Survey of Distributed Multipath Routing Protocols for Traffic Management

Ditixa Vyas
CHARUSAT University,
AT & Po.Changa, Dist-Anand
PIN-388421, Gujarat, India

Ritesh Patel
CHARUSAT University,
AT & Po.Changa, Dist-Anand
PIN-388421, Gujarat, India

Amit Ganatra, PhD.
CHARUSAT University,
AT & Po.Changa, Dist-Anand
PIN-388421, Gujarat, India

ABSTRACT

In the Internet, *Traffic management* includes *congestion control*, *routing protocols* and *traffic engineering*. Traffic management is the adaptation of source rates and routing to achieve the goals of users and operators. The end hosts adapt their sending rates to network congestion, and based on that network operators adapt the routing to the measured traffic. *Congestion Control* is done by source node depending on the ICMP message sent by any intermediate node of the packet delivery path towards the end node. *Routing protocols* are implemented on the routers for routing purpose. *Traffic engineering* is an important mechanism for Internet providers seeking to optimize network performance and traffic delivery. As network grows in size and complexity network management needs the optimized protocols instead of existing protocols. There may be a question whether the joint system of congestion control (transport layer) and routing (network layer) is optimal and stable. By using the optimal traffic engineering and optimization model for TCP, the new algorithms can be implemented. The latest algorithms uniting the network and transport layers in a multi-layer approach and can be optimal and maximizes aggregate user utility. In this paper we present various distributed multipath routing protocols which are nearer to optimized and one of them Distributed Adaptive Traffic Engineering(DATE) is the joint system of congestion control and routing.

General Terms: Optimization, Traffic Management

Keywords: Traffic engineering, Routing, Congestion control, Optimization.

1. INTRODUCTION

Congestion Control, Traffic Engineering and Routing are included in Traffic Management. Traffic Management has three main players: users, routers and operators. The operators are involved in traffic engineering, routers do routing and users do congestion control [1], [2]. On the specific edge, the users do congestion control to achieve the rate for sending the data, routers run shortest path routing based on link weights, operators need to observe the network for congestion and tune the link weights to direct traffic to another links other than congested links. Three players are related to each other. Operators monitor the whole network for congestion and based on that the users set their sending rates and then the routers forward the traffic on the specific links. In traffic engineering the tuning the link weights is NP-hard problem and it forces the operators to convert it into solvable problem.

Optimization theory has been successfully used to analyze and design various components of traffic management [3], [4], [8]. Optimization theory has been used to guide the design of new congestion control protocols. In addition, traffic engineering imposes an optimization problem on the system, and optimization theory has been used to analyze proposed traffic-engineering protocols. Optimization theory can be used in traffic management using the concept of decomposition. Optimization decomposition is the process to decompose a single optimization problem into multiple sub-problems and each of those can be solved locally. Using optimization decomposition to derive any protocol the mathematical model is assumed and based on that the actual protocol is built [6], [8].

Section 2 describes the scenario how distributed traffic management protocols can be implemented in today's internet: where sources adapt their sending rates along multiple paths according to congestion feedback from the links.

Section 3 describes various distributed multipath routing protocols. Some of them are working with congestion control, some of them are working on traffic engineering and routing. The DATE protocols is the joint system of congestion control and routing.

2. TRAFFIC MANAGEMENT WITH DISTRIBUTED MULTIPATH ROUTING

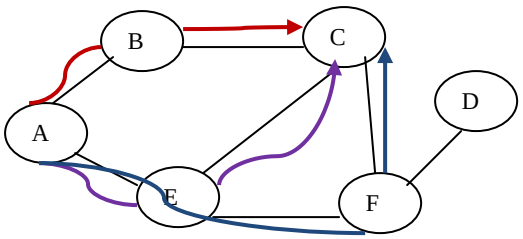

Fig 1: Three paths between node A and destination node C

Traffic management decides how much traffic traverses each path in a network. The end hosts run congestion control to adapt sending rates, and routing protocols select a *single path* between source node and destination node. In this paper, we present several *distributed* traffic-management protocols where sources adapt sending rates on *multiple paths* to a destination. This is illustrated in Figure 1, source node A computes its sending rate on each of its three paths to destination node C, based on feedback regarding the actual load on the path. The advantage of distributed algorithms is

that they adapt at a single timescale (on the order of RTTs), and is able to respond quickly to traffic shifts [8], [12].

