balancing way.

   The figure 1 depicts the developed Extended Johnson algorithm values and compared with randomised, round robin algorithm values [15],[16], the following results are shown in table 1.

   Extended Johnson: 0.10ms, Round robin: 0.14ms, Randomised: 0.17ms

   Hence the extended Johnson algorithm produces average response time when compared to round robin &randomised where the response time is low with in-
stability for various number of clients. Extended Johnson algorithm approximately gives 33.33 % better response time than the other algorithms.

2. Through put:

Amount of data successfully moved from one place to another. The figure 2 depicts the developed Extended Johnson algorithm values and compared with randomised, round robin algorithm values [15], [16], the following results are shown in table 1. Extended Johnson: 720Mbps, Roundrobin: 700Mbps, Randomised: 680Mbps Throughput is higher for extended Johnson, Approximately 13.33 % better than the other algorithms.

3. Latency:

Network latency says about of time taken for a packet of data to move from one allocated point to another. The figure 3 depicts the developed Extended Johnson algorithm values and compared with randomised, round robin algorithm values [15], [16], the following results are shown in table 1. Extended Johnson: 1.1ms, Roundrobin: 1.8ms, Randomised: 2ms Latency time of Johnson algorithm increase, almost stabilize with higher node capacity.

4. Server load variation:

The standard deviation measures the amount of variation or dispersion from the average server load. The figure 4 depicts the developed Extended Johnson algorithm values and compared with randomised, round robin algorithm values [15], [16], the following results are shown in table 1. Extended Johnson: 0, Round robin: 0.6, Randomised: 1.2

The standard deviation measures the amount of variation or dispersion from the average server load which is almost negligible.
6. Tabulated Results

**No of clients: 4**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Response time ms</th>
<th>Latency ms</th>
<th>Throughput Mbps</th>
<th>Server Load Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomised</td>
<td>0.12</td>
<td>1.95</td>
<td>550</td>
<td>2.5</td>
</tr>
<tr>
<td>Round robin</td>
<td>0.11</td>
<td>1.75</td>
<td>560</td>
<td>0.5</td>
</tr>
<tr>
<td>Djistikra</td>
<td>0.9</td>
<td>1</td>
<td>650</td>
<td>0.5</td>
</tr>
<tr>
<td>Extended Johnson</td>
<td>0.85</td>
<td>0.9</td>
<td>640</td>
<td>0.3</td>
</tr>
</tbody>
</table>

**No of clients: 8**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Response time ms</th>
<th>Latency ms</th>
<th>Throughput Mbps</th>
<th>Server Load Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomised</td>
<td>0.12</td>
<td>1.98</td>
<td>650</td>
<td>1.5</td>
</tr>
<tr>
<td>Round robin</td>
<td>0.12</td>
<td>1.8</td>
<td>720</td>
<td>0.6</td>
</tr>
<tr>
<td>Djistikra</td>
<td>0.11</td>
<td>1.2</td>
<td>740</td>
<td>0</td>
</tr>
<tr>
<td>Extended Johnson</td>
<td>0.9</td>
<td>1.1</td>
<td>750</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**No of clients: 12**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Response time ms</th>
<th>Latency ms</th>
<th>Throughput Mbps</th>
<th>Server Load Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomised</td>
<td>0.17</td>
<td>2</td>
<td>680</td>
<td>1.2</td>
</tr>
<tr>
<td>Round robin</td>
<td>0.14</td>
<td>1.8</td>
<td>700</td>
<td>0.6</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>0.11</td>
<td>1.2</td>
<td>710</td>
<td>0</td>
</tr>
<tr>
<td>Extended Johnson</td>
<td>0.10</td>
<td>1.1</td>
<td>720</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Tabulated results
7. Graphs

Figure 1: Response time

Figure 2: Throughput

Figure 3: Latency

Figure 4: Server Load Variation

8. Conclusion

This paper described a performance analysis of various parameters used in load balancing method in virtual environment based on OpenFlow technology. The extended Johnson algorithm is been used in analysing the parameters like response time, throughput, latency and server load variation. OpenFlow method grants flexibility for the realization of different load balancing strategies, which is conveniently used in software-defined network to achieve different load balancing strategies in different network environment. Extended Johnson algorithm approximately gives 33.33 % better response time than the other algorithms. The Throughput is 13.33 % better than the other algorithms. Also Latency time increase, almost stabilize with higher node capacity. The algorithm results in efficient utilization of network resources with high throughput in shortest paths identification in a sparse weighted, dense and directed graph.

References


**Received: March 8, 2016; Published: May 4, 2016**