Energy-Aware Two Link-Disjoint Paths Routing

Gongqi Lin, Sieteng Soh, Mihai Lazarescu
Department of Computing
Curtin University of Technology
Perth, Australia
{gongqi.lin@postgrad., s.soh@,
m.lazarescu@}curtin.edu.au

Kwan-Wu Chin
University of Wollongong
Wollongong Australia
kwanwu@uow.edu.au

***Abstract*— Network robustness and throughput can be improved by routing each source-to-terminal (*s*, *t*) demand via two link-disjoint paths (TLDP). However, the use of TLDP incurs higher energy cost. Henceforth, we address the problem of minimizing the energy usage of networks that use TLDP. Specifically, our problem is to maximally switch off redundant network links while maintaining at least $0 \le T \le 100\%$ of (*s*, *t*) TLDP in the network, for a given *T*, and limiting the maximum link utilization (MLU) to no greater than a configured threshold. To address this problem, we present a fast heuristic, called TLDP by Shortest Path First (TLDP-SPF), and extensively evaluate its performance on both real and/or synthetic topologies and traffic demands. Our simulation results show that TLDP-SPF can reduce network energy usage, on average, by more than 20%, even for MLU below 50%. As compared to using Shortest Path routing, while reducing energy by about 20%, TLDP-SPF does not significantly affect (*s*, *t*) path length, even for MLU<50%.**

Keywords – algorithm; robustness; thoughput; power savings; routing; maximum link utilization; two link-disjoint paths

I. INTRODUCTION

Increasing the robustness and reliability of networks are of concern to Internet Service Providers (ISPs), especially given the recent emergence of real time applications such as video on demand and voice over IP [1, 2, 3, 4]. To this end, previous studies [5, 6, 7] have combined link/node-disjoint paths with QoS routing to guarantee various performance requirements, *i.e.*, reliability and delay. The authors of [5] and [7] consider restorable QoS routing. Specifically, the algorithm in [7] constructs a restoration topology containing a set of bridges, each of which provides backup for a portion of the primary QoS path. In [6], the authors present a problem, called multiple constrained link-disjoint path pair (MCLPP), to find link/node-disjoint paths in multiple dimensions, and prove its NP-Completeness when one or more link metrics are used. The existence of high path redundancy means there is opportunity to save energy via power-aware traffic engineering [8].

Our paper describes an optimization problem that aims to reduce the energy usage of networks that support two link-disjoint paths (TLDP). Specifically, our problem aims to reduce network energy usage while ensuring that a network maintains at least $0 \le T \le T_{\max}$ percent of all possible (*s*, *t*) TLDP and each link's maximum link utilization (MLU) is at most $0 \le U_T \le 1.0$; $T_{\max}$ is the percentage of the total number of (*s*, *t*) paths that have at least one TLDP. Specifically, to provide energy efficient, fault-tolerant and high bandwidth routing service to upper layer applications, we route each (*s*, *t*) demand through its (*s*, *t*) TLDP, if one exists, while powering off idle/unused links to reduce energy expenditure. The MLU constraint is important because ISPs typically bound links' MLU in order to minimize forwarding delay and to absorb any spike in traffic resulting from network failures [10]. In summary, our contributions are twofold: firstly, we pose a problem, called Energy-Aware Two Link-Disjoint Paths Routing (EAR-TLDP). Our problem is important in reducing the energy usage of networks that employ TLDP to improve fault-tolerance and/or bandwidth/throughput; secondly, we propose a novel algorithm, called Two Link-Disjoint Paths by Shortest Path First (TLDP-SPF), to solve EAR-TLDP. Our algorithm identifies network links that can be powered-off under two constraints: the threshold *T* and the maximum link utilization U_T. To the best of our knowledge, this is the first algorithm that jointly reduces the number of links in a wired network whilst satisfying these two constraints. We note that reference [8] and [9] have proposed power-aware routing algorithms, but they do not require their generated paths to be link disjoint.

The rest of the paper is organized as follows. Section II defines notations and EAR-TLDP. Section III describes TLDP-SPF. Section IV evaluates TLDP-SPF using both real and/or synthetic topologies and data. Finally, Section V concludes the paper.