For implementing distributed algorithm, multiple paths must exist between a source destination pair, the sources must have knowledge of and control over the multiple paths, and packet forwarding should be possible on each of the specified paths. We address all these requirements in this section. The 'sources' and 'paths' in an algorithm can map to different network elements depending on the scenario. For example, in today's traffic-management system, sources can be end hosts (which run distributed congestion-control algorithms), or edge routers (which have been proposed to run distributed traffic engineering protocols).

Multiple Paths Exists: Because of many small networks are connected to multiple upstream ISPs, multiple end-to-end paths often exist, most ISPs have multiple paths between a pair of edge routers, and large ISPs often connect to each other in multiple locations. Most routing protocols only forward packets on a single path between a source and destination, though the underlying resources can be more efficiently utilized if traffic is *dynamically* balanced between multiple paths.

Multiple Paths for Sources: Although path diversity exists in the Internet, sources cannot always access the multiple paths, since directing packets onto *any* end-to-end path can require cooperation between multiple networks.

Packet Forwarding to Specific Paths: Once sources have decide to forwarding the specific number of packets between multiple paths, the data plane must ensure packets are split between the specified paths accordingly. A packet can be forwarded onto a specific path through *tunnels*. There are two prevalent tunneling techniques in use today: IP-in-IP tunnels and Multi Protocol Label Switching (MPLS). In both cases, establishing a tunnel involves "pushing" an extra IP header (or label) at the tunnel ingress and "popping" the IP header (or label) at the tunnel egress. In the case of MPLS, each intermediate router also stores a label-based forwarding table, so that it can direct a packet to an appropriate outgoing link based on the label. No tunneling is required in the case of a multihomed stub, where the stub network just chooses an outgoing link, rather than the entire path a packet follows. Today, routers can already split traffic equally amongst multiple paths using a variety of techniques, thus it is not difficult to extend such techniques to achieve arbitrary splitting percentages [8].

3. VARIOUS DISTRIBUTED MULTIPATH ROUTING PROTOCOLS

3.1 MATE: MPLS Adaptive Traffic Engineering

3.1.1 Introduction

MATE is targeted for switched networks such as Multi Protocol Label Switching (MPLS) networks. The main goal of MATE is to avoid or minimize the network congestion by adaptively balancing the load among multiple paths based on measurement and analysis of path congestion. MATE adopts an approach in that intermediate nodes are not required to perform traffic engineering or measurements besides normal packet forwarding [10].

Some of the features of MATE include:

- Partially distributed adaptive load-balancing algorithm
- End-to-end control between ingress and egress nodes
- No knowledge of traffic demand is required
- No assumption on the scheduling or buffer management schemes at a node
- Optimization decision is taken based on path congestion measure Minimal packet reordering

3.1.2 MATE Overview

Fig 2 shows a functional block diagram of MATE located at an ingress node. Incoming traffic enters into a filtering and distribution function whose objective is to facilitate traffic shifting among the LSPs in a way that reduces the possibilities of having packets arrive at the destination out of order. The mechanism does not need to know the statistics of the traffic demands or flow state information [10].

The traffic engineering function consists two phases:

- Monitoring phase.
- Load balancing phase.

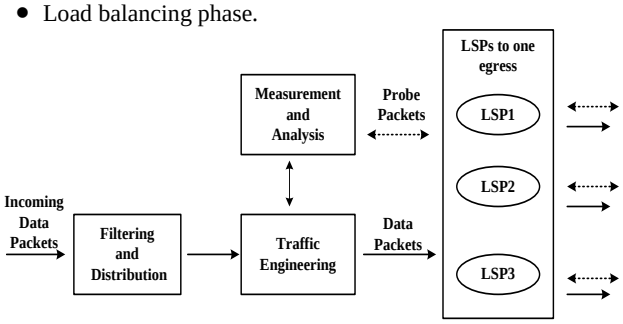

Fig 2: MATE function in an Ingress Node

In the monitoring phase, if an appreciable and persistent change in the network state is detected, transition is made to the load balancing phase. In the load balancing phase, the algorithm tries to equalize the congestion measures among the LSPs. Once the measures are equalized, the algorithm moves to the monitoring phase and the whole process repeats[10].

