Self-optimizing Mechanism for Prediction-based Decentralized Routing

Abutaleb Abdelmohdi Turky, Florian Liers and Andreas Mitschele-Thiel

Integrated Communication Systems Group
Ilmenau University of Technology
98693 Ilmenau, Germany
{abutaleb-abdelmohdi.turky,florian.liers,mitsch}@tu-ilmenau.de

Abstract. In this paper, we introduce an adaptive traffic prediction approach for self-optimizing the performance of a Prediction-based Decentralized Routing (PDR) algorithm. The PDR algorithm is based on the Ant Colony Optimization (ACO) meta-heuristics in order to compute the routes. In this approach, an ant uses a combination of the link state information and the predicted available bandwidth instead of the ant's trip time to determine the amount of deposited pheromone. A Feed Forward Neural Network (FFNN) is used to build adaptive traffic predictors which capture the actual traffic behavior. Our contribution is a new self-optimizing mechanism which is able to locally adapt the prediction validity period depending on the prediction accuracy in order to efficiently predict the link traffic. We study three performance parameters: the rejection ratio, the percentage of accepted bandwidth and the effect of prediction use. In general, our new algorithm reduces the rejection ratio of requests, achieves higher throughput when compared to the AntNet and Trail Blazer algorithms.

Keywords: Traffic engineering, self-organization, ant-based routing, quality of service, artificial neural network.

1 Introduction

The rapid growth of the Internet forces the Internet Service Providers (ISPs) to search for a new technology which has the capability to maximize the network utilization. They hope to increase their revenues by deploying the concept of service differentiation and offering higher quality services. To support such capabilities, the conventional IP technologies should use the methodology of Traffic Engineering (TE).

TE is defined as that aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks [1]. TE aims to cover different optimization issues that are related to the network performance such as providing the requested Quality of Service (QoS), minimizing the total delay and maximizing the network throughput, improving the network resources utilization by optimally distributing the traffic over the network topology and quick recovery in from failures.

QSHINE 2010, November 17-19, Houston, United States

DOI 10.4108/eai.17-11-2010.2245704

There are different sets of routing classifications that depend on different point of views. Routing algorithms can be classified as static or dynamic. With static routing algorithms such as the Minimum Hop Algorithm (MHA), the administrator computes off-line all possible routes using statistic information and updates the routing table accordingly. Dynamic routing algorithms use information about the current state to compute the requested route on demand. All routing decisions in the dynamic routing approach are performed online to reflect changes of network states. This paper focuses on dynamic routing approaches. Most routing protocols are dynamic such as the Open Shortest Path First (OSPF) protocol [2], the Routing Information Protocol (RIP) and Multi-Protocol Label Switching (MPLS) [3].

The efficiency of TE schemes mainly depends on the routing optimization. Most of the dynamic routing algorithms use the available Bandwidth (BW) to choose the paths between the source and destination pairs. The provided QoS depends on accurate measurements of the available BW. Due to the varying nature of the available BW, updating the link state with the current measured BW is not an efficient approach to represent the link utilization. Therefore, new approaches perform an estimation of the link utilization in the future using the actual traffic profile. The proposed routing mechanism should optimize the network utilization, improve the network survivability and reduce future interference between requests.

Routing algorithms can be classified into centralized or decentralized also. In the centralized routing approach; the source router has all necessary information to compute the routes. In the decentralized routing approach, the routing decisions are taken based on the local state information only and by each network node individually. Most decentralized routing approaches use ant-based mechanisms [4]. These algorithms are based on Ant Colony Optimization (ACO) meta-heuristics. Ant systems represent a self-organizing approach which applies the principle of indirect communication between agents by handling the changes to their environment [5]. Ant routing algorithms are inspired from real ants' behaviors which have the ability of discovering the shortest path to a food source and their nest without any knowledge of geometry but with a keen sense of smell. By applying reinforcement learning techniques, ant routing algorithms can find the optimal or a close-to-optimal path between the source and destination through a positive feedback mechanism.

