Modified Widest Disjoint Paths Algorithm for Multipath Routing

Shangming Zhu[1], Zhili Zhang[2], and Xinhua Zhuang[3]

[1] Department of Computer Science, East China University of Science and Technology, Shanghai, 200237,China
[2] Department of Computer Science, University of Minnesota,Minneapolis, MN 55455, USA
[3] Department of Computer Science, University of Missouri-Columbia, Columbia, MO65211,USA
{zhusm@ecust.edu.cn, zhzhang@cs.umn.edu, zhuangx@missouri.edu}

Abstract. Widest Disjoint Paths (*WDP*) algorithm is a promising multipath routing algorithm aimed at selecting good paths for routing a flow between a source-destination pair, where their bottleneck links are mutually disjoint. Nevertheless, the complexity of *WDP* algorithm is relatively high due to the fact that the good path selection process considers all available paths. To reduce its complexity, This paper proposes a modified *WDP* algorithm, which uses only a subset of available paths based on shortest widest paths, thereby limiting the number of candidate paths considered. As a result, the number of iterations in the good path selection process is significantly reduced. Performance analysis shows the modified scheme is more efficient than the original algorithm in a large network. Simulation results demonstrate that, in comparison with the original *WDP* algorithm, the modified *WDP* algorithm leads to lower latency and faster packets transferring process as the number of available paths increases.

Keywords: Widest disjoint paths, multipath routing, candidate path, width of path set

1 Introduction

Multipath routing is designed to optimize the operational performance of a network by routing simultaneously along multiple paths on demand. It improves QoS (Quality of Service) of a network in using multiple "good" paths rather a single "best" path in routing. Several multipath routing algorithms have been proposed for balancing the load across a network [1-3]. *WDP (Widest Disjoint Paths)* [1,2] and *LDM (Load Distribution over Multipath)* [4] schemes make routing decisions at the flow level. The *WDP* algorithm is based on two concepts: path width and path distance. Path width is used to detect bottlenecks in the network and avoid them if possible. Path distance is indirectly dependent on the utilization ratio of each constituent link of the path. The *LDM* algorithm attempts to find a minimal set of good paths based on two criteria: (a) the metric hop-count associated with each path kept as low as possible and (b) link utilization maintained inferior to a certain parameter. Schemes like

ECMP (Equal Cost MultiPath) [5], *MPLS-OMP (Optimized MultiPath)* [6] and *MATE (MPLS Adaptive Traffic Engineering)* [7] perform packet level forwarding decisions. ECMP splits the traffic equally among multiple equal cost paths. But these paths are determined statically that may not reflect the congestion status of the network. It is always desirable to apportion the traffic according to the quality of each individual path. *MPLS-OMP* uses updates to gather link load information, selects a set of optimal paths and then distributes traffic among them. *MATE* does constant monitoring of links by using probe packets to measure link quality such as packet delay and packet loss.

The simulation results reported in [1] and [2] show that *WDP* is a promising multipath routing algorithm that can provide higher throughput and is capable of adapting to the changing network conditions at different time scales comparing to other schemes. In this paper, we attempt to improve the *WDP* algorithm in terms of computational efficiency.

2 Widest Disjoint Path Problem

The *WDP* algorithm is aimed at selecting good paths for routing a flow between a source-destination pair so that their bottleneck links are mutually disjoint. There are three tasks in the multipath routing: (a) dynamically select the candidate paths, (b) determine the good paths from the candidate paths and (c) proportion traffic among the good paths. This section lays out the basic assumptions for path selection.

In *WDP* algorithm, source routing is used and the network topology information is assumed available to all source nodes. One or multiple explicit-routed paths or label switched paths are set up statically between each source-destination pair using, e.g., MPLS. A flow is blocked when routed to a path whose bottleneck link has no bandwidth left.

Consider a network topology with N nodes and L links. Let s be a source node and d be a destination pair, and let r denote a path in the network, i.e., r is a set of links from a source to a destination. Let the maximum capacity of link l be $\hat{C}_l > 0$, which is fixed and known in advance.

Let $\hat{R}$ denote the set of all available paths for routing flows between the source and destination node. The set of good paths R is a subset of $\hat{R}$ selected by the good path selection algorithm for multipath routing. The *WDP* algorithm aims at computing a path set R for routing a flow between a source-destination pair such that all paths in R are mutually disjoint with regard to (w.r.t.) the bottleneck links.