The role of the measurement and analysis function is to obtain *one-way* LSP statistics such as packet delay and packet loss. This is done by having the ingress node transmit *probe* packets periodically to the egress node which returns them back to the ingress node. Probing may be done per class, i.e., probe packets have the same type of service header information as the traffic class being engineered. Based on the information in the returning probe packets, the ingress node is able to compute the one way LSP statistics. Estimators of LSP statistics from the probes may be obtained reliably and efficiently using bootstrap resampling techniques. These techniques provide a dynamic mechanism for sending the probe packets so that the smallest number is automatically selected as the traffic conditions change to provide a given desirable degree of accuracy.

The efficacy of any state-dependent traffic engineering scheme depends crucially on the traffic measurement process. MATE does not require each node to perform traffic measurement. Only the ingress and egress nodes are required to participate in the measurement process [10].

3.1.3 Limitations of MATE

MATE minimizes the sum of the delays in the network. Unless the network is congested, the delay on a path is constant and equal to propagation delay [13].

MATE is not fully distributed algorithm. Because it assume that ingress nodes have instantaneous knowledge of the whole network state. On the other hand, MATE does not need the core routers to report link utilization to ingresses.

3.2 TeXCP

TeXCP is an *online* distributed TE protocol that balances load in real time, responding to actual traffic demands and failures. TeXCP uses multiple paths to deliver traffic from an ingress to an egress router, moving traffic from over utilized to under-utilized or not utilized paths. These adaptations are designed such that, though done independently by each edge router based on local information, they balance load in the whole network without oscillations [11].

TeXCP has two components.

- A load-balancer Takes as input the state of the network and shifts traffic from one path to another to minimize the utilization.
- Close loop controller
  Each path in the network has a closed-loop feedback controller that collects network feedback and ensures traffic stability on the path.

For large network, report packet/T_p is sent by router for each outgoing line which contains the link utilization and the feedback.

To balance traffic, a TeXCP agent needs to keep track of the utilization of each path available to it (i.e., maximum link utilization along the path). The TeXCP agent maintains a probe timer which fires every T_p seconds. T_p should be larger than the maximum round trip time in the network. The default value is 100ms. Smaller values of T_p make TeXCP converge faster whereas larger values decrease the overhead (§6). When the timer fires, the TeXCP agent sends a small probe on each of its paths. A router that sees a probe packet checks whether the utilization reported in the packet is smaller than the utilization of its output link, in which case it overwrites the utilization in the probe with its own. The egress node at the end of the path uncast the contents of the probe packet to the ingress node, which delivers it to the appropriate TeXCP agent. This packet goes directly to the ingress node and is not processed by intermediate routers [11].

3.3 DEFT: Distributed Exponentially-weighted Flow Splitting

3.3.1 Introduction

The extension to the existing protocols OSPF and ISIS, called Distributed Exponentially-weighted Flow Splitting (DEFT), in which the routers direct traffic on non-shortest paths. DEFT leads not only to a simpler optimization problem, but also to weight settings that provably perform always better than OSPF and ISIS [7].

The flexibility of routing on non-shortest paths of DEFT can bring tremendous improvement in approaching network wide traffic engineering objective and still keep the simplicity and scalability of link-state routing protocols. DEFT can achieve a flow with total link cost of 383.31 units (within 0.9% of optimality).

There are two main reasons for the difficulty in tuning OSPF for good performance. First, the routing mechanism restricts the traffics to be routed only on shortest paths. Second, link weights and the traffic matrix are not integrated together into the optimization formulation. Both bottlenecks are overcome in DEFT as follows [7]:

- Traffics are allowed to be routed on non-shortest-paths, with exponential penalty on path lengths.
- An innovative optimization formulation is proposed, where both link weights and flows are variables. It leads to an effective two-stage iterative method.

