An Algorithm for Computing Average Packet Delay

M. R. Hassan m_r_hassan73@yahoo.com
Computer Science Department,
Faculty of Science, South Valley University,
Aswan, Egypt.

Abstract

Average Packet delay is considered as a vital performance measure for a computer-communication network especially in the network designing problem. Average Packet delay evaluation depends on two main parts: the first part is the capacity of each link in the network, the last one is the flow of each link. The capacity of each link is assumed to be fixed but the flow of each link is computed by using routing algorithms and the traffic requirement matrix. The paper presents an algorithm based on FLOYD's routing algorithm to calculate the flow of each link and then we can compute the average packet delay.

Keywords: Computer Networks, QoS, Average Packet Delay.

1. INTRODUCTION

Performance measures [1], eg, delay, throughput are very important measures in designing reliable networks. The performance measure, Average Network Throughput (ANT), which takes into consideration the network topology, link reliabilities, the capacity of the links in the network, and the total installed capacity, [2] and [3]. The optimization priblem of ANT has been studied by many researches, [4-6].

The other vital performance measure for a communication computer network is the average packet delay, T, that is, the mean time that a packet takes to travel from a source node to a destination node in the network, [7]. The average packet delay optimization problem formulated and solved in [8-10]. The average source-to-destination packet delay is considered as an important criteria in the computer network design problems [11].

This paper presents a simple algorithm to calculate the average packet delay (T) of a given computer network by generating the set of all shortest paths by using Floyd's algorithm, [12], to determine the route to be taken by packets between each source-destination pair of nodes based on Euclidean distance between them. Then assumed the traffic that has been actually carried by each link in the network to calculate the flow of this link.

The paper is organized as follows: The assumptions and notation used given in Section 2. Section 3 describes the problem of calculating the average packet delay. Section 4 presents the proposed algorithm for calculating the average packet delay. Section 5 shows how to use the proposed algorithm to calculate the average packet delay for a given network. Conclusion and future work are given in section 6.

2. NOTATIONS AND ASSUMPTIONS

Notations:

N is the number of nodes.
M is the number of links in the network.
C_k is the capacity of link k.
F_k is the flow of link k.
L_k is the length of link k.
γ_{ij} is the traffic between nodes i and j.
γ is the total traffic in the network.
$1/\mu$ is the average packet length.

Assumptions:

1. The location of each network node is given.
2. The traffic between each node-pair has a Poisson distribution.
3. The packet size has exponential distribution with mean $1/\mu$ (bits/packet).
4. The nodal memory is infinite.
5. Independence between interarrival and transmission times on each link.

3. PROBLEM DESCRIPTION

The computer network is modeled as a directed graph G (N, L), |N| = n and |L| = *m*, where n represents the number of nodes of the network and *m* represents the number of links of the network. If we consider the sample network of 4 nodes and 5 links (n = 4 and *m* = 5) as shown in Fig. 1.

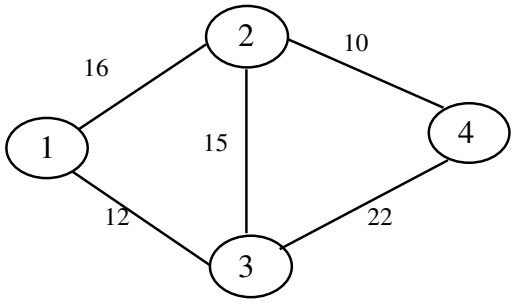

FIGURE 1: A sample network

The distance matrix is:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 16 & 12 & 0 \\ 16 & 0 & 15 & 10 \\ 12 & 15 & 0 & 22 \\ 0 & 10 & 22 & 0 \end{bmatrix} \end{array}$$

FIGURE 2: The Distance matrix for the network in Figure1.

The shrtest paths are as follows by using Floyd's algorithm, [12] :-

P_{12} = 1-2
P_{13} = 1-3
P_{14} = 1-3-4
P_{23} = 2-3
P_{34} = 3-4
P_{24} = 2-4

The flow of each link can be computed by summed the packets that travel across that link. The total number of packets that carried by the link depends on the number of shortest paths contains that link. The following table summerizes the link and the paths that contains that link.

The link	The paths
1-2	Appears in P_{12}
1-3	Appears in P_{13} and P_{14}
2-3	Appears in P_{23}
2-4	Appears in P_{24}
3-4	Appears in P_{13} and P_{34}

TABLE 1: The links and the corresponding paths

So, if the number of packets equal *p* then the following table summerize the number of packets carried by each link.

The link	The number of packets
1-2	*p*
1-3	*2p*
2-3	*p*
2-4	*p*
3-4	*2p*

TABLE 2: The number of packets carried by each link

So, the total flow is equal to *14p*, if we consider the link is bidirectional.

In the following subsections we will describe the main parts of the algorithm to calculate the flow of a given network and then calculate the average delay.

