

Design Infrastructure-based Overlay Network to Reduce Multi-cast Delay

Shaoyu Yang, Bing Wang, Yoo-Ah Kim

Computer Science & Engineering Department,
University of Connecticut, Storrs, CT, 06269

Abstract—In an intra-domain network where OSPF or IS-IS protocol is used, the underlay routing may not be optimized. The delay in the shortest weight path may be much larger than delay in the shortest delay path from a sender to a receiver. For delay sensitive applications, we investigate the possibility of adding proxies in networks to make the delay in the overlay path between senders and receivers close to the delay in shortest delay path and to reduce the delays from senders to receivers. By solving the integer linear programming formulations developed on the overlay logical network, we get optimization results: number of optimal proxies, optimal proxy locations and the improvement in reducing delays from senders to receivers. We also apply Dijkstra’s algorithm on the overlay logical network to explore the maximum benefit of placing proxies on reducing delays. From simulation results of 6 ISP networks, we observe that adding proxies significantly reduces the delay from senders to receivers for some networks. Finally, we identified a network metric, *Default Delay Minimum Delay Ratio(DDMDR)*, which will provide guidance on whether adding proxies will significantly reduce the delays from senders to receivers in a network.

I. INTRODUCTION

Internet is a pervasive media for large scale group oriented communications such as video-conferencing, online-gaming, chart-room, IPTV, and long-distance learning. These applications have stringent bandwidth and/or delay requirements and can often involve a large number of users. The current Internet, however, only provides a single class of best-effort service, with no delay or bandwidth guarantee to the applications. Due to the deficiencies of the routing protocols [1] and/or some internal policies in setting the physical routes [2], the routing may not be able to find the optimal path from a sender to a receiver. The application layer overlay routing usually can compensate the un-optimized network layer routing [3]. Adding proxies in network to form an overlay network is one way to reduce the delay from the senders to receivers and to reduce the network link bandwidth usage.

We designed overlay networks that can minimize the delays from senders to receivers in multiple multicast groups, especially for the applications with low bandwidth requirements. In intra-domain routing, protocols such as OSPF and IS-IS find the path with shortest link weights [4]; we term the shortest link weight path as *default path*. If the link weights are not set properly, the delay of the searched default path may be much larger than the delay in the shortest delay path. By adding one or more proxies at proper router location(s), we can create a path from a sender to a proxy and from the proxy

to a receiver; we term the path through one or more proxies as *overlay path*. The created overlay path may have much lower delay than that of the default path. With the added proxies, we can find overlay paths whose delays are close to the delays in the shortest delay paths. If the overlay path doesn’t have delay improvement for a sender-receiver pair, the pair will use the default path for data transmitting. Furthermore, multi-cast trees can be created from senders and packets can be replicated at proxies to save the sender and the network bandwidth.

Many works have been done on improving the performance in transmitting data to large number of users [5], [6], [7]. Two existing techniques, end-host-only overlay [6] and Content Delivery Networks(CDNs) have been developed and successfully applied in large scale group communications. However, the scalability and performance of the two techniques may not be able to meet the increasing delay and bandwidth requirements of the applications. Even through end-host-overlays don’t need additional infrastructure and are easy to deploy, the control of the overlay network with large number of end hosts is difficult and therefore limits the number of end users in applications. Since CDN servers are typically placed at the edge of the network and messages are directly sent from content server to the CDN server, the content server needs to make multiple copies of the message and send each copy to each CDN server. This may lead to a significant content server and network bandwidth usage for group communications.

In our research, we first create overlay logical network on top of physical network and use linear programming to find the optimal number of proxies and their optimal locations. By changing the maximum number of proxies in linear programming formulation, we can discover how the number of proxies affects the maximum percentage improvement of total delays. Second, we run Dijkstra’s algorithm on the overlay logical network to find the shortest delay paths from senders to receivers. By comparing the total delays of the overlay paths with the delays of the default paths, we can calculate the maximum percentage improvement when we don’t limit the number of proxies. Third, we study how the link weight setting affects the total delay percentage improvement when adding proxies in one single ISP. Since the link weight is the main parameter that affects the routing default path, our research results may help network administrators in setting proper link weights to optimize the network. Fourth, we identify a network characteristic metric which provides guidance on

whether adding proxies will reduce the delays from senders to receivers.