As a result, DEFT has the following desirable properties:

- It finds a unique flow of traffic for a given link weight setting in polynomial time.
- It is probably better than OSPF in terms of minimizing the maximum link utilization or the sum of link cost.
- It is readily implemented as an extension to the existing IGP (e.g. OSPF).
- The traffic engineering under DEFT with the two-stage iterative method realize near-optimal flow of traffic even for large-scale network topologies.
- DEFT converges much faster than that OSPF in optimizing procedure.

Here observation is done that the running time per iteration of DEFT is comparable with local search OSPF but the iteration number required for DEFT is much less than that for local search OSPF. Therefore, DEFT is very promising to achieve near optimal traffic engineering within a reasonable time, for large-scale networks also.

3.3.2 Limitations of DEFT

- DEFT for a given traffic matrix. The next challenge would be to explore robust optimization under DEFT, optimizing to select a single weight setting that works for a range of traffic matrices and/or a range of link/node failure scenarios [5].
- DEFT works on link-based flow splitting, while PEFT works on path based flow splitting.
- The two-stage method for DEFT is much slower than the algorithms developed for PEFT [5].

DEFT can realize near optimal TE in terms of a particular objective (total link cost).

3.4 The TRUMP Algorithm

TRUMP is a combination of four distributed algorithms, Partial Dual, Primal Dual, Full Dual and Primal Decomposition. In TRUMP the best features of these four algorithms are combined [8].

3.4.1 Partial-Dual

The partial-dual algorithm works with *effective capacity y* as an additional primal variable [9]. The constraint $y \leq c$ is enforced, resulting in the following equation for updating effective capacity:

$$y_l(t+1) = \text{mininize}_{(y_l \le c_l)} \omega f(y_l / c_l) - s_l(t) y_l \quad (1)$$

In (1), y_l is updated by solving a local optimization using information from the feedback price and the cost function *f*. An economic interpretation is that the effective capacity balances the cost of using a link (represented by *f*) and revenue from traffic transmission (represented by the product of feedback price with the effective capacity).

3.4.2 Primal-Dual

$$\begin{aligned} &\text{maximize} \quad \textstyle\sum_i U_i(\Sigma_j z_j^i) - \omega \sum_l f\left(y_l / c_l\right) \\ &\text{subject to} \quad y \circ c, \\ &\qquad y_l = \textstyle\sum_i \sum_j H_j^i z_j^i, \quad \forall l. \end{aligned} \quad (2)$$

The primal-dual decomposition first decomposes (2) into two sub problems, one responsible for each primal variable. The master problem solves for y assuming a given x*, while the sub problem solves for x assuming a fixed y.
The master problem is as follows:

$$\begin{aligned} &\text{maximize} \quad \textstyle\sum_i U_i(x^*) - \omega \sum_l f\left(y_l / c_l\right) \\ &\text{subject to} \quad y \circ c \end{aligned} \quad (3)$$

where x* is a solution to the following sub problem:

$$\begin{aligned} &\text{maximize} \quad \textstyle\sum_i U_i(x_i) \\ &\text{subject to} \quad Rx \circ y. \end{aligned} \quad (4)$$

In (4) constraint is y(effective capacity) rather than c(actual capacity).

The master problem can be solved through an iterative update of effective capacity :

$$y_l(t+k) = \min(c_l, y_l(t) + \beta_y(s_l(t) - \omega f'(y_l(t)))) \quad (5)$$

where β_y is the effective capacity step size. The primal-dual decomposition is identical to the partial-dual decomposition except that the effective capacity is updated iteratively rather than by solving a local minimization problem.