II. PROBLEM FORMULATION

In this section, we first outline all definitions and notations, before formalizing the problem at hand.

A. Notation

Consider a computer network that is represented by a weighted directed graph $G(V,E)$, where V is the set of n nodes, and E is the set of m links. Each node in V represents a router, and each link $e \in E$ between nodes v_i and v_j ($v_i, v_j \in V$, $v_i \neq v_j$) represents a communication channel with finite capacity/ bandwidth $c(e)>0$. For edge e, let $f(e)$ be the total flow on an edge e that has $0 \le f(e) \le c(e)$, and hence, its utilization is given by $u(e)=f(e)/c(e)$. This utilization is bound by a given threshold $0 \le U_T \le 1.0$; *i.e.*, any $u(e)$ cannot be larger than U_T. A link's remaining capacity is defined as $r(e)=U_T * c(e)-f(e) \ge 0$. We assume each link e can be switched-off independently.

Let $D = \{d_i=(s, t, d^{st}) \mid$ demand i from a source node $s = 1, \ldots, n$ to a terminal node $t = 1,\ldots, n$ that has traffic flow $d^{st}\}$.

For each demand d_i, let SP_i=$\{sp_{iq}$ | all (s, t) path for d_i indexed by q>0$\}$, and TLD_i=$\{tld_{ip}$| all (s, t) TLDP for d_i indexed by p>0$\}$. Note that tld_{ip}=$\{sp_{ix}, sp_{iy}\}$, where sp_{ix}, sp_{iy} ∈SP_i have no common links. Let R_j be a possible set j that contains $|D|$ number of (s, t) single or multi paths with sufficient capacity to route all demands in D, *i.e.*, R_j=$\{R_{ji}$ | a set of one or more paths in SP_i and/or in TLD_i that can be used to route $d_i\}$. The set of all possible solutions to route all demands in D is denoted as R=$\{R_j | j$=1, 2, …, $|R|\}$. The length of R_{ji}, $\text{L}(R_{ji})$, is equal to the maximum hop count among all paths in R_{ji} for d_i, while its bandwidth, $\text{B}(R_{ji})$, is calculated from the sum of the smallest $r(e)$ for each path in R_{ji}, and by the definition of R_{ji}, we have $\text{B}(R_{ji})\geq d^{st}$. We assume the network has sufficient resources, *i.e.*, link capacities, to route all demands. Let $TP_j \subseteq R_j$ be the set of all $R_{ji} \in R_j$ that include at least one $tld_{ip} \in TLD_i$, and $\text{M}(R_j)$ be the percentage of the total number of (s, t) pairs in R_j that are routed over TLDPs, *i.e.*, $\text{M}(R_j)$=$|TP_j|/|R_j|$*100%. Finally, let $\text{S}(R_j)$ be the total number of links used in set R_j and $\text{U}(R_j)$=max$\{f(e)/c(e)| \forall e \in R_j\}$ be the maximum link utilization of R_j.

B. Problem Statement

Consider a tuple (G, D, T, U_T), where G (V, E) is a network topology, D is a set of traffic demands, $0 \leq T \leq T_{\max}$ is a given threshold, where $0 \leq T_{\max} \leq 100\%$ is the percentage of the total number of (s, t) pairs that have at least one TLDP, *i.e.*, $T_{\max}$=MAX$\{\text{M}(R_j)| j$=1, 2, …, $|R|\}$, and U_T is the maximum link utilization threshold. Our Energy Aware Two Link-Disjoint Paths Routing (EAR-TLDP) problem is defined as follows.

EAR-TLDP: Find a set of paths $R_{\min} \in R$ that can be used to route all demands in D such that

$$\text{S}(R_{\min}) = \text{MIN}\ \{\text{S}(R_1), \text{S}(R_2), \ldots, \text{S}(R_{|R|})\},\ R_i \in R \quad (1)$$

$$\text{M}(R_{\min}) \geq T \quad (2)$$

$$\text{U}(R_{\min}) \leq \text{U}_\text{T} \quad (3)$$

Eq. (1) computes the solution, *i.e.*, to find $R_{\min}$ that contains the minimum number of total power-on links. Eq. (2) states that the ratio of the total number of (s, t) pairs that use TLDP for routing in $R_{\min}$ must be no less than a given threshold T. Eq. (3) ensures the MLU of links must be no greater than U_T. Each link e in $R_{\min}$ can be either *on* or *off*, and therefore our EAR-TLDP is a mixed integer-programming (MIP) problem, a well-known NP-hard problem [11].