For this work, we focus on placing proxies within a single ISP and leave the problem of proxy placement in the inter-domain as future work. We apply our algorithms and solutions to 6 ISP networks in USA, Australia and Europe where network topology, link weights and delays were measured by Rocketfuel [8], [4]. The simulation results prove that some ISPs are not optimized and adding proxies and using overlay path can reduce the delay from senders to receivers.

The rest of the paper is organized as follows. Section II describes the problem setting. Section IV studies the average delay improvements as increasing the number of proxies by using linear programming. Section V describes maximum benefits in reducing delay by using Dijkstra’s algorithm. Section VI concludes the paper and presents future work.

II. PROBLEM SETTING

Consider a physical network, represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where \mathcal{V} represent the set of nodes (routers) in the network and \mathcal{E} is the set of physical links connecting the nodes. Each link l has delay d_l . We assume that the underlying routing protocol determines the default IP path based on metrics other than the delay — e.g., the weights of links — and therefore, the default IP path between two nodes may not necessarily be the path with the minimum delay. For any pair of nodes, let $D_p(u, v)$ denote the delay on the default IP path between u and v . Then $D_p(u, v)$ is simply the sum of delays of links on the default IP path.

Our goal is to place proxies at appropriate places so that we can minimize delays experienced in group communications. Suppose that we are allowed to choose at most N_V proxies from V and at most N_E pairs of proxies can be maintained as overlay links. We assume that proxies can be located only at the routers. Let S denote the set of sources. Each source $s \in S$ has a group of receivers, denoted as R_s . A source and its corresponding receivers form a multicast group, denoted as (s, R_s) . Let $R = \bigcup_s R_s$. Once we choose a set of proxies, we can use proxies as relays to send data from a source to receives. That is, s can send data to a proxy first and then the proxy forwards the data to another proxy or to the receiver directly. Let $(s, o_1, o_2, \dots, o_k, r)$ be the overlay path from s to r where o_i ’s are proxies. Recall that $D_p(u, v)$ is the delay over the default IP path between u and v . Then the delay $D_o(s, r)$ of an overlay path between u and v is given as $D_p(s, o_1) + \sum_{i=1}^{k-1} D_p(o_i, o_{i+1}) + D_p(o_k, r)$.

Given a set of multicast groups, our problem is to find the optimal placement of proxies and an overlay path between each source and receiver pair so that we can minimize the delay experienced by end users. In particular, we consider two objective functions — minimizing the total delay over all source and receiver pairs

$$\sum_{s \in S} \sum_{r \in R_s} D_o(s, r), \quad (1)$$

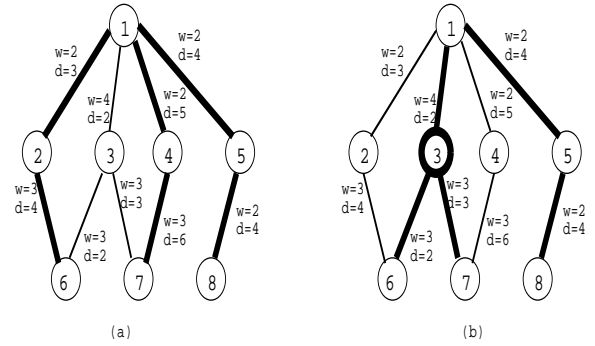


Fig. 1. Illustration of default path, overlay path and placement of proxies. Link weights and delays are marked on links and a proxy is placed at router node 3

and minimizing the maximum delay

$$\sum_{s \in S} \max_{r \in R_s} D_o(s, r). \quad (2)$$

Fig. 1 illustrates how proxies and overlay links can be used to reduce the delay. Suppose that the routing protocol determines the default IP path based on edge weights and node 1 wants to send data to node 6, 7 and 8. The network in Fig. 1 (a) shows a physical network and thick lines indicate the default path from the sender to all the receivers. For example, the default path from node 1 to node 6 is (1, 2, 6) which has the total weight of 5 and the total delay of 7. For node 7, path (1, 4, 7) is the default path and the total delay of the path is 11.