3.4.3 Full Dual

The full-dual decomposition is similar to the partial dual decomposition but a second dual variable p is introduced to relax the constraint $y \le c$. This dual variable can be interpreted as *consistency price* as it ensures *consistency* between the effective capacity and the capacity constraint at the equilibrium point. As with the feedback price, the consistency price is updated over time using a sub gradient method:

$$p_l(t+1) = [p_l(t) - \beta_p(c_l - y_l(t))] \quad (6)$$

where β_p is the step size for consistency price. Consistency price only comes into play when the capacity constraint is violated, therefore, it is mapped to a non-negative value. The effective capacity update is based on both link prices:

$$y_l(t+1) = \text{minimize}_{y_l} \omega f(y_l / c_l) - (s_l(t) + p_l(t)) y_l \quad (7)$$

The path rate update and feedback price update are identical to that of the previous two algorithms. The full-dual algorithm closely resembles an algorithm presented in, though our objective contains w as a weighing factor.

3.4.3 Direct Path Rate Update: Primal

In all the previous algorithms, auxiliary dual variables were introduced to relax the constraints. In this primal decomposition, we find a direct solution by introducing a penalty function,

$$\text{maximize} \sum_i U_i\left(\sum_j z_j^i\right) - \omega \sum_l P_l\left(\sum_i \sum_j H_{lj}^i z_j^i\right) \quad (8)$$

The derivative of (8) is:

$$\frac{dz_i}{dt} = \beta_z \frac{\partial U_i}{\partial z_j^i}(x_i(t)) - \omega \sum_l P_l'\left(\sum_i \sum_j H_{lj}^i z_j^i(t)\right) \quad (9)$$

where β_z is the stepsize for path rate. Converting (9) into a sub gradient update form and separating link information from source information, we obtain the algorithm as follows.

Path rate update:

$$z_j^i(t+1) = z_j^i(t) + \beta_z z_j^i(t)\left(\frac{\partial U_i}{\partial z_j^i}(x_i(t)) - \sum_l H_{lj}^i s_l(t)\right)$$

Feedback price update:

$$s_l(t+1) = \omega P_l'\left(\sum_i \sum_j H_{lj}^i z_j^i(t)\right)$$

Table 1. Four Decompositions - Differences(Differ in how link & source variables are updated) [14]

Algorithms	Features	Parameters
Partial-dual	Effective capacity	1
Primal-dual	Effective capacity	3
Full-dual	Effective capacity, Allow packet loss	2
Primal-driven	Direct *s* update	1

3.4.6 The TRUMP Algorithm

In the algorithm TRUMP the best features of all four algorithms are combined. In TRUMP, the feedback price has two components as in the *full-dual* algorithm: p_l and q_l. It has been observed that local optimization worked better than sub gradient update, so here, the feedback price update from *primal* algorithm is used as q_l. This has the additional benefit of removing one tuning parameter from the protocol since the update of q_l involves no step size [8].

A local optimization is used for the path rate update as in the dual-based algorithms. The value of *w* is only known at the sources where the *z*'s are computed, and there is only a single value for the network.

TRUMP performs well for a large range of *w*-values when an appropriate step size is chosen.

TRUMP Algorithm [15]:

Feedback price update:

$$s_l(t+1) = p_l(t+1) + q_l(t+1),$$
$$p_l(t+1) = [p_l(t) - \beta_p(c_l - \sum_i \sum_j H_{lj}^i z_j^i(t))]^+ ,$$
$$q_l(t+1) = \omega f'\left(\sum_i \sum_j H_{lj}^i z_j^i(t) / c_l \right)$$

Path rate update:

$$z_j^i(t+1) = \text{maximize}_{z_j^i} U_i\left(\sum_j z_j^i \right) - \sum_l s_l(t) \sum_j H_{lj}^i z_j^i$$

Table 2. Summary of TRUMP [14]

Property	TRUMP
Tuning Parameters	Universal parameter setting Only need to be tuned for small *w*
Robustness to link dynamics	Reacts quickly to link failures and recoveries
Robustness to flow dynamics	Independent of variance of file sizes, for larger files it is more efficient
General	Trumps other algorithms Two or three paths suffice

3.4.7 Limitations of TRUMP algorithm

- Decomposition-based techniques TRUMP (Traffic-management Using Multipath Protocol) alleviates the instability, but their convergence is not guaranteed.
- Not much optimal.
- At different links, inflexible in differentiating the control.