In this paper, we introduce an adaptive traffic prediction mechanism for self-optimizing the performance of Prediction-based Decentralized Routing (PDR) algorithm which is based on the ACO meta-heuristics to compute the routes. The idea of the PDR algorithm is to use the combination of the link state information and the predicted available BW instead of the ant’s trip time to determine the amount of pheromone to deposit. We build our traffic predictor using the FFNN which has proved its accuracy to capture the actual traffic behavior. The proposed predictor uses a new adaptive mechanism to be able to locally adapt the prediction validity period depending on the prediction accuracy in order to efficiently predict the link traffics. Depending on the predicted load value, the algorithm computes the available BW and combines it with the current available BW in the link weight formula that is used to select the optimal path. The remainder of this paper is organized as follows: Section 2 gives an overview of the related work. Section 3 introduces the design details of our approach. Section 4 demonstrates the comparative results and discusses the performance evaluation. Future work and conclusions are presented in section 5.

2 Related Work

The traditionally used algorithm within the MPLS domain is the Shortest Path First (SPF) algorithm proposed by E. Dijkstra [6] to solve the shortest path problem. The basic idea of the SPF algorithm is used in many routing protocols such as the OSPF protocol. Dijkstra's algorithm depends on the fact that states on subsections of shortest paths are also shortest paths. It does not just compute the shortest path to a specific destination, but computes the shortest paths to all possible destinations in the network. Roch A. Guerin [7] has introduced a modification to the shortest path algorithm, called Widest Shortest Path (WSP), which is based on the computation of the shortest paths in the first stage with the extension that if there is more than one shortest path, it chooses the one with the maximum BW. This work is introduced to provide extensions to the OSPF protocol in order to support QoS routing in IP-based networks.

Since all previous algorithms aim to select the best paths without considering future path requests, the related performance does not achieve the best results in maximizing the network utilization or the acceptance rate of requests. The Minimum Interference Routing Algorithm (MIRA) is an example of an advanced routing algorithm [8]. The idea of MIRA is avoiding the routing over links that may interfere with another path request in the future. The definition of interference in MIRA depends on computing the maximum flow (max-flow) value between a given ingress and egress pair. The minimum interference path is the path that maximizes the minimum max-flow between all other ingress-egress pairs.

A.B. Bagula [9] introduced a Least Interference Optimization Algorithm (LIOA) which reduces the interference among competing flows by balancing the number and quantity of flows that are carried by a link to achieve efficient routing of BW guaranteed requests. In general, simulation studies demonstrate that LIOA outperforms many routing algorithm such as MHA, and MIRA algorithms. The comparative study of theses algorithms depends on different performance metrics including the rejection ratio of requests and the successful re-routing of requests upon single link failures.

E. Einhorn and A. Mitschele-Thiel introduced the Reinforcement Learning for Traffic-Engineering (RLTE) algorithm [10]. This work presents a novel distributed and self-organized QoS routing approach that is based on reinforcement learning. We have introduced a primary version of the Predicting of Future Load-based Routing (PFLR) algorithm in [11]. The PFLR algorithm uses the predictions of the future load in order to solve the routing problem. The performance of the PFLR algorithm has been compared to earlier routing approaches like the WSP and CSPF algorithms. The primary version of the PFLR algorithm has reduced the rejection ratio of requests and achieves a higher throughput.

We have proposed an enhanced version of PFLR in [12]. PFLR v.2 combines the predicted future load and current residual BW of each link in a formula to represent the Reciprocal of available BW (*RBW*) and then updates the link weight formula with *RBW*. The PFLRv.2 performance has been compared with current routing approaches like DORA and LIOA algorithms and demonstrated the efficiency of PFLRv.2 by testing three performance criteria: the rejection ratio of requests, the percentage of

accepted BW and the rejection ratio of rerouted requests within the link failure scenario.

AntNet [13] is an ACO algorithm for distributed and adaptive best-effort routing in IP networks. AntNet is considered the first algorithm that is inspired by ant colony behavior to solve the routing problem. The behavior of AntNet depends on the mobile agents or the ants' framework. During the forward phase, each ant constructs a path by taking a sequence of decisions based on a stochastic policy parameterized by local pheromones and heuristic information. Once it arrives at the destination, the backward phase starts. The backward ants retrace the route, followed by their forward ant, and update the local routing information with an amount of pheromone in all the intermediate nodes.

Yun and Zincir [14] introduced an adaptive routing algorithm based on the AntNet algorithm. This approach uses a new structure of the routing table to overcome the problem of unrealistic requirements for global information of the original AntNet algorithm. The Trail Blazer (TB) routing algorithm minimizes the network congestion through local decisions, based on latency measurements collected by scout packets [15]. TB is meant to be an extension of existing link-state protocols such as OSPF, which provides shortest-path information to initialize the probability table. Therefore, TB does not require a learning period to discover the network topology. TB is also simpler than the AntNet algorithm.