3 Modified WDP Algorithm

The original *WDP* algorithm [1,2] compares all available paths against the width of the set of existing good paths for good path selection. To reduce its computation complexity, we modify the *WDP* algorithm in the following two aspects: (a) we use only a subset of available paths based on shortest widest paths rather than all available paths as the candidate paths; (b) we limit the number of candidate paths, hence, the number of iterations in good path computation.

3.1 Width of Path Set

To determine whether a candidate path is a good one and whether to include it in the existing good path set, the width *W* of a path set *R* is calculated as follows.

First, the difference $C_l = \hat{C}_l - v_l$ is defined as the average residual bandwidth on link *l*, where v_l is the average load on the link. The width and distance of a path *r* are defined as $w_r = \min_{l \in r} C_l$ and $d_r = \sum_{l \in r} 1/C_l$, respectively. Disjoint paths in a path set are recursively determined *w.r.t.* the bottleneck links, and the sum of their widths is used to calculate the width of the path set.

Initially, *W* is set to 0. The width *w* of each disjoint path *r* is added to the width *W* of path set *R* according to their widths and distances. In each iteration, only the shortest widest paths are considered. A subset R^* of paths with the widest width w^* is identified. There may be more than one path with the widest width w^*, from these widest paths, a path r^* with the shortest distance d^* is selected and the width w^* of shortest widest path r^* is added to the total width *W* by $W = W + w^*$.

To ensure the disjoint paths are only used to compute the width of set *R*, the residual capacities of all the links along each shortest widest path r^* is reduced by an amount w^* by $C_l = C_l - w^*$ and the path r^* is removed from the set *R*. The above iteration process is repeated until the set *R* becomes empty. The resulting *W* is considered to be the width of *R*. The narrowest path, i.e., the last path removed from the set *R*, is referred to as *Narrowest(R)*.

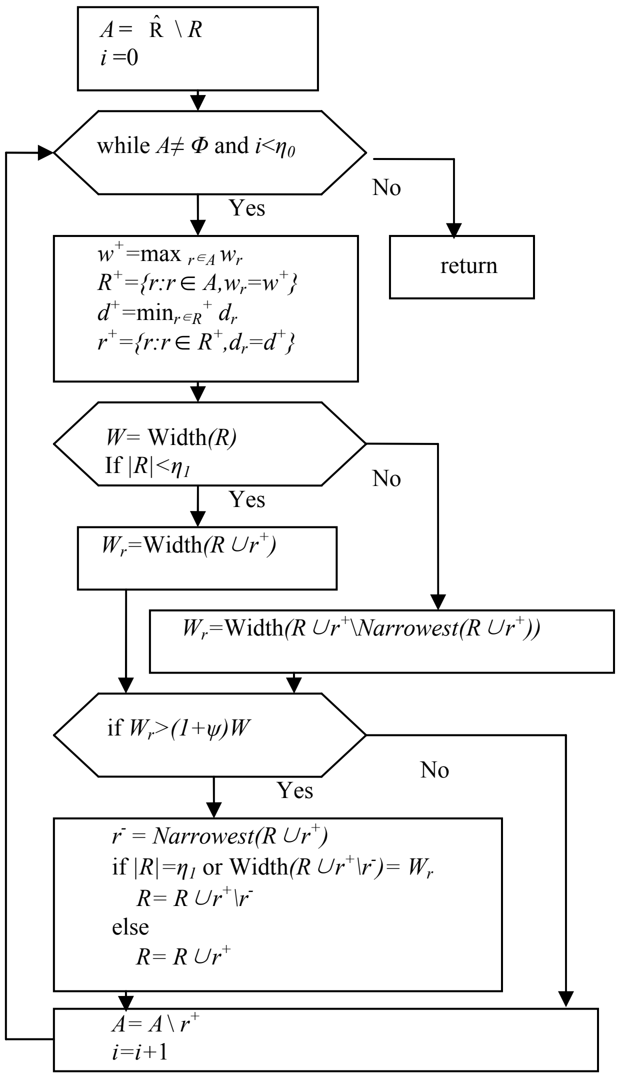
Based on the defined width of a path set, a new candidate path is added only if its inclusion increases the width of the set of good paths, and an existing good path is removed if its exclusion does not decrease the total width. When the number of good paths reaches a specified limit, a good path is replaced with another path if this change increases the width.