To reduce the delay, we can place a proxy at node 3 (See Fig. 1 (b)). By using the proxy as a relay, node 6 can now receive data over path (1, 3, 6) and node 7 can use path (1, 3, 7). The delay to node 6 and node 7 is reduced from 7 to 4, and from 11 to 5, respectively.

III. OPTIMAL PLACEMENT OF PROXIES

In this section, we present the algorithms to find an optimal placement of proxies to minimize end-to-end delays. In fact, we can show that for both metrics — (1) and (2) — the problem is NP-hard by a reduction from the SET COVER PROBLEM. The NP-hardness proof is given in the Appendix.

We first consider a special case of problem where there are no constraints on the number of proxies and overlay links. In this case, the problem can be solved using the shortest path algorithm. The results give the maximum delay benefits of using proxies without considering the cost of overlay deployment. When we can use only N_V proxies and N_E overlay links, we formulate the problem as an integer linear programming. Even though solving ILP is still NP-hard, we can obtain an optimal solution of ILP for graphs from real networks in a reasonable amount of time, using CPLEX.

A. Maximum Benefit of Proxies

When there are no constraints on the number of proxies and overlay links, we can find an optimal placement and multicast trees with minimum total delay by using the shortest path

algorithm. Given a physical network graph \mathcal{G} , we can construct a logical graph $G = (V, E)$, which is a complete graph where $V = \mathcal{V}$ and each edge in E represents a possible overlay link. A logical link e in E connecting a pair of nodes represents the default IP path determined by the underlying routing protocol. Each edge e (logical link) has its delay $l(e)$, which is the same as the delay on the default IP path.

We can now find the shortest path between each pair of source and receive (s, r) , $r \in R_s$ in G , based on $l(e)$. Any internal node on the shortest path between (s, r) has to be a proxy. The solution gives the maximum benefit of overlay networks when we are allowed to use as many proxies as necessary.

In the following subsection, we consider the case when the number of proxies and overlay links are limited and find an optimal placement of proxies using integer linear programming approach.

B. Limited Number of Proxies

Once we restrict the number of proxies that we can use, the problem becomes NP-hard by a reduction from THE SET COVER problem. To minimize the end-to-end delays from senders to receivers, we formulate the problem as integer linear programming (ILP). We have obtained optimal solutions to ILP for 6 real networks, using CPLEX.

Our ILP formulation is as follows: We consider a complete logical graph $G = (V, E)$ as in Section III-A. Let x_v represent whether a vertex $v \in V$ is chosen as a proxy. The value of x_v is 1 if v is chosen as a proxy, and 0 otherwise. Similarly, let x_e represent whether an edge $e \in E$ is selected as an overlay link connecting two proxies in the overlay network. Then, for an edge $e = (u, v)$, we have Constraints (9) $x_e \leq x_u$ and $x_e \leq x_v$. That is, x_e can be an overlay link only if both u and v are proxies. We can choose at most N_v proxies and N_e overlay links, which gives Constraints (10) and (11), respectively.

We now construct a multicast tree using overlay links. We assume a single path is used to deliver data from a source to a receiver. Let $f_r^s(u, v)$ represent whether we use the overlay link (u, v) to send data from s to r where $s \in S$, $r \in R_s$. Then we have Constraints (13) $0 \leq f_r^s(u, v) \leq x_e$ for $u \neq s$, $v \neq r$, and $e = (u, v)$ as we can use the link only when it is selected as an overlay link. Note that the above constraint does not apply to the logical link of (s, r) since a source and receiver pair is always allowed to use the default IP path between them. Furthermore, we allow source s to send to a proxy node v even if the link (s, v) is not an overlay link, which leads to Constraints (14), $0 \leq f_r^s(s, v) \leq x_v$. Similarly, we have Constraints 15, $0 \leq f_r^s(v, r) \leq x_v$ for receiver r . Constraints (16)-(20) ensure that exactly one unit of flow is sent from s to r , which determines a unique path from s to r .

Recall that $l(u, v)$ denotes the delay of the default path between u and v , $e = (u, v)$. Let $D(s, r)$ denote the delay from source s to receiver r . Then $D(s, r)$ can be computed as $\sum_{e \in E} f_r^s(e)l(e)$, which gives Constraints (21).