We have proposed a first version of our TE algorithm named Prediction-based Decentralized Routing (PDR) algorithm [16] that can efficiently enhance the routing performance. This algorithm is a member of a class of traffic-aware routing algorithms based on the behavior of ants. We have compared the performance of the PDR algorithm with WSP and SPF algorithms under two different network load scenarios and have shown that the PDR algorithm performs considerably better.

3 Prediction-based Decentralized Routing

This section provides a detailed description of our improved version for the Prediction-based Decentralized Routing algorithm (PDRv.2). Figure 1 outlines the operation of PDRv.2. In the algorithm, ants are distributed through the network to discover the best paths. The ants use a combination of the link state information and the predicted link load instead of the ant's trip time to determine the amount of pheromone to deposit. This is simpler and requires less control parameters. After selecting the best path, the routing algorithm forwards the packets through the network and updates the reserved BW of each link that belongs to the best path between the source and the destination.

The idea behind the design of PDR is to consider the future link load to enhance the performance of Ant-based routing algorithms. Therefore, we propose a traffic predictor that able to accurately predict the traffic behavior. ANN offers prediction capability with different types of network traffic and has the ability to learn and adapt dynamically. Experimental results show that, ANN can accurately estimate a complicated network traffic pattern efficiently [17].

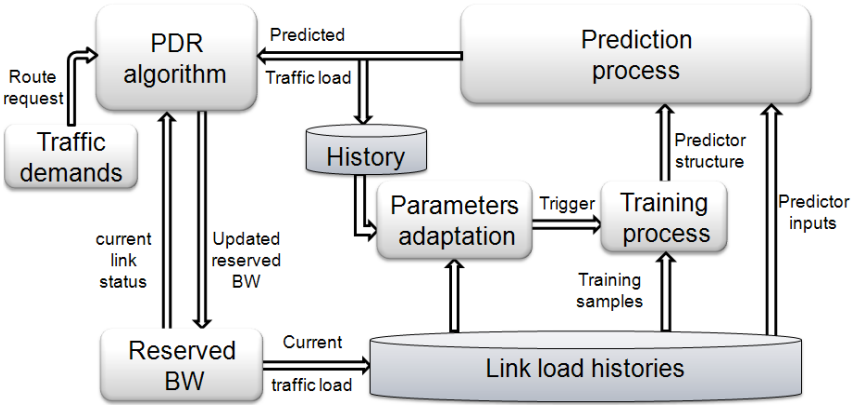

Fig. 1. Prediction-based Decentralized Routing (PDRv.2) algorithm.

The proposed predictor has two different processes: the training, and the prediction process. In the training process, the internal structure of FFNN is constructed by a training based on traffic samples of link histories. During the prediction processes, the future link load on every link is estimated after a specified period of time or a specified prediction interval, which is named Window Size (*WS*). This means that, each link has a predictor which is placed on one of the directly connected nodes. Each predictor works on its link history and has its own parameter values. In other words, the predictions are made decentralized to achieve a fast prediction and to conquer the complexity of prediction.

The new proposed feature of the PDRv.2 algorithm is the parameter adaptation process. The Prediction Validity Period (*PVP*) parameter is adapted and self-optimized depending on the prediction accuracy. The *PVP* parameter represents the duration of a period for which the prediction is valid with a high degree of confidence. With the help of this feature, the training of each predictor is triggered independently of each other.

3.1 Training Process

The structure of the used FFNN is shown in Figure 2. It consists of three layers: The input layer contains three neurons; the hidden layer contains fifteen neurons and only one neuron in the output. The Levenberg-Marquardt [18] training algorithm is used because it is the fastest and most accurate one in our case. We have tested different FFNN design and different values of training period size to achieve an efficient predictor. In contrast to the training process in the previous version of PDR algorithm that is event-based, the training process in PDRv.2 algorithm is time-based.

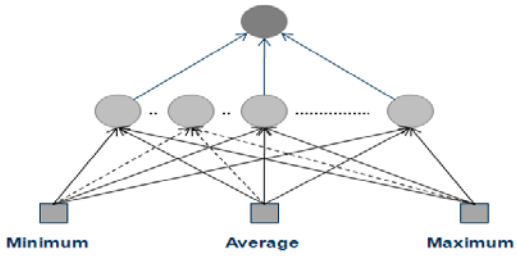

Fig. 2. Feed forward neural network architecture.