III. GREEN ROUTING ALGORITHMS

We now present our heuristic called Two Link-Disjoint Paths by Shortest Path First (TLDP-SPF), see Fig. 1, to solve EAR-TLDP. Initially, the set of deleted (remaining) links E_d (E_r) is an empty set (E); TLDP-SPF produces E_d as its output.

A. TLDP-SPF Algorithm

As shown in Fig. 1, TLDP-SPF has four main steps, divided into two phases: initialization and re-routing. The initialization phase first generates a set of candidate paths (Step 1), and then uses function **TLDP-Routing**(), described later, to distribute the traffic of all demands in D through candidate paths (Step 2), and calculate T_{max}. The second phase determines the links that have to be switched off such that remaining links are sufficient to route all traffic demands while satisfying the threshold $0 \leq T \leq T_{max}$ and MLU constraints.

One can use *Yen*'s algorithm [12] in Step l to generate the $k \geq 1$ shortest paths, KSP_i=$\{sp_{i1}, sp_{i2}, \ldots, sp_{ik}\}$, for each demand d_i; we assume each link has a delay of one unit, and thus each (s, t) path length is equal to its (s, t) hop count. Note that for each demand d_i, we have $KSP_i \subseteq SP_i$. Let KSP=$\{KSP_i|$ i=1,…,$|D|\}$ be the set of all k-shortest paths for all demands in D. In Step 2, our algorithm calls function **TLDP-Routing**(), shown in Fig. 2, that contains two sub-parts to distribute traffic in the network.

```
TLDP-SPF(G(V,E),D,T,U_T)
Begin
E_d = Φ, E_r = E;
/*Phase 1*/
1) Generate KSP_i in G(V,E) for each demand d_i;
2) Call TLDP-Routing(G(V, E), D, T, KSP);
/*Phase 2*/
3) For each e∈E_r do
     If f(e) = 0 then
       E_d = E_d + e; E_r = E_r – e; // remove e
     Else
       r(e) = U_T*c(e) – f(e); // update the remaining capacity
   End-For
4) For each e ∈ E_r in descending order of its r(e) do
     //routing D in G with one less edge is feasible
     If TLDP-Routing(G(V, E_r-e),D,T, KSP) = true then
       E_d = E_d + e; E_r = E_r – e; // remove e
       Go to 3);
   End-For
   Return E_d
End
```

Figure 1. TLDP-SPF algorithm

```
TLDP-Routing(G(V, E_r),D,T, KSP)
Begin
count =0;
  For each d_i ∈D do
 /* Sub-Part 1 */
    Call TLDPP(G(V, E_r), KSP_i) to generate TLD_i; //KSP_i ∈ KSP
 /* Sub-Part 2 */
    If count < T*|D| then
      If |TLD_i| >0 then
        If Distribute_TLD (KSP_i , TLD_i, d^st)=true then
          count++;
        Else If Distribute_SP(KSP_i, d^st) = false then
          Return false;
      Else
        If Distribute_SP(KSP_i, d^st) = false then
          Return false;
  End-For
  Return true;
End
```

Figure 2. Function TLDP-Routing()

```
TLDPP(G(V, E_r), KSP_i)
Begin
  For each sp_ix ∈ KSP_i do
    Generate G_1(V, E_1) from G(V, E_r) by deleting all edges in sp_ix;
    Generate sp_iy, the (s,t) shortest path from G_1;
    If G_1 contains sp_iy then
      Store (sp_ix, sp_iy) into TLD_i
  End-For
  Return TLD_i.
End
```

Figure 3. Function TLDPP()