TRUMP performs well when selecting of two or three shortest-hop path.

3.5 DATE : Distributed Adaptive Traffic Engineering

3.5.1 Introduction

Combination of the network and transport layers in a multi-layer approach, this algorithm, Distributed Adaptive Traffic Engineering (DATE), jointly optimizes the goals of end users and network operators and reacts quickly to avoid bottlenecks [13].

In developing an algorithm that jointly optimizes rates and routes in a way that satisfies design criteria. First identify an objective function that balances the goals of end users and network operators, and then explore how to construct a stable, dynamic, distributed system that optimizes for this objective. In the resulting Distributed Adaptive Traffic Engineering (DATE) algorithm, edge routers compute the sending rate per source per path, and each link compute]s its effective capacity. Congestion price is fed back from the links to the sources to prevent the source rates from exceeding the *effective capacity*. On each link, *consistency price* is introduced to force the effective capacity to stay below the actual capacity.

DATE focuses routing and congestion control in a single Autonomous System, where the operator has full view of the offered traffic load and complete control over routing, and a multipath routing model where traffic between source destination pairs can be split arbitrarily across multiple paths. This is not the OSPF or IS-IS protocols used today, but can be implemented using MPLS.

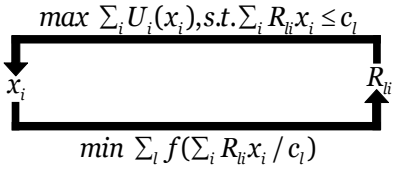

Fig 3: Joint System for congestion control and traffic engineering [13]

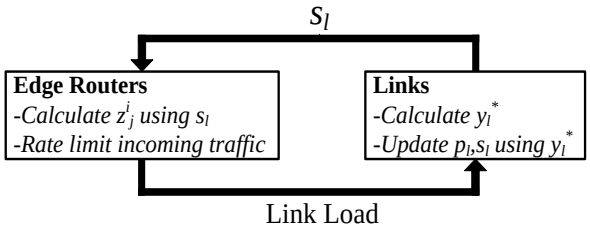

Fig 4: A graphical view of DATE algorithm [13]

Table 3. DATE algorithm

Edge Routers:

$$z_j^i(t+T_p) = \text{maximize}_{z_j^i} U_i\left(\mathbf{1}^T z^i\right) - z_j^i \sum_l s_l(t) H_{lj}^i$$

where z_j^i is the amount of load that TCP session *i* places on its j^{th} path where $x_i = \sum_j z_j^i$

Links:

- Congestion price Update:

$$s_l(t+T_p) = s_l(t) - \beta_s\left(y_l(t) - \sum_i \sum_j H_{lj}^i z_j^i(t) \right),$$

where β_s is the congestion price step size.

- Consistency price Update:

$$p_l(t+T_p) = [p_l(t) - \beta_p(c_l - y_l(t))]^+,$$

where β_p is the consistency price step size. Since, $p_l \geq 0$, it must be mapped to a non-negative value.

- Effective Capacity Update:

$$y_l(t+T_p) = \text{minimize}_{y_l} f(y_l / c_l) - (s_l(t) + p_l(t)) y_l.$$

where y_l is the effective capacity.

3.5.2 Limitations of DATE

Analytic study then proves that a modification to the operator's cost function indeed leads to a provably stable and optimal system, but at the expense of robustness.

3.6 PEFT: Penalizing Exponential Flow-splitting

Link-state routing protocol, PEFT, that directs traffic over multiple paths with an exponential penalty on longer paths. It *provably* achieves *optimal* traffic engineering while retaining

the *simplicity* of hop-by-hop forwarding. It also leads to a significant reduction in the time needed to compute the best link weights [5].

A link-state routing protocol has three components:

- *Weight computation*: The network-management system computes a set of link weights through a periodic and centralized optimization.
- *Traffic splitting*: Each router uses the link weights to decide traffic-splitting ratios among its outgoing links for every destination.
- *Packet forwarding*: Each router independently decides which outgoing link to forward a packet based only on its destination prefix in order to realize the desired traffic splitting.