In the event-based approach, if a new path is requested in the network, a new event is generated. During the previous version of PDR, a history of the last thousand events (plus WS) of link traffic values is used for training purpose. However in PDRv.2 algorithm, a history of the last hundred time units of link traffic values is used for training purpose. One training pattern contains the minimum, maximum and average of traffic during a time unit. This pattern is formed in a row as input values and one expected output value. The expected output value is a history value WS time after the input values. By shifting, one hundred of training patterns are generated. In the previous version of PDR, the training process is triggered every one hundred traffic samples. In the new version, the training process is triggered every PVP period which is adapted depending on the prediction accuracy.

3.2 Prediction Process

In the prediction process, the minimum, maximum and average of the traffic during the last time unit are used as input for the FFNN which predicts a value for the link load after a WS period. The prediction process is triggered every WS period. An analysis study is done to select the best value of WS. In other words, the prediction happens every WS period and the predictor structure is not changed until the PVP period has elapsed.

3.3 Parameter Adaptation Process

In the PDRv.2 algorithm, we propose to use a new adaptive feature called parameter adaptation process. The main objective for this process is to give the predictor the ability to optimize the PVP parameter. A PVP parameter contains multiple WS periods to represent how many times the prediction is done.

$$PVP = WS \times PN, \tag{1}$$

Since PN is the Prediction Numbers.

The parameter adaptation process depends on the predictions accuracy that is calculated by comparing the actual and predicted traffic loads. Therefore, archiving processes are required to archive the actual and predicted traffic loads. The prediction accuracy can be represented by the prediction error and there are different error representation methods. In this paper, we use the Root Mean Square Error (*RMSE*) to represent the prediction accuracy. If *AL* is the Actual traffic Load and *PL* is the Predicted traffic Load, then the *RMSE* value is:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(AL_i - PL_i)^2}{N}} \quad (2)$$

During this process, the *PN* parameter is adjusted depending on the comparison between the *RMSE* and Error Threshold (*ETh*) parameters. Then, the *PVP* parameter is updated.

3.4 PDRv.2 Algorithm

The PDR algorithm is built on the principles of the TB routing framework. In the TB design, each router has two tables: a link probability table *Pt* and an average transmission delay table *avg*. *Pt* contains *m* rows, one for each destination node. Each row has *K* entries, one for each outgoing link of the router. The entry *pt[d,i]* is the probability of sending a packet to destination *d* on the outgoing link *i*. The table *avg* has *m* entries, one for each destination node. The entry *avg(d)* is the average transmission delay from the current node to the destination *d*, which is computed from the last *M* scout packets that arrived from *d*. The scout packet is sent from the source to the destination to explore the network. At every intermediate node, the scout packet selects the outgoing link randomly. When scout packets find their destination, they return to their source on the same path they have arrived on and update their accumulated latency *td* in every intermediate node by $td = td + t(i)$,where *t(i)* is the current latency of the outgoing link *i*. Then, the scout packets use the accumulated latency *td* to update the *pt* table as follows:

$$f(td) = max(min(avg(d)/td, 10), 0.1) \quad (3)$$

$$\Delta p = \delta \times f(td) \quad (4)$$

$$pt[d,i] = (pt[d,i] + \Delta p) / (1 + \Delta p) \quad (5)$$

$$pt[d,j] j \neq i = (pt[d,j]) / (1 + \Delta p) \quad (6)$$

The average latency *avg(td)* is used to scale the positive reinforcement value of the scout packet. A larger value of *f(td)* indicates a better (shorter) path. *f(td)* is limited to the range [0.1,10] to prevent wide fluctuations in Δp, which is the reinforcement value of *pt[d,j]*. δ defines the learning rate of the algorithm. All entries in *Pt* table of the same destination *d* are scaled by $1+\Delta p$ to ensure that their sum remains 1.

In our approach, an ant uses a combination of the link state information and the predicted link load instead of the ant's trip time to determine the amount of pheromone to deposit, so that it has a simpler process and less control parameters. The current latency $t(i)$ of an outgoing link i in the TB algorithm is replaced by the Link Weight formula $LW(i)$. $LW(i)$ represents a combination of PFLR and LIOA to reduce the interference among competing flows by balancing the number and required BW of flows carried by a link to achieve efficient routing.