Sub-Part 1 uses function **TLDPP**(), shown in Fig. 3, to generate two-link-disjoint paths, tld_{ip}, from each sp_{ix} in KSP_i. The function stores each tld_{ip} for d_i in set TLD_i in increasing path length order. Sub-Part 2 aims to distribute each traffic demand in D through its TLDP, if possible, *i.e.*, when

```
Distribute_SP(KSP_i, d^x)
Begin
  //Distribute traffic with multiple paths routing
  For each sp_iq ∈KSP_i in ascending order of its length do
     If d^x ≤ B(sp_iq) then
          Insert sp_iq in R_ji;
          Increase f(e) of each edge e in sp_iq by d^x;
          Return true;
      Else
          Route B(sp_iq) flow of the traffic through sp_iq;
          Insert sp_iq in R_ji;
          d^x = d^x - B(sp_iq);
   End-For
End
```

$|TLD_i|>0$; otherwise, it calls function **Distribute_SP**(), shown in Fig. 4, to distribute the traffic volume d^x through one or more paths in its KSP_i.

Figure 4. Function Distribute_SP()

```
Distribute_TLD (KSP_i , TLD_i, d^st)
Begin
 //Distribute traffic with TLD; let tld_ip ={sp_ix, sp_iy}
  For each tld_ip∈ TLD_i in ascending order of L(tld_ip ) do
     If d^st /2 ≤ B(sp_ix) then  //let  B(sp_ix) ≤ B(sp_iy)
          // Route traffic flow d^st/2 through sp_iy and sp_ix
          Increase f(e) of each edge e in  sp_ix and sp_iy  by d^st/2;
          Insert sp_iy and sp_ix in R_ji;
          Return true;
   End-For
//there is no tld_ip with enough capacity, use the shortest tld_ip ; i.e., p=1
    Increase f(e) of each edge e in  sp_ix by B(sp_ix);
    Increase f(e) of each edge e in  sp_iy  by B(sp_ix);
    Insert sp_iy and sp_ix in R_ji;
    Return Distribute_SP(KSP_i, d^st – 2*B(sp_ix));
End
```

Figure 5. Function Distribute_TLD()

For each demand with $|TLD_i|>0$, **Distribute_TLD**(), shown in Fig. 5, routes traffic demand d^{st} similar to ECMP [13]. That is, it aims to route the traffic of d_i evenly through the two paths in $tld_{ip}=\{sp_{ix}, sp_{iy}\}$, *i.e.*, with $d^{st}/2$ of the traffic in each path and we assume $B(sp_{ix}) \le B(sp_{iy})$. Note, $B(sp_{iq})= \min \{r(e)| e\in sp_{iq}\}$. Specifically, **Distribute_TLD**() assumes the application requires two link-disjoint paths to carry the same sized traffic since both of them can be a protection path for each other. If $d^{st}/2 \le B(sp_{ix})$, then each path carries traffic size $d^{st}/2$, and the selected TLDP is put into R_{ji}; otherwise, each path carries traffic of size $B(sp_{ix})$, and the remaining traffic, *e.g.*, $d^{st}-2*B(sp_{ix})$ is routed through one or more paths in TLD_i and/or KSP_i. The function updates flow $f(e)$ of each edge e in the selected paths.

Function **TLDP-Routing**() returns true if it can successfully route all demands in D. For this case, it updates only R_{ji} for each affected demand d_i and the total flows on each edge e, $f(e)$, used for the demand. Otherwise, it returns false. Notice that, for Phase 1, we assume the network contains sufficient bandwidth to carry the traffic demands, and thus it never returns false. Further, we set T=1 so that each demand can be routed through its TLDP whenever possible, and $T_{\max}$, the maximum ratio between the number of demands with $|TLD_i|>0$ and the total number of demands, as shown in Fig. 2, is set to $T_{\max} = count/|D|$.