Our objective is to minimize the sum of $D(s, r)$ over all source and destination pairs or the sum of maximum $D(s, r)$

TABLE I

ISP AS3257 LINEAR PROGRAMMING RESULTS BY VARYING THE NUMBER OF PROXIES. NUMBER OF PROXIES, TOTAL DELAY, NUMBER OF PATHS THAT USED PROXIES AND DELAY PERCENTAGE IMPROVEMENT(DPI)

Proxy Num	Total Delay(ms)	Num of Paths use Proxies	DPI(%)
0	562	0	0
1	480	16	17.1
2	440	23	27.7
3	425	26	32.2
4	418	30	34.4
5	414	32	35.7
6	413	29	36.1

TABLE II

ISP AS1755 LINEAR PROGRAMMING RESULTS BY VARYING THE NUMBER OF PROXIES. NUMBER OF PROXIES, TOTAL DELAY, NUMBER OF PATHS THAT USED PROXIES AND DELAY PERCENTAGE IMPROVEMENT(DPI)

Proxy Num	Total Delay(ms)	Num of Paths use Proxies	DPI(%)
0	635	0	0
1	546	15	16.3
2	531	24	19.6
3	522	19	21.6
4	514	28	23.5
5	507	29	25.2
6	503	29	26.2
7	499	28	27.3
8	497	30	27.8
9	496	31	28.0
10	495	28	28.3

in each multicast group.

IV. LINEAR PROGRAMMING RESULTS

We used ILOG CPLEX solver to solve the integer linear programming formulation. CPLEX is a robust and reliable mathematical programming optimizer with high performance. We ran linear programming on ISP networks AS3257 and AS1755. We randomly chose 2 senders in each network and randomly chose 20 receivers for each sender. We ran linear programming solver to calculate the *total default path delay* from all senders to all receivers by setting the number of proxies to zero. When adding proxies, the linear programming solver can return the *total overlay path delay* from all senders to all receivers in the results. By varying the number of proxies in constraint (10), we ran LP formulation to get different total overlay delays. We use *Delay Percentage Improvement(DPI)* defined in II to show the benefits of increasing the number of proxies.

Table I and Table II show the linear programming results of one case in ISP AS3257 and ISP AS1755, respectively. The two tables show the number of proxies, total delay, number of paths from senders to receivers that use one or more proxies and the *Delay Percentage Improvement(DPI)* associated with the number of proxies. When the number of proxies is zero, the total delay is the total delay of default paths. In the test case of each ISP network, there are 2 senders and each sender has 20 receivers. Therefore, there are 40 paths from senders to receivers in each ISP.

Increasing the number of proxies in each network will reduce the total delay from senders to receivers and this trend is shown in Fig. 3. When the number of proxies reaches certain number, adding more proxies will not reduce the total delay. In ISP AS3257, the maximum number of proxies that can reduce the total delay is 6 for the testing case; In ISP AS1755, the maximum number of proxies that can reduce the total delay is 10. We also found that the first and second added proxy give much improvement in reducing the total delay. Adding more proxies, the delay percentage improvement (DPI) changes slowly for both ISP networks.

In addition to generating the optimal number of proxies and the delay average percentage improvement, Linear Programming formulation also finds the optimal locations of the optimal proxies. The results of linear programming will help the internet service provider in placing limited proxies at optimal locations.

V. OVERLAY DIJKSTRA'S ALGORITHM RESULTS

Running Dijkstra's algorithm on the overlay network can generate maximum benefits of reducing total delay when the number of proxies is not limited. Compared to linear programming, running Dijkstra's algorithm on overlay network is much more efficient in generating statistical results which will help us in identifying the network metrics that affect the maximum benefit of reducing total delay by placing proxies.

A. Link Weight Settings

Link Delay Weight Ratio(LDWR) of a link is defined in Section II. When the LDWR values of all links in a network are same, the default shortest weight path from a sender to a receiver is same as the shortest delay path from the sender to the receiver and adding proxies will not reduce the delay. In ISP network AS3257, we randomly set the link weights and controlled the LDWR in certain range. For each setting of the network link weights and the calculated standard deviation of LDWR, we run Dijkstra's algorithm overlay optimization 500 times. In each time, we randomly chose two senders and for each sender we randomly chose 20 receivers. Then, we calculated the average Delay Percentage Improvement(DPI) of the 500 runs. We set the link weights multiple times and for each time we repeat the same procedure to calculate the standard deviation of LDWR and the average Delay Percentage Improvement.