3. 1. Shortest Path Generation

We wil use Floyd's all-pairs shortest path algorithm, [12], to find all shortest paths from the source node to the destination node by using the distance matrix as follows:

SubAlgorithm_1

Input:
V = set of vertices, labeled by integers 1 to N.
E = set of edges, labeled by ordered pairs (*u, v*) of vertex labels.
D[u][v]: The distance matrix. *D[u][v]=0* if $u \neq v$ and $(u, v) \notin E$ or *u=v*
P[u][v]: the shortest path from *u to v* . *P[u][v]=u and P[u][v]=0* if $u \neq v$ and $(u, v) \notin E$ or *u=v*.

For all edges (*u, v*) in E:
W[u][v] = D[u][v].. And *W*(*u, v*) = infinity if $u \neq v$ and $(u, v) \notin E$ or *u=v*.

```
Begin
 for(k =1; k<= n; k++)
  for(u =1; u<= n; u++)
    for(v =1; v<= n; v++)
        W[u][v] = min (W[u][v], W[u][k]+ W[k][v])
          P[u][v]=k;
end.
```

3. 2. Finding the Intermediate Nodes for Each Shortest Path

If $P(i, j)=V_q$ then intermediate nodes on shortest path from i to j can be deduced as follows:

SubAlgorithm_2:

```
for(i=1;i<n;i++)
  for(j=i+1;j<=n;j++)
          Begin
           if(p[i][j]!=i) then
          repeat
                    q++;
                     v[q]=p[i][v[q-1]];
          until v[q]=v[q-1]
    for(k=1;k<q;k++) print  v[k];
End
```

3. 3. Calculate the Flow of Each Link

For each path generated in 3.2 count the total traffic that carried by each link connects the corresponding pair of nodes:

SubAlgorithm_3
Initially set flow[][]=0;
For each path generated in 3.2 do
Begin
If the path contains q vetices then
For(k=1; k<q,k++)
flow[i][V[k]]+= $\gamma_{iv[k]}$;
END

3. 4. The Total Traffic
The total traffic across the network is calculated by:

$$\gamma = \sum_{i,j}^{n} \gamma_{ij} \qquad \ldots(1)$$

SubAlgorithm_4:
Set γ = 0
for(i=1;i<n;i++)
for(j=i+1;j<=n;j++)
$\gamma = \gamma + \gamma_{ij}$

3. 5. The Average Packet Delay
The Average packet Delay, the mean time that a packet takes to travel from a source node to a destination node in the network is given by [11]:

$$T = \frac{1}{\gamma} \sum_{i=1}^{m} \frac{f_i}{c_i - f_i} \qquad \ldots.(2)$$

4. AN ALGORITHM FOR COMPUTING THE AVERAGE PACKET DELAY

The steps of the algorithm for calculating the average delay of the computer network are as follows:

Step 1: Read the distance matrix D and the capacity for each link C_i
Step 2: Generate shortest paths using Sub*Algorithm_1* and deduce the intermediate nodes for each path using SubAlgorithm_2.
Step 3: Use SubAlgorithm_3 to calculate the flow of each link, f_i
Step 4: Calculate the total traffic, γ by using Sub*Algorithm_4.*
Step 5: Calculate the Average Packet Delay, T, using equation (2).

Note: The algorithm has been implemented using VC++ 6.0.

5. CASE STUDY

To illustrate the proposed algorithm for computing the Average Packet Delay, consider an example networks taken from [9]. As shown in figure 2 the network has 8 nodes 13 links. The numbers written in bold represent the link lengths.

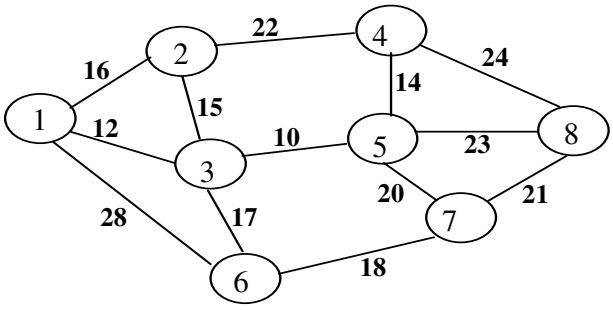

FIGURE 3: Example network

Table 3 shows the link flows (computed by the proposed algorithm) and corresponding capacities (taken from [10]) for the given network shown in Figure 1.

Link	Flow (Kbps)	Capacity (Kbps)
1 - 2	10	19.2
1 - 3	50	56
1 - 6	10	19.2
2 - 3	40	56
2 - 4	20	56
3 - 5	100	200
3 - 6	40	100
4 - 5	30	56
4 - 8	20	56
5 - 7	50	56
5 - 8	30	56
6 - 7	20	56
7- 8	20	56

TABLE 3: Link flows and Capacities

The total traffic, $\gamma = 65.$

The Average Packet delay (T) is equal to 0.419 s

The following table shows the link flows (computed by the proposed algorithm) and corresponding capacities for the given network with 10 nodes and 14 links shown in Figure 2, taken from [8].