The LIOA algorithm represents a cost metric which balances the number and the intensity of the flows offered to the routes. In the LIOA design, $LW(i) = I^{lc} / (Available\ BW)^{(1-lc)}$,whereas I is the number of flows carried on the link and lc is the least interference control parameter which represents a trade-off between the number and the magnitude of the flows traversing a link. On the other hand, the PFLR algorithm proposes to incorporate the Predicted Available BW ($PABW$) in the link weight formula to optimize the performance of routing. Therefore, we propose to use the $LW(i)$ formula as follows:

$$LW(i) = I^{lc} \left(\frac{(1-\alpha)}{(Available\ BW)^{(1-lc)}} + \frac{\alpha}{(PABW)^{(1-lc)}} \right) \quad (7)$$

The $LW(i)$ formula is controlled by a parameter called α ,which represents the prediction weight. A low α reduces the influence of the predicted value on the BW. A high value of α increases the influence and suppresses the current value of the available BW.

PDRv.2 Algorithm:

1) Repeat the following step until the time of the PVP has elapsed.
   a) At regular intervals of WS, predict the available BW in all links in the network after a specified WS.
   b) At regular intervals of N, each node generates and sends an ant to a destination.
   c) When a node receives an ant:
      i. It will forward the ant and selects the next link for the ant´s route randomly.
      ii. The ant never selects an outgoing link that leads to a node that has been visited earlier in its path (a loop). If there is no such outgoing link, the ant will die.
   d) When the current node is the destination, then, the ant will return to the source on the same path on which it has arrived.
   e) At each intermediate node :
      i. Compute $LW(i)$ of the outgoing link i on every link in the backward path using Equation (7).

ii. Compute *td*, *td=td+ LW(i).*

iii. Update the *pt* and *avg* tables using Equations (3), (4), (5) and (6).

2) Call the parameters adaptation procedure to adapt the *PVP* parameter.

3) Train the predictor on the link load histories.

4) Go to step 1.

5) On the other hand, when a node receives a data packet, which needs to be forwarded, data packets will be routed according to the probabilities in the pt table.

The parameter adaptation procedure consists of three steps. The first step is the computation of *RMSE* using Equation (2). In the second step, the *PN* parameter is adjusted depending on the comparison between the *RMSE* and Error Threshold (*ETh*) parameters. For example, if the *RMSE* value is equal or less than the *ETh* value, this means that the prediction accuracy is very good and the number of predictions should be increased by two. The last procedure step is calculating the new value of the *PVP* parameter using Equation (1).

Parameters adaptation procedure:

---

1) Compute the *RMSE* of prediction using Equation (2).

2) Update the *PN* respect to the following comparisons:

   a) If $RMSE \leq ETh$, $PN=PN+2$.

   b) If $RMSE > ETh$ & $RMSE \leq ETh*1.5$, $PN=PN+1$.

   c) If $RMSE > ETh*1.5$ & $RMSE \leq ETh*2$, $PN=PN-1$.

   d) If $RMSE > ETh*2$, $PN=PN-2$.

3) Compute the new *PVP* value, $PVP=WS*PN$.

---

4 Performance Evaluation

In this section, we evaluate the performance of PDRv.2 based on some test scenarios and discuss the results. All test scenarios are implemented using Visual Basic and the ANN toolbox in MATLAB [18]. We modify both, the AntNet and TB algorithm, by replacing the transmission delay with the available BW information to be able to compare the PDRv.2 algorithm with them. Three performances parameters are studied:

1. The rejection ratio of path requests,
2. The percentage of accepted BW and
3. The effect of prediction use.

Our experiment is done on two network topologies. The first one is the MIRA network [8] that is shown in Figure 3, where the thicker links have a capacity of 4800 capacity units while the thinner links have a capacity of 1200 capacity units. The second one is a real network topology that is shown in Figure 4. It is a reference topology suited for an advanced hybrid optical and packet network in the U.S. named Internet2 [19]. In contrast to the performance study of PDRv.1, we consider the requests from all possible combinations of source and destination pairs.

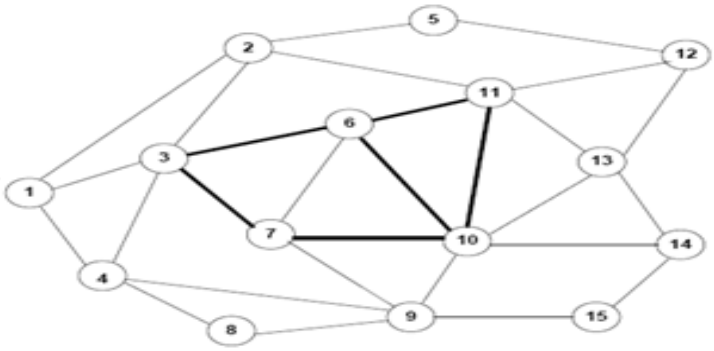

Fig. 3. MIRA network topology.