Phase 2 uses the initial distribution of traffic and T_{max} produced by Phase 1 as its inputs, and produces a set of switched-off links and the routes of all demands in D. Step 3 switches off each link that is not used to carry any traffic in Phase 1; *i.e.*, each link with $f(e)$=0; otherwise, for all links with $f(e)>0$, it calculates their spare capacity $r(e)=U_T*c(e)-f(e)$. Step 4 aims to remove a link e with the largest spare capacity since rerouting traffic that goes through the link is more probable. This step uses **TLDP-Routing**() to check if all traffic can be routed through the remaining edges in E_r, excluding e, while satisfying the constraints that there are at least $T*|D|$ demands that are routed through their TLDP and each link utilization does not exceed U_T. If the function returns true (feasible), then the step removes e from E_r, and stores it into E_d. Note that for this case the function has changed the total flow of each affected edge, and therefore Step 3 is reused to update E_r, E_d and the remaining capacity of each edge before repeating Step 4 for the next candidate e to be switched off. If **TLDP-Routing**() fails in Step 4, it recovers the status of link e, and continue with the remaining edges in E_r. Thus, Step 4 is iterated no more than $|E|$ times.

B. Time Complexity

Yen's algorithm [12], used in Step 1 of TLDP-SPF, requires $O(kn(m+n\log n))$ time to generate k shortest paths in a network with m links and n nodes for each demand d_i; thus Step 1 requires $O((|D|*kn(m+n\log n))= O((kn^3(m+n\log n))$ since $|D|\le n^2$. In Step 2, function **TLDPP**(), used in **TDP-Routing(),** requires $O(k*n^2)$ time for each demand d_i, and thus requires $O(n^2*k*n^2)$ for all demands. Further, either function **Distribute_TLD**() or **Distribute_SP**() requires, in the worst case, $O(k*m)$ time to route the flow of each d_i through its TLD or multiple paths, respectively. Therefore the time complexity of Step 2, *i.e.*, function **TDP-Routing(),** is $O(k*n^4+ k*m*n^2)$. Step 3 is based on the set of edges, so they take $O(m)$ time. Step 4 takes $O(m)$ time to choose a candidate edge and call function **TDP-Routing**(); therefore the total complexity of Step 4 is $O(m*(k*n^4+ k*m*n^2)$. Lastly, the total complexity of TDP-SPF is $O((m+1)*(k*n^4+k*m*n^2)+kn^3(m+n\log n)) = O(k*m*n^4+ k*m^2*n^2)= O(k*n^6)$ since $m=O(n^2)$.

IV. EVALUATION

In this section, we provide experimental findings and present numerical results to confirm the effectiveness of TLDP-SPF. We show that it can achieve considerable power savings in real networks with low impact on network performance, such as delay and maximum link utilization.

A. Experiment Setup

Our experiments are conducted over different network topologies and traffic matrices. We consider three topologies: Abilene [14], GÉANT [15], and the Sprint topology derived from Rocketfuel [16]. We used the Abilene topology and its 288 traffic matrices measured on Sep. 5th, 2004 for every 5 minutes within 24 hours – all of which are provided by the authors of [14]. For GÉANT, the 96 traffic matrices [15] used

were collected on May 5th, 2005 at an interval of 15 minutes. For Sprint, we set its link capacity following the method in [8], *i.e.*, links connecting Level-1 PoPs are 10Gb/s and the others are 2.5Gb/s, and randomly generate a traffic matrix using the gravity model [8].

For each network, when using shortest path routing with all links switched-on, Phase 1 of our TLDP-SPF finds only up to T_{max}=83.3%, 100%, and 37.4% of the demands in Abilene, GÉANT, and Sprint networks can be routed through their TLDP, respectively. We compute the power saving ratio as the total power of in-active links over the total power of all links in the network. The power consumption of line-cards used in our simulation is specified in [17]. We assume all links use OC3 line cards. Note that as the maximum delay of a single OC-3 link is around 1 ms [18], the length of R_{ji}, $L(R_{ji})$, is reported in milliseconds. Our simulations were performed on a Linux PC with 3.07 GHz CPU and 8 GB RAM. TLDP-SPF requires 3.2, 19.8, and 206.3 CPU seconds, to produce each result for Abilene, GÉANT and Sprint network, respectively.