Fig. 4 shows that when the link delay weight ratio standard deviation increases, the *Delay Percentage Improvement*(DPI) due to placing proxies will increase. As the LDWR standard deviation increases, the link weights are more random and the delay percentage improvement will be higher. when the LDWR is greater than 1, the link weights become random and the DPI will not change too much.

B. Network Default Delay Minimum Delay Ratio

We identified a network metric, *Default Delay Minimum Delay Ratio*(DDMDR), that provides guidance on whether adding proxies will reduce the total delay from senders to receivers. As explained in Section II, DDMDR is the ratio

of the average delay of the default path(shortest weight path) and the the average delay of the shortest delay path. For the 6 ISP networks, we calculated the average delay of the default path and average delay of the shortest delay path between two nodes. From the two calculated average delays, we calculated the *Default Delay Minimum Delay Ratio*(DDMDR). For each ISP network, we ran Dijkstra's algorithm overlay optimization 500 times. In each run, we randomly chose 2 senders and randomly chose 20 receivers for each sender. We calculated the average *Delay Percentage Improvement*(DPI) of the 500 runs.

TABLE III
AVERAGE DEFAULT PATH DELAY BETWEEN 2 NODES, AVERAGE SHORTEST DELAY PATH DELAY BETWEEN 2 NODES, DEFAULT DELAY MINIMUM DELAY RATIO(DDMDR) AND AVERAGE DELAY PERCENTAGE IMPROVEMENT (DPI) OF THE 6 ISP NETWORKS.

ISP	Average Default Delay(ms)	Average Min Delay(ms)	DDMDR	DPI(%)
AS1755	14.17	12.65	1.120	11.76
AS3257	18.06	15.65	1.154	13.26
AS6461	36.54	35.11	1.041	3.67
AS3967	25.37	24.18	1.049	4.32
AS1221	16.00	15.79	1.013	1.24
AS1239	24.36	23.16	1.052	4.42

Table III shows the average default path delay between 2 nodes, average minimum delay path delay between two nodes, *Default Delay Minimum Delay Ratio*(DDMDR) and average *Delay Percentage Improvement* (DPI) of each ISP network. Fig. 5 is a plot of the relationship between network *Default Delay Minimum Delay Ratio*(DDMDR) and the average *Delay Percentage Improvement*(DPI). From the figure, we observe that *Default Delay Minimum Delay Ratio*(DDMDR) and the *Delay Percentage Improvement*(DPI) have almost linear relationship. As the DDMDR increases, adding proxies will generate higher *Delay Percentage Improvement*(DPI).

We also introduced the metric *multicast default delay minimum delay ratio*. It is defined as the ratio of total default path(shortest weight path) delay and the total minimum delay path(shortest delay path) delay from each sender to its multiple receivers. For ISP AS3257, we ran Dijkstra's algorithm overlay optimization 500 times. In each run, we randomly chose 2 senders and randomly chose 20 receivers for each sender. We calculated the *multicast default delay minimum delay ratio* and the optimized delay percentage improvement after adding proxies in each run.

Fig. 6 is the scatter plot of multicast default delay minimum delay ratio and the total delay percentage improvement after adding proxies. From the figure, we observe the trend that runs with higher default delay minimum delay ratio have higher delay percentage improvement after adding proxies. There are a few exceptions where high default delay minimum delay ratio didn't produce high delay percentage improvement after adding proxies. That's because the searched overlay paths are far away from the shortest delay paths due to link weight

setting(i.e, large weights are set for some short delay links)

C. Network Delay Percentage Improvement Histogram

When running Dijkstra’s algorithm overlay optimization, we recorded the delay percentage improvement for each selection of senders and receivers and analyzed the distribution of delay percentage improvement. Fig. 7 is the delay percentage improvement histogram of 500 runs on ISP AS3257. From the figure, we observe that most of the selections of senders and receivers have the delay percentage improvement close to the average percentage improvement of 13.3%. Some selections of senders and receivers have delay percentage improvement higher than 30% after adding proxies.