Fig. 4. Internet2 network topology.

In the MIRA scenario, we examine the performance of the routing algorithms for two generated traffic demands. The first load scenario is a Moderate Load (ML): The arrival of requests follows a Poisson distribution and the holding time of the requests is based on an exponential distribution. The second is a heavy load (HL). In the Internet2 scenario, we examine the performance of the PDRv.2 algorithm for a real traffic demand. The real traffic demands are collected from the trace flies of the NetFlow tool for the first day of 2009 year [19]. Table 1 describes the PDRv.2 parameters and shows the range and used value in our simulation.

Table 1. The PDRv.2 algorithm parameters.

Variable	Value
lc (least interference control parameter)	0.1
M (keep the average of the last M of td)	{15, 20}
δ (learning rate)	{0.01, 0.02}
α (prediction weight)	0.9
WS (window size)	1

4.1 Generated Traffic Scenario

In the next scenario, we consider the MIRA topology and generate two different traffic demands using different values for the Poisson and exponential distributions.

4.1.1 Moderate Load Scenario

Figure 5 shows the rejection ratio of requests for the moderate load scenario. The results show that, the PDRv.2 algorithm rejects approximately 18.36% less requests than the TB algorithm and 49.80% less requests than the AntNet algorithm.

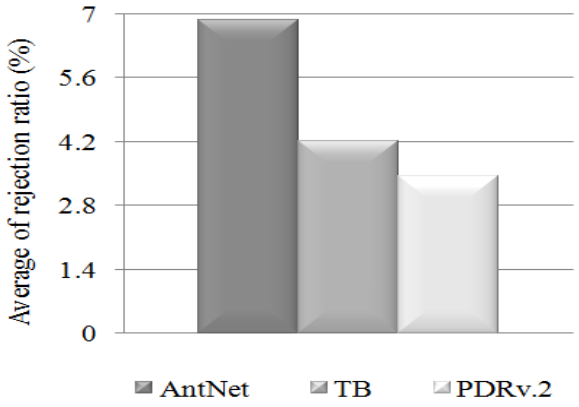

Fig. 5. The rejection ratio of requests for the moderate load scenario.

Figure 6 shows the percentage of accepted BW for the moderate load scenario. The PDRv.2 algorithm accepts approximately 1.0% more bandwidth than the TB algorithm and 4.31% more bandwidth than the AntNet algorithm.

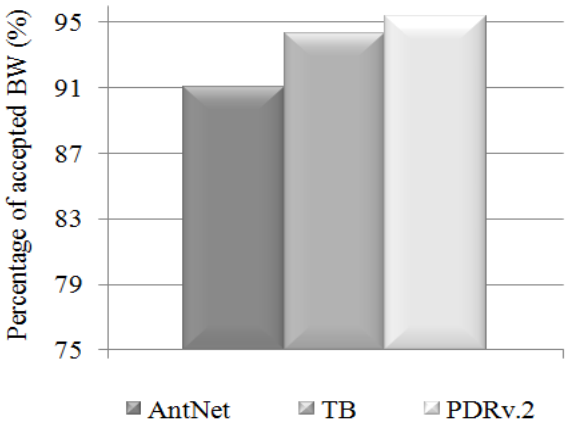

Fig. 6. The percentage of accepted BW for the moderate load scenario.

4.1.2 Heavy Load Scenario

Figure 7 shows the rejection ratio of requests for the heavy load scenario. The results show that, the PDRv.2 algorithm rejects approximately 7.55% less requests than the TB algorithm and 44.23% less requests than the AntNet algorithm.

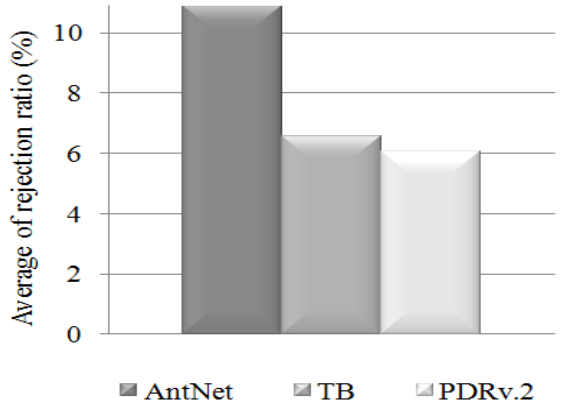

Fig. 7. The rejection ratio of requests for the heavy load scenario.