B. Research Network—Abilene and GÉANT

Fig. 6(a) shows the power savings (top) and average delay (bottom) for the Abilene network when we set T=T_{max}=83.3%, *i.e.*, at its highest constraint, and U_T from 0.1 to 1.0. Specifically, for each of the 10 levels of link utilization, we aim to study the effect of reducing energy, *i.e.*, due to switched-off links, on the delay experienced by users while maintaining T=T_{max}. Note that TLDP-SPF could not find any links to be switched off when we set $U_T \leq 0.2$. For U_T=0.3, TLDP-SPF can switch off between 16.5% and 20% of the links except at time 16h due to insufficient network capacity during peak periods; see top of Fig. 6(a). Further, for $U_T \geq 0.4$, TLDP-SPF is able to yield 20% energy saving while maintaining link utilization to no more than 40%. However, as shown in the bottom of Fig. 6(a), switching off links may affect the average end-to-end delay. While setting $U_T \geq 0.5$ does not affect delay, reducing U_T from 0.5 to 0.4 forces TLDP-SPF to find longer alternative paths, and thus increases the average delay by up to 20%.

Fig. 6(b) presents the power saving (top) and average delay (bottom) of GÉANT network when we set T=T_{max}=100%. TLDP-SPF is able to switched off 24.67% of links in the GÉANT network for U_T=0.3 to U_T=1.0, and thus we only show the results for the threshold of MLU no less than 0.3, *i.e.*, $U_T \geq 0.3$. Notice that U_T=0.2 and U_T=0.3 produce the same energy saving but incur different delays. Although the power saving curve fluctuates during the day due to traffic changes, it always remains around 20.27% ~ 25.97%. However, as shown in the bottom of Fig. 6(b), reducing U_T from 0.2 to 0.1 increases the average delay.

C. Commercial Network—Sprint

Given the traffic matrix generated as described in Section IV.A, we scale the matrix using 10 scaling factors to generate 10 different traffics such that when the demands in each matrix are routed using their shortest paths (SP) the MLU in the network is 10% to 100%, with an increment of 10%. We refer to each traffic matrix as TM_X under SP, where X is the MLU of the network due to the traffic; *e.g.*, TM_40 is the traffic that produces MLU=40% when using SP to route demands in TM_40. Finally, for each TM_10 to TM_100, we set T=T_{max}=37.4%, and run TLDP-SPF with U_T from 0.1 to 1.0.

We found that TLD_SPF produces the same set of routes for the network with TM_10, TM_20 and TM_30 since increasing MLU from 10% to 30% only raises most traffics slightly; hence we only report the result for TM_30. We found similar situation for TM_40 to TM_60, and for TM_70 to TM_80. Note that TLD_SPF failed to save energy for TM_90 and TM_100.

Fig. 7(a) shows that the power saving with TM_30 increases from 17.26% to 19.04% when U_T is set from 0.3 to 0.5, and remains at 19.04% for U_T>0.5. However, the average delay increases to the highest level, *i.e.*, 4.5 ms, when U_T=0.4, decreases when $0.5 \leq U_T \leq 0.6$, and remains at 4.2 ms for U_T>0.6. Fig. 7(b) and (c) show the results of TM_50 and TM_70 in terms of power saving and average delay respectively; as shown, they have the same trend as TM_30. However, the latter has a U_T value, *i.e.*, 0.5 for TM_50 and 0.6 for TM_70 respectively, which means that TLDP_SPF fails to save energy for TM_30 and TM_50 when U_T is set less than its lower bound on each of the two scenarios. Notice that average delay does not decrease monotonically when U_T increases since our approach does not consider path delay when switching off links.

D. The Effects on Link Utilization and Path Delay

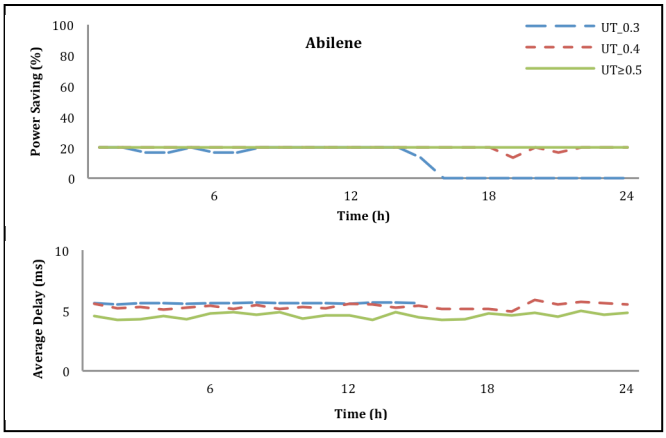
In this subsection, we show the effect of switching off links on link utilization and path delay. We compare the link utilization and delay when using TDLP-SPF against SP routing as benchmark.