In PEFT, packet forwarding is just the same as OSPF: destination based and hop-by-hop. The key difference is in traffic splitting. OSPF splits traffic *evenly* among the shortest paths, and PEFT splits traffic along all paths, but penalizes longer paths (i.e., paths with larger sums of link weights) exponentially. While this is a difference how link weights are *used* in the routers, it also mandates a change in how link weights are *computed* by the operator. It turns out that using link weights in the PEFT way enables optimal traffic engineering. Using the Abilene topology and traffic traces, it has been observed that 15% increase in the efficiency of capacity utilization by PEFT over OSPF. Furthermore, an exponential traffic-splitting penalty is the *only* penalty that can lead to this optimality result. In the control plane, PEFT does not change the routing-protocol messages that are sent between the routers but does change the computation done locally on each router based on the weights[5].

In the data plane, routers implement hash-based splitting over multiple outgoing links, typically with an even splitting ratio. PEFT requires flexible splitting over multiple outgoing links, thus it is needed to store the splitting percentages whereas for spitting, the splitting ratio is implicitly even. It requires a little extra storage and processing, not enough to become a new bottleneck, when packets arrive to direct packets to the appropriate outgoing links.

Table 4. Maximum Link Utilization of Optimal Traffic Engineering, PEFT, and Local Search OSPF for Light - Loading Networks [5]

Net. ID	Optimal TE	PEFT	OSPF
Abilene	33.9%	33.9%	39.8%
hier50a	56.4%	56.5%	58.6%
hier50b	44.7%	45.0%	59.2%
rand50	60.6%	60.6%	60.6%
rand50a	60.8%	60.8%	64.7%
rand100	55.0%	55.0%	71.5%

Table 4 shows the maximum link utilizations of optimal traffic engineering, PEFT, and Local Search OSPF for the test case with the lightest loading of each network.

PEFT is very close to optimal traffic engineering in minimizing MLU and increases Internet capacity over OSPF by 15% for the Abilene network and 24% for the hier50b network, respectively.

4. COMPARISION OF VARIOUS DISTRIBUTED PROTOCOLS

Table 4. Comparison of Distributed Protocols

Tech.	Description	Distributed	Congestion Control	Traffic Engineering
TRUMP	Based on feedback from the link it changes the source rate	√	√	×
PEFT	Based on new link weight, using Network Entropy Maximization framework it does link state routing	√	×	√
DEFT	Routers can direct the traffic on non-shortest path based on link weights. Splitting is done by pseudo-random method	√	×	√
DATE	Maximizes the aggregate user utility and minimizes the network congestion	√	√	√
TeXCP	Balanced the load from maximum utilized link to minimum utilized link based on feedback of the link	√	×	√
MATE	Minimizes the sum of the delays in the network	Partially distributed	×	√

5. RELATED WORK

In traffic management research the optimization theory is used for tuning configuration parameters of existing protocols and guiding the design of new protocols . Most of the proposed traffic management protocols consider congestion control or

traffic engineering alone[8]. Several proposed dynamic traffic engineering protocols also load balance over multiple paths based on feedback from links but they do not adapt the source rates. According to recent research, congestion-control and traffic engineering practices may not interact well. In response, many new designs are proposed. Some research analyzes stability of joint congestion control and routing algorithms as DATE algorithm described here. Instated of using the joint system the various optimization methods can be used to proposed new distributed algorithms as TRUMP algorithm.

Same way the distributed algorithms can be proposed for interdomain routing as well as for different types of autonomous systems. The new algorithms can be proposed for ad-hoc network also.

6. CONCLUSION

By using the optimal traffic engineering and optimization model for TCP various multipath distributed algorithms are proposed which are described in this paper. Some algorithms work based on congestion control and some work on routing and traffic engineering independently. The algorithm DATE unites the network and transport layers in a multi- layer approach as a joint system and can be optimal and maximizes aggregate user utility. We have also described the comparisons of various multipath distributed protocols.