Figure 8 shows the percentage of accepted BW for the heavy load scenario. The results show that, the PDRv.2 algorithm accepts approximately 0.52% more bandwidth than the TB algorithm and 5.93% more bandwidth than the AntNet algorithm.

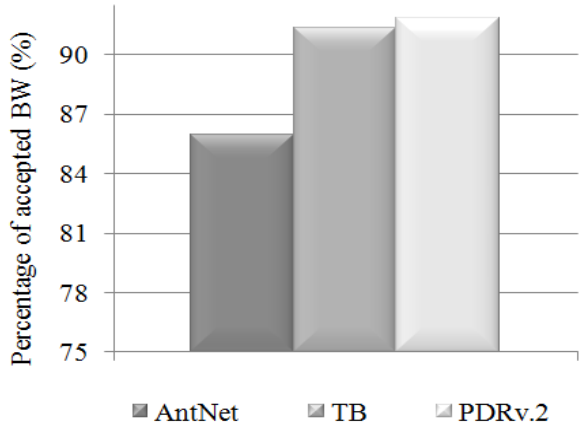

Fig. 8. The percentage of accepted BW for the heavy load scenario.

4.2 Real Traffic Scenario

Figure 9 shows the rejection ratio of requests for the real traffic scenario. The results show that, the PDRv.2 algorithm rejects approximately 26.68% less requests than the TB algorithm and 46.14% less requests than the AntNet algorithm.

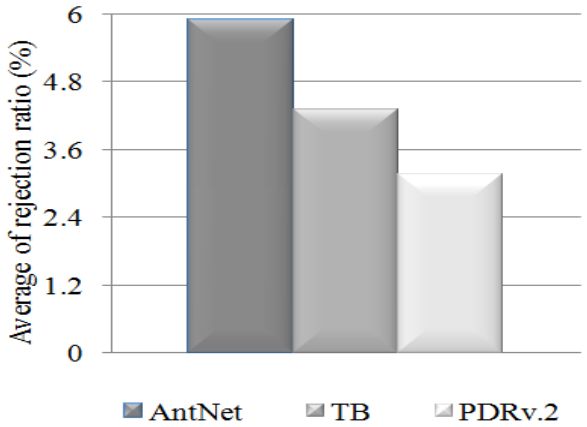

Fig. 9. The rejection ratio of requests for the real traffic scenario.

Figure 10 shows the percentage of accepted BW for the real traffic scenario. The results show that, the PDRv.2 algorithm accepts approximately 0.43% more bandwidth than the TB and 2.16% more bandwidth than the AntNet algorithm.

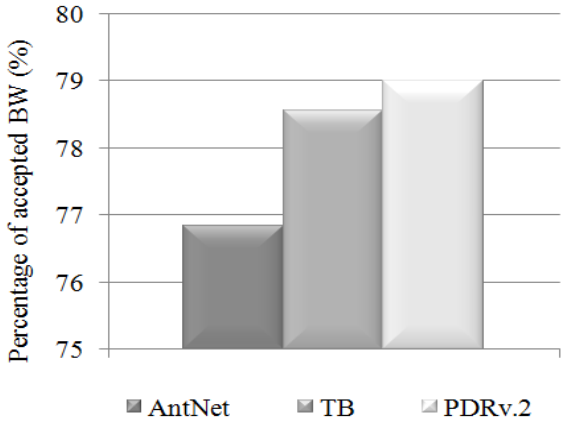

Fig. 10. The percentage of accepted BW for the real traffic scenario.

4.3 The effect of Prediction Use

Table 2 shows the rejection ratio of requests for different prediction weights. In this section, we aim to study the effect of prediction use. Therefore, we run the PDRv.2 algorithm one time with prediction weigh (α) = 0.9 and another time with prediction weigh (α) = 0. In general, the PDRv.2 algorithm with prediction weigh (α) = 0.9 algorithm rejects the least requests in all scenarios of traffic types. In other words, the use of prediction has a positive impact on the routing performance.

Table 2. The rejection ratio of requests (%) for different prediction weighs.