The top of Fig. 8(a) plots the Cumulative Distribution Function (CDF) of link utilization for the Abilene network. We see that more than 60% of links when using SP have utilization of at most 0.1, as compared to at most 30% in TDLP-SPF; thus SP is better for this case. However, while using only 80% of total links in Abilene, each switched-on link in TDLP-SPF has utilization of no more than 0.3, which is better than when using SP, where links may reach a utilization level of 0.5. We observe a similar trend for GÉANT. As shown in the top of Fig. 8(b), almost 90% of links in SP has utilization $u_{ij} \leq 0.05$, better than only 53.68% in TLDP-SPF. However, while able to switch off 22.97% of total links in the network, 98.65% links in TLDP-SPF have $u_{ij} \leq 0.1$, better than 93.64% of all links for SP. The results for Abilene and GÉANT show that TDLP-SPF is able to distribute traffic flows more evenly than SP routing. The bottom of Fig. 8(a) and 8(b) plot the CDF of the path delay for Abilene and GÉANT, respectively. Both figures show shorter hop counts when using SP. This is expected because SP routes each traffic through its shortest path and hence is optimal in terms of path delay. Further, SP routes all traffic demands through more switched-on links, providing more options on shortest paths, than TLDP-SPF. However neither SP nor TLDP-SPF produces path longer than 6 ms.

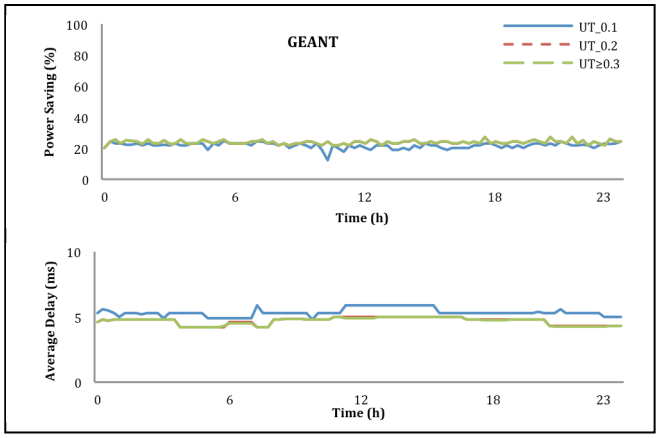
E. The Effects of Threshold T on Energy Savings

As shown in Fig. 9(a), the average energy saving on the Abilene network is 46.38%, when T is set to 0%, but reduces sharply to around 20% when $T \geq 13.3\%$. Similarly, Fig. 9(b) and Fig. 9(c) show the average energy saving on GÉANT and

Sprint decreases by more than half, from 52.7% to 23.26% and from 39.29% to 19.05% when T increases from 0% to 100% and from 0% to 37.4%, respectively. Notice that for all three networks, after a sharp decline in energy saving at the following T values, T=13.3%, T=15%, and T=8% for Abilene, GEANT, and Sprint respectively, increasing T does not significantly affect their respective energy saving. Thus, networks that support TLDP should use TLDP-SPF with the highest T possible, *i.e.*, set T=$T_{\max}$ to optimize the fault-tolerance and performance of applications.