Link	Flow (Kbps)	Capacity (Kbps)
1 - 2	80	100
1 - 5	220	230.4
2 - 3	100	230.4
2 - 4	60	100
3 - 8	140	230.4
4 - 5	60	100
4 - 8	80	100
5 - 6	100	230.4
5 - 7	180	230.4
6 - 7	60	100
6 - 10	80	100
7 - 9	140	230.4
8 - 9	160	230.4
9 - 10	100	230.4

TABLE 4: Link flows and Capacities

The total traffic, $\gamma = 140$.
The Average Packet delay (T) is equal to 0.3423 s

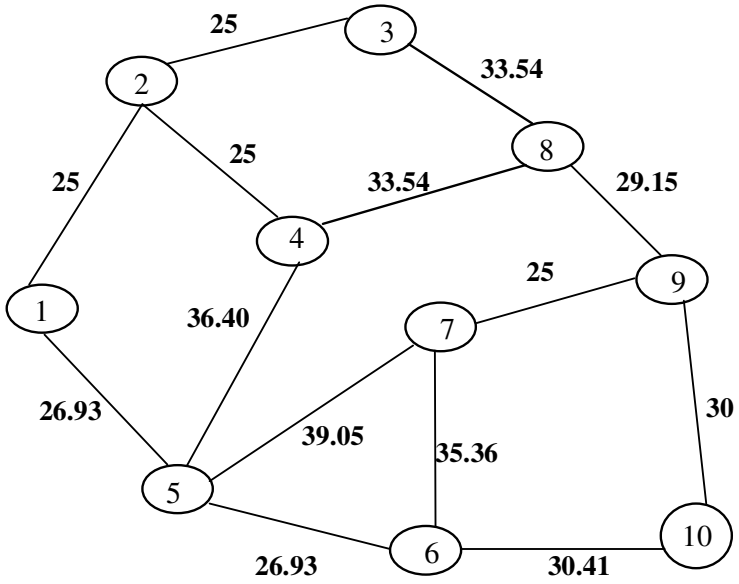

FIGURE 4: Example network taken from [8].

The above results shows that the proposed algorithm is working properly and efficient for the computer networks with large number of nodes.

6. CONCLUSION AND FUTURE WORK

This paper presented a simple algorithm to calculate the average packet delay. The algorithm is based on determining the flow of each each link of a given network. The flow claculations depends on the shortest path generated by the routing algorithm. Finally we illustrate the using of the proposed algorithm by calculating the average packet delay to a given sample network. In the future work we hope that the algorithm can be used in the average delay optimization problems.

7. REFERENCES

1. Tokumi Yokohira, Masashi Sugano and Hideo Miyahara, “Fault Tolerant Packet-Switched Network Design and Its Sensitivity”, IEEE Transactions on Reliability, 40(4):452-460, 1991.

2. Ahuja Sanjay P. and Kumar Anup “Reliability and Performance based measure for computer networks and distributed systems “, IEEE Southeastcon, Charlotte, NC,1993.

3. M. R. Girgis, A. Younes and M. R. Hassan, “An Algorithm for Computing Throughput of Computer Networks”, Egyptian Informatics Journal, 9(1):205-218, 2009.

4. Ahuja Sanjay P., “Performance based reliability Optimization for computer networks “, Engineering the New Century Conference Proceedings-IEEE-Southeastcon. IEEE, Piscataway, NJ, USA, 97CB36044, PP. 121-125, 1997.

5. Kumar Anup, Elmaghraby Adel S. and Auja Sanjay P., "Performance and reliability optimization for distributed computing systems", 3rd IEEE Symposium on Computers and Communications, Athens, Greece, pp. 611-615, 1998.

6. M. R. Girgis, A. Younes and M. R. Hassan, “Optimizing The Performance-Based Reliability For Computer Networks By Using Fuzzy Optimization With Genetic

Algorithms", International Journal of Computational and Applied Mathematics, 2(2):139–148, 2007.

7. S. Pierre and A. Elgibaoui," Improving Communication Network Topologies Using Tabu Search", Proceedings of 22nd Annual Conference on Local Computer Networks, pp. 44-53, 2-5 Nov 1997.

8. R. Beaubrun and S. Pierre, " A Routing Algorithm for Distributed Communication Networks", Proceedings of 22nd Annual Conference on Local Computer Networks, pp. 99-105, 2-5 Nov1997.

9. Peirre Samuel and Legault, "A genetic algorithm for designing distributed computer network topologies", IEEE Transactions Systems , MAN and Cybernetics-Part B: Cybernetics, 28(2):249-258, 1998.

10. M. R. Girgis, A. Younes and M. R. Hassan, "Optimizing the transmission delay of a computer network by using fuzzy optimization with genetic algoritms", International Journal of Intelligent Computing and Information Science, 8(1):163-171, 2008.

11. Gerla, M., and Kleinrock. L. "On theTopological Design of Distributed Computer Networks". IEEE Transactions on communications,25(1): 48-60,1977.

12. Maclej M. Syslo, "Discrete Optimization Algorithms: With Pascal Programs", Prentice Hall, (1983).