VI. CONCLUSION AND FUTURE WORK

In this paper, we studied reducing the multicast delays by placing proxies at optimal locations. We first created overlay network by creating logical links between every two nodes. Then, we created integer linear programming formulations and used CPLEX optimization solver to solve the formulations. From linear formulation results, we can find the optimal number of proxies, optimal locations of proxies and the delay improvement by placing proxies. By changing the number of proxies in linear formulation, we observed that placing proxies will significantly reduce the total delay for some networks. We also ran Dijkstra’s algorithms on 6 ISP networks to explore the maximum benefit in reducing delays by adding proxies. The Dijkstra’s algorithm results show how the link weight setting affect the delay percentage improvement when placing proxies. Finally, we identified a network metric, Default Delay Minimum Delay Ratio, which will provide guidance on whether adding proxies will greatly reduce the total delay from senders to receivers in a network.

Although linear programming can produce theoretical solutions, it may not be practical to solve the optimization problem with 10s of millions of optimizing parameters and constraints when placing proxies in large networks. A heuristic algorithm needs to be developed to improve the performance in proxy placement optimization. In this work, we focus on placing proxies within a single ISP and we need to extend the proxy placement solution to inter-domain network.

REFERENCES

- [1] B. Fortz and M. Thorup, “Internet traffic engineering by optimizing ospf weights,” in *Proc. IEEE INFOCOM*, 2000.
- [2] N. T. C. R. Keralapura and G. Iannacone, “Can isp’s take the heat from overlay networks?,” in *ACM HotNets Workshop*, 2004.
- [3] J. K. Honggang Zhang and D. Towsley, “Can an overlay compensate for a careless underlay?,” in *Proc. IEEE INFOCOM*, April 2006.
- [4] “Inferring link weights using end-to-end measurements,” in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, 2002.
- [5] A. M. K. Andreev, B. Maggs and R. Sitaraman, “Designing overlay multicast networks for streaming,” in *Proc. ACM Symposium on Parallel Algorithms and Architectures*, 2003.
- [6] C. K. S. Banerjee and B. Bhattacharjee, “Scalable application layer multicast,” in *ACM SIGCOMM*, August 2002.
- [7] S. B. S. B. S. Ganguly, A. Saxena and R. Izmailov, “Fast replication in content distribution overlays,” in *Proc. IEEE INFOCOM*, March 2005.
- [8] R. M. Neil Spring and D. Wetherall, “Measuring isp topologies with rocketfuel,” in *Proc. ACM SIGCOMM*, August 2002.

$$\text{minimize: } \sum_{s \in S} \sum_{r \in R_s} D(s, r) \quad (3)$$

or

$$\text{minimize: } \sum_{s \in S} \max_{r \in R_s} D(s, r) \quad (4)$$

subject to:

$$x_v \in \{0, 1\}, v \in V \quad (5)$$

$$x_e \in \{0, 1\}, e \in E \quad (6)$$

$$x_e \leq x_u, x_e \leq x_v, e = (u, v) \in E \quad (7)$$

$$\sum_{v \in V} x_v \leq N_V \quad (8)$$

$$\sum_{e \in E} x_e \leq N_E \quad (9)$$

$$f_r^s(u, v) \in \{0, 1\}, s \in S, r \in R_s, (u, v) \in E \quad (10)$$

$$0 \leq f_r^s(u, v) \leq x_e, s \in S, r \in R_s,$$

$$e = (u, v) \in E, u \neq s, v \neq r \quad (11)$$

$$0 \leq f_r^s(s, v) \leq x_v, s \in S, r \in R_s \quad (12)$$

$$0 \leq f_r^s(v, r) \leq x_v, s \in S, r \in R_s \quad (13)$$

$$\sum_{v \in V \setminus \{s\}} f_r^s(s, v) = 1, s \in S, r \in R_s \quad (14)$$

$$\sum_{v \in V \setminus \{s\}} f_r^s(v, s) = 0, s \in S, r \in R_s \quad (15)$$

$$\sum_{v \in V \setminus \{r\}} f_r^s(v, r) = 1, s \in S, r \in R_s \quad (16)$$