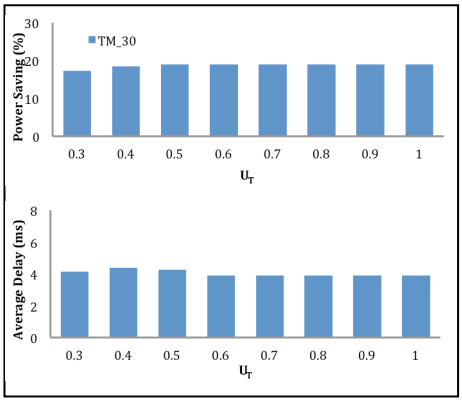


(a) Abilene (T=T_{max}=83.3%)

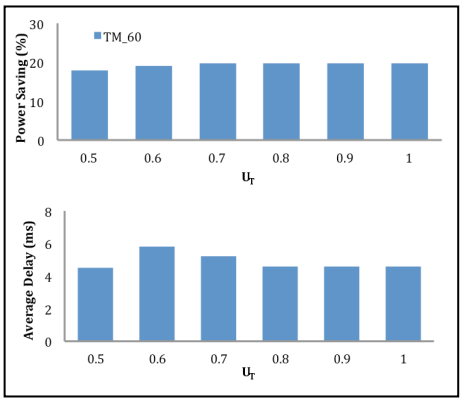


(b) GÉANT (T=T_{max}=100%)

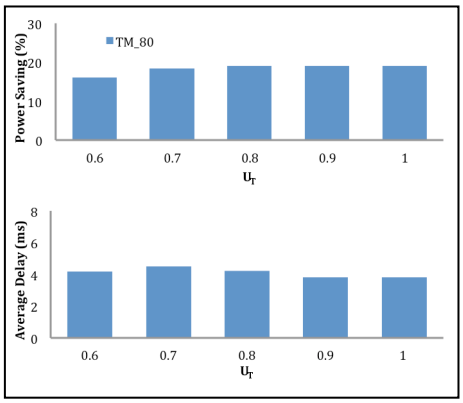
Figure 6. Energy saving and average delay on Abilene and GÉANT



(a)TM_30

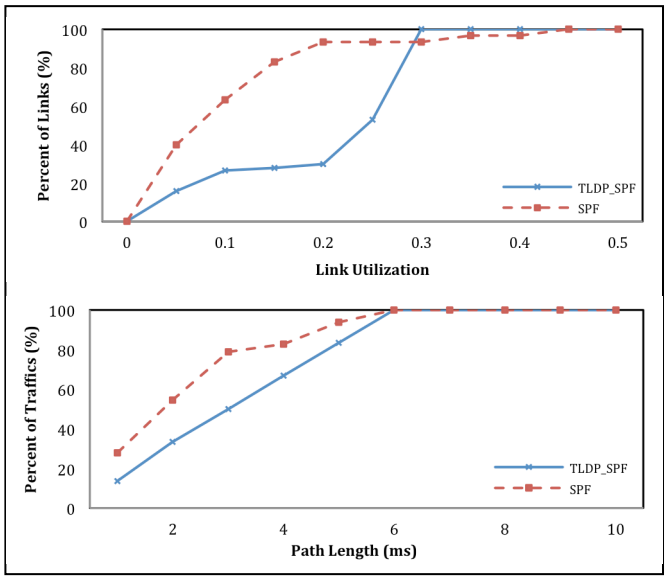


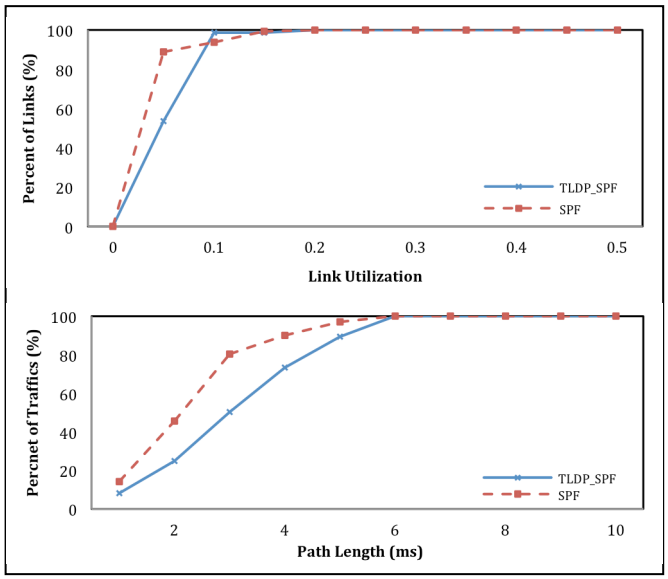
(b)TM_60



(c) TM_80