3.2 Modified Good Paths Selection

The success of *WDP* relies on how to efficiently select good paths. We observe that there is a significant overhead and high complexity in the original *WDP* algorithm because all available paths are considered. As the number of available paths increases, it becomes increasingly computational expensive to compute good paths.

To improve the efficiency of the original *WDP* algorithm and shorten the time for computing good paths, we use only a subset of available paths based on shortest widest paths, thereby reducing the number of iterations in good path computation. Our proposed good path selection algorithm is shown in Fig. 1.

First we exclude *R* from $\hat{R}$ to initiate candidate path set *A*, namely, $A = \hat{R} \setminus R$ (excluding *R* from $\hat{R}$). Different from the original *WDP* algorithm, we set up a parameter η_0 as the maximum of iterations in this procedure to reduce the number of candidate paths and limit the number of iterations.



$A=\hat{R}\setminus R$
$i=0$
while $A\neq\Phi$ and $i<\eta_0$
Yes
No
$w^{+}=\max_{r\in A} w_r$
$R^{+}=\{r:r\in A, w_r=w^{+}\}$
$d^{+}=\min_{r\in R^{+}} d_r$
$r^{+}=\{r:r\in R^{+}, d_r=d^{+}\}$
return
$W=$ Width(R)
If $|R|<\eta_1$
Yes
No
$W_r=$Width$(R\cup r^{+})$
$W_r=$Width$(R\cup r^{+}\setminus Narrowest(R\cup r^{+}))$
if $W_r>(1+\psi)W$
Yes
No
$r^{-}=Narrowest(R\cup r^{+})$
if $|R|=\eta_1$ or Width$(R\cup r^{+}\setminus r^{-})=W_r$
$R=R\cup r^{+}\setminus r^{-}$
else
$R=R\cup r^{+}$
$A=A\setminus r^{+}$
$i=i+1$

Fig. 1 Modified WDP algorithm

In each iteration, rather than consider all available paths, our proposed approach considers only a subset of available paths based on shortest widest paths to compute good paths, namely, it only selects a candidate path r^{+} with the widest width w^{+} and shortest distance d^{+} to compute good paths. It is possible that many candidate paths may have the same widest width w^{+}. Among such paths, the path r^{+} with the shortest distance d^{+} is chosen.

Second, the procedure of adding a new path to good path set and removing an existing good path from good path set or replacing it with another path is similar to the original *WDP* algorithm.

The width of a set of paths before inclusion r^+ is defined as $W=Width(R)$. The function $Width()$ that computes width of a path set has been discussed in III.*A*. Inclusion of a candidate path r^+ is determined based on the number of good path set R and the width increase due to path r^+.

Depending on the number of multiple paths required by an application, a parameter η_1 is set up as the maximum number of good paths in *WDP* algorithm. If the number of good paths in the current good path set R is below the specified limit η_1, the resulting width W_r is the width after including r^+ among R, i.e., $W_r = Width(R \cup r^+)$; Otherwise, it is the width after including r^+ but excluding the narrowest path among $R \cup r^+$, i.e., $W_r = Width(R \cup r^+ \setminus Narrowest(R \cup r^+))$.

This width W_r is compared to the current width W of the good path set. A candidate path is made a good one if its inclusion in set R increases the width by a fraction ψ. Here $\psi > 0$ is a configurable parameter to ensure that each addition improves the width by a significant amount.

If the width W_r is more than $(1+\psi)W$, r^+ will be added to the good path set. Before adding r^+, the path set $R \cup r^+$ should be pruned, in other words, its narrowest path may be removed if needed using the following criterion Let r^- be the narrowest path in the set $R \cup r^+$. The path r^- is replaced with r^+ if either the number of good paths has already reached the limit η_1 or the path r^- does not contribute to the width of the set. Otherwise, the path r^+ is simply added to the set of good paths. If the width W_r is not more than $(1+\psi)W$, no new path is added.

After each computation of candidate path set, the candidate path r^+ is removed from A, and the number of iteration i is modified. The iteration is repeated until A is empty or its iteration reaches the maximum η_0.