In our ongoing work, by using the optimal traffic engineering and optimization model for TCP the new algorithm is proposed which unites the network and transport layers in a multi- layer approach and can be optimal and maximizes aggregate user utility as DATE algorithm. Using optimization theory the new algorithm computes the link price based on feedback from the links along each path then updates the source path rate and depending on that the traffic is distributed on multiple paths. We are in the process of evaluating proposed algorithm in the ns-2 simulator under realistic topologies and workloads. Another thing is that the analysis of various methods to send feedback to source node - piggy-backed on acknowledgment packets, attached to probe packets or flooded throughout the network.

7. ACKNOLEGGEMENT

The authors would like to thank Prof. Amit Thakkar, Chintan Gajjar, Prof. Jennifer Rexford for their helpful discussion related to this paper.

8. REFERENCES

[1] J. He, M. Bresler, M. Chiang, and J. Rexford, "Rethinking Internet traffic management: From multiple decompositions to a practical protocol" in *Proc. CoNEXT*, December 2007.

[2] Ning Wang, Kin Hon Ho, George Pavlou and Michael Howarth, University of Surrey, "An Overview of Routing Optimization for Internet Traffic Engineering" in Volume 1 IEEE, 2008.

[3] Jiayue He and Jennifer Rexford, "Toward Internet-Wide Multipath Routing", in IEEE Network, March/April 2008.

[4] Jiayue He, Jennifer Rexford, Mung Chiang, : Don't Optimize Existing Protocols, Design Optimizable Protocols" in News Letter, ACM SIGCOMM Computer Communication, Volume 37 Issue 3, July 2007.

[5] Dahai Xu, Mung Chiang, Jennifer Rexford, "Link-State Routing With Hop-by-Hop Forwarding Can Achieve Optimal Traffic Engineering", IEEE/ACM TRANSACTIONS ON NETWORKING, Vol. 19, no. 6, December 2011.

[6] Jiayue He, Mung Chiang, Jennifer Rexford, "Can Congestion Control and Traffic Engineering Be at Odds?", GLOBECOM 2006.

[7] Dahai Xu, Mung Chiang, Jennifer Rexford, " DEFT: Distributed Exponentially-weighted Flow Splitting", INFOCOM 2007. IEEE International Conference on Computer Communication May 2007.

[8] Jiayue He, Martin Suchara, Malayan Bresler, Jennifer Rexford, and Mung Chiang, "From Multiple Decomposition to TRUMP: Traffic Management Using Multipath Protocol", http://www.techrepublic.com/whitepapers April 2008.

[9] Daniel P Palomar, Mung Chiang, " A Tutorial on Decomposition Methods for Network Utility Maximization", IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 24, NO. 8, AUGUST 2006.

[10] Anwar Elwalid, Cheng Jin, Steven Low, Indra Widjaja, " MATE: MPLS Adaptive Traffic Engineering", INFOCOM 2001. Twentieth Annual Joint Conference on the IEEE Computer and Communications Societies, 2001

[11] Srikanth Kandula, Dina Katabi, Bruce Davie, Anna Charny, " Walking the Tightrope: Responsive Yet Stable Traffic Engineering", SIGCOMM 2005.

[12] Ke Xu, Hongying Liu, Jiangchuan Liu, Jixiu Zhang, " LBMP: A Logarithm- Barrier-Based Multipath Protocol for Internet Traffic Management", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 3, MARCH 2011.

[13] Jiayue He, Ma'ayan Bresler,Mung Chiang, Jennifer Rexford, " Towards Robust Multi-Layer Traffic Engineering: Optimization of Congestion Control and Routing", IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 25, NO. 5, JUNE 2007.

[14] Jennifer Rexford, "Rethinking Internet traffic management: From multiple decompositions to a practical protocol" in University of Washington, Computer Science and Engineering, Colloquium Series, "uw_cse08_intraff_ipodv", 2008.

[15] Jiayue He, Thesis - "Rethinking Traffic Management: Design of Optimizable Networks" , 2008.