$$\sum_{v \in V \setminus \{r\}} f_r^s(r, v) = 0, s \in S, r \in R_s \quad (17)$$

$$\sum_{u \in V} f_r^s(u, v) = \sum_{w \in V} f_r^s(v, w), s \in S, r \in R_s,$$

$$v \in V \setminus \{s, r\} \quad (18)$$

$$D(s, r) = \sum_{e \in E} f_r^s(e) l(e), s \in S, r \in R_s \quad (19)$$

Fig. 2. Integer linear programming formulation for the optimal proxy placement

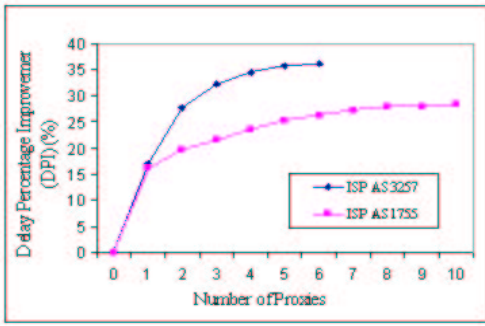


Fig. 3. Proxy numbers and Delay Percentage Improvement(DPI) in ISP AS3257 and ISP AS1755

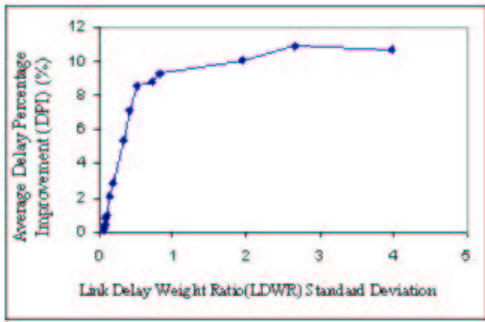


Fig. 4. Link Delay Weight Ratio(LDWR) standard deviation and Delay Percentage Improvement(DPI) in ISP AS3257.

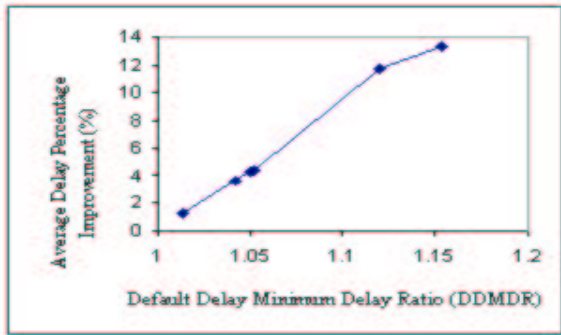


Fig. 5. Relationship between *Default Delay Minimum Delay Ratio*(DDMDR) and the average delay percentage improvement. The 6 points in the plot are from the results of 6 ISPs.

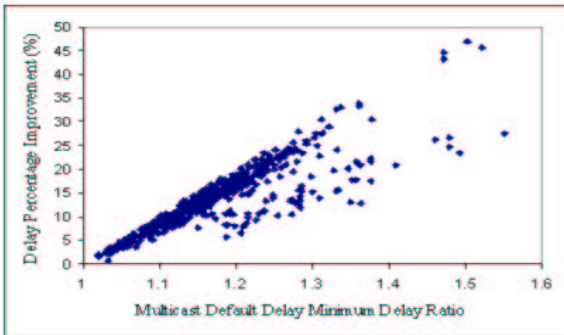


Fig. 6. Scatter plot of multicast default delay minimum delay ratio and the total delay percentage improvement after adding proxies. proxies in ISP AS3257.

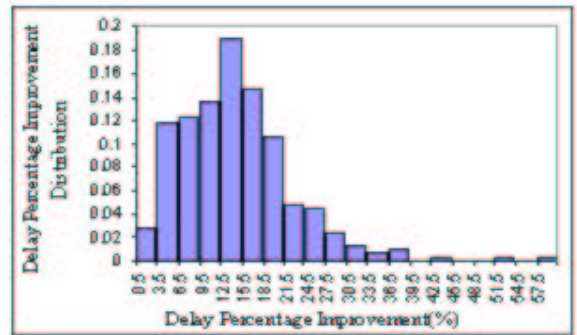


Fig. 7. ISP AS3257 delay percentage improvement distribution histogram.