4 Performance Analyses

4.1 Analysis of Computational Efficiency

It is obvious that the scheme described above always converges for all values of η_0 and η_1. The original *WDP* algorithm takes $O(n^3)$ iterations, where n is the number of available paths between the pair of ingress-egress routers [3]. In contrast, the complexity of our proposed algorithm is $O(\eta_0\eta_1 n)$, as selecting good paths from a set of n paths depends on η_0 candidate paths and η_1 good paths. In general, η_0 and η_1 are far smaller than n, especially in a large network. Hence the modified scheme is more efficient than the original algorithm in terms of computational complexity.

4.2 Simulation Results

In this section the performance of the modified *WDP* algorithm is studied through simulation experiments using NS2. The simulation environment we use is shown in Fig. 2. There are two types of links: solid and dotted. To simplify the simulation, all solid links have the same capacity with 2Mbps of bandwidth and all the dotted links have the same capacity with 3Mbps of bandwidth. We also ignore the propagation delay of each links. There are 18 nodes and 30 links in the topology. To analyze the performance of our modified *WDP* algorithm, node *1* is chosen as the source node and node *18* is as the destination node. The default values for the *WDP* parameters are ψ=0.2, η_0=3, η_1=2.

We use the *end-to-end packet delivery latency* (in short, latency) as a metric to evaluate the impact f the computational overheads of the modified *WDP* and the original *WDP* algorithm on the efficacy of multi-path routing. To measure the end-to-end packet delivery latency, the packet sending rate at the source node is set to 4.5Mbps.

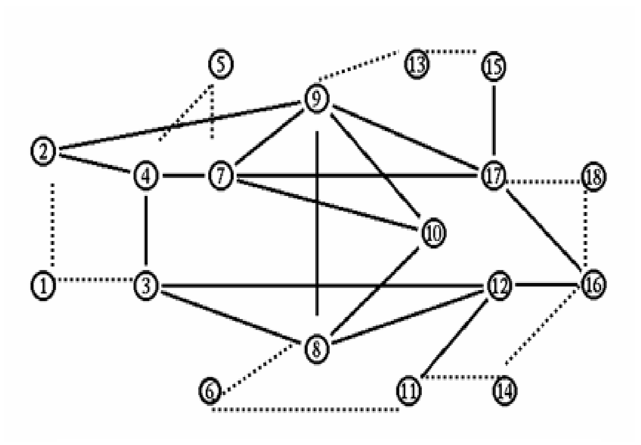


Fig. 2 Network model for the simulation

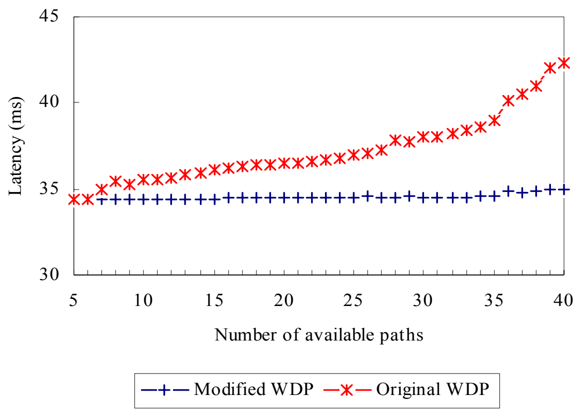
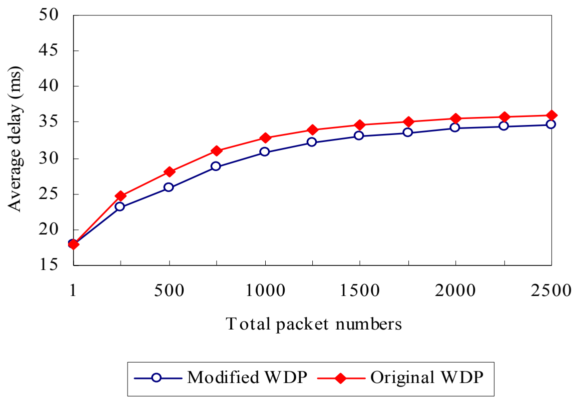
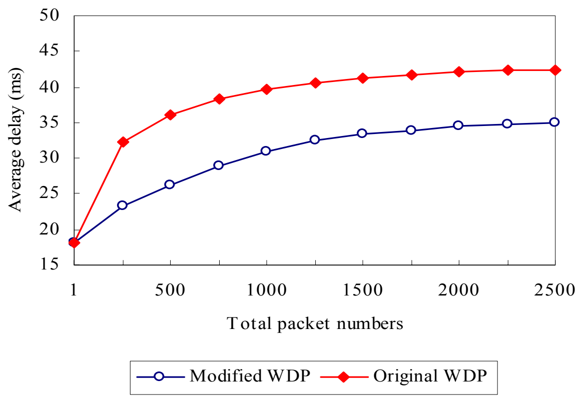