Traffic type	Medium load	Heavy load	Real traffic
PDRv.2(α=0.9)	3.45	6.08	3.16
PDRv.2(α=0)	3.79	6.46	3.26

5 Conclusion and Future Work

We have introduced a new self-optimizing mechanism to enhance the performance of the PDR algorithm. The PDRv.2 algorithm is a member of a class of traffic-aware routing algorithms based on the behavior of ants. The main idea of PDR is to let the ants use a combination of the link state information and the predicted available bandwidth instead of the ant's trip time to determine the amount of pheromone to deposit. The new mechanism has the ability to locally adapt the prediction validity period depending on the prediction accuracy in order to efficiently predict the link traffics. We have compared the performance of our proposed PDRv.2 algorithm with the TB and AntNet algorithms in two different networks and with different traffic types. In general, our algorithm performs considerably better than the comparative algorithms with respect to different performance comparison criteria.

In the future, we plan to test the performance of the PDR algorithm with more complex network topologies. We plan to test the performance of PDR with respect to other performance criteria too. In addition, a comparison of the PDR algorithm with other ant algorithms is planned.

References

1. Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., Xiao, X.: Overview and Principles of Internet Traffic Engineering. RFC3272, (2002)
2. Moy, J.: OSPF Version 2. RFC 2328, (1998)
3. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031, (2001)
4. Sim, K.M., Sun, W.H.: Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions. In: IEEE Trans. on Sys. , Man and Cyber. , vol. 33, no 5, pp. 560-572, (2003)
5. Kunkle, Daniel R.: Self-organizing Computation and Information Systems: Ant Systems and Algorithms. Technical report, Rochester Inst. of Technology, (2001)
6. Dijkstra, E. W.: A note on two problems in connexion with graphs. J. Numerische Mathematik, vol. 1, no. 1, pp. 269–271, (1959)
7. Guerin, R., Orda, A., Williams, D.: QoS routing mechanisms and OSPF extensions. J. IEEE Global Telecommunication, vol. 3, pp. 1903-1908, (1997)
8. Kar , K., Kodialam, M., Lakshman, T.V.: Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. IEEE J. Selected Areas in Comm., vol. 18, no. 2, pp. 2566-2579, (2000)
9. Bagula, A.B., Botha, M., and Krzesinski, A.E.: Online Traffic Engineering: The Least Interference Optimization Algorithm. In: ICC '04, pp. 1232-1236, (2004)
10. Einhorn, E., Mitschele-Thiel, A.: RLTE: Reinforcement Learning for Traffic-Engineering. In: 2nd Inter. Conf. on Autonomous Infrastructure, Man. and Sec., pp. 120-133, (2008)
11. Turky, A. A., Mitschele-Thiel, A.: MPLS Online Routing Optimization Using Prediction. In: Second Workshop on Network Control and Optimization, LNCS, vol. 542, pp. 45–52, (2009)
12. Turky, A. A., Mitschele-Thiel, A.: Use of Load Prediction Mechanism for Dynamic Routing Optimization. In: IEEE Symposium on Comp. and Communications, pp. 782-786, (2009)
13. Caro, G.D., Dorigo, M.: AntNet: Distributed stigmergetic control for communications networks. J. Artificial Intelligence Research, vol. 9, pp. 317-365, (1998)
14. Yun, H., Heywood, A.: Intelligent Ants for Adaptive Network Routing. In: CNSR ’04, pp. 255-261, (2004)
15. Gabber, E., Smith, M. A.: Trail Blazer: A Routing Algorithm Inspired by Ants. In: ICNP’04, pp. 36-47, (2004)
16. Turky, A. A., Mitschele-Thiel, A.: Prediction-based Decentralized Routing Algorithm. In: Self-organizing, Adaptive, Context-sensitive distributed systems, EASST, vol. 17, (2009)
17. Eswaradass, A., Sun, X.H., Wu, M.: Network Bandwidth Predictor (NBP): A System for Online Network performance Forecasting. In: IEEE International Symposium on Cluster Computing and the Grid, pp. 265-268, (2006)
18. Hagan, M.T., Demuth, H.B., Beale, M.H.: Neural Network Design, Boston, MA: PWS Publishing, (1996)
19. Neural Network Toolbox, MATLAP R2009a version, http://www.mathworks.com /products/neuralnet.
20. Internet2 Observatory Data Collections, http://www.internet2.edu/observatory/archive/.