Fig. 3 Latency with variable numbers of available paths



(a) *n*=10



(b) *n*=40

Fig. 4 Comparison of average one way delay

Fig. 3 shows the latency variation as the number of available paths *n* is varied. It is seen that as the number of available paths increases, the modified *WDP* performs better than the original *WDP* algorithm. When the number of available paths is small ($n \le 6$), there is not much difference of latency between the modified *WDP* and the original *WDP* algorithm; As the number of available paths increases, latency of the modified *WDP* varies slightly and is much lower than that of the original *WDP* algorithm.

Fig. 4 shows the comparison of the average one way delay using varying numbers of available paths. In Fig. 4(a) the number of available paths *n* is set to 10; it is seen that the modified *WDP* is a little faster than the original *WDP* algorithm in the packets transferring process. In Fig. 4(b) *n* is set to 40; it is seen that the modified *WDP* is much faster than the original *WDP* algorithm in the packets transferring process because of its better computational efficiency.

5 Conclusions

In this paper we propose an improvement to the *Widest Disjoint Paths (WDP)* routing algorithm which performs good paths selection based on the computation of the width of the disjoint paths with regard to bottleneck links. To reduce the complexity of this algorithm and shorten the time for computing good paths, we develop a modified *WDP* algorithm by limiting the number of candidate paths and thereby reducing the number of iterations needed for good path computation. Our proposed approach only uses a subset of available paths based on shortest widest paths rather than all available paths to compute the good paths. Performance analysis shows the modified scheme is more efficient than the original algorithm in a large network. Simulation results demonstrate that the modified *WDP* algorithm leads to lower latency and faster packets transferring process as the number of available paths increases in comparison with the original *WDP* algorithm.

Acknowledgments. This work is supported in part by NIH, NSF, and the National Science Council (NSC) of Republic of China under grant number NIH DHHS 1 R01 DC04340-01A2, NSF EIA 9911095, and NSC-94-2213-E-006-068 respectively. This work was also supported by China Scholarship Council (CSC).

References

1. Srihari Nelakuditi, Zhi-Li Zhang, David H.C. Du: On Selection of Candidate Paths for Proportional Routing. Computer Networks, Vol. 44 (2004) 79-102
2. Srihari Nelakuditi, Zhi-Li Zhang: On Selection of Paths for Multipath Routing. In: Proceedings of IWQoS'2001:9th International Workshop, Karlsruhe, Germany, Vol. 2092 (2001) 170-184
3. Kyeongja Lee, Armand Toguyeni, Aurelien Noce, Ahmed Rahmani: Comparison of Multipath Algorithms for Load Balancing in a MPLS Network. Lecture Notes in Computer Science, Vol. 3391. Springer-Verlag, Berlin Heidelberg New York (2005) 463–470
4. Jeonghwa Song, Saerin Kim, Meejeong Lee: Dynamic Load Distribution in MPLS Networks. Lecture Notes in Computer Science, Vol. 2662. Springer-Verlag, Berlin Heidelberg New York (2003) 989–999
5. J. Moy: OSPF Version 2, RFC 2328. Internet Engineering Task Force. http://www.ietf.org/rfc/rfc2328.txt (1998)
6. C. Villamizar: MPLS Optimized Multipath (MPLS-OMP). Work in Progress Internet-Draft. <draft-villamizar-mpls-omp-01> (1999)
7. A. Elwalid, C. Jin, S. Low, I. Widjaja: MATE: MPLS Adaptive Traffic Engineering. In: Proceedings of INFOCOM'2001, Alaska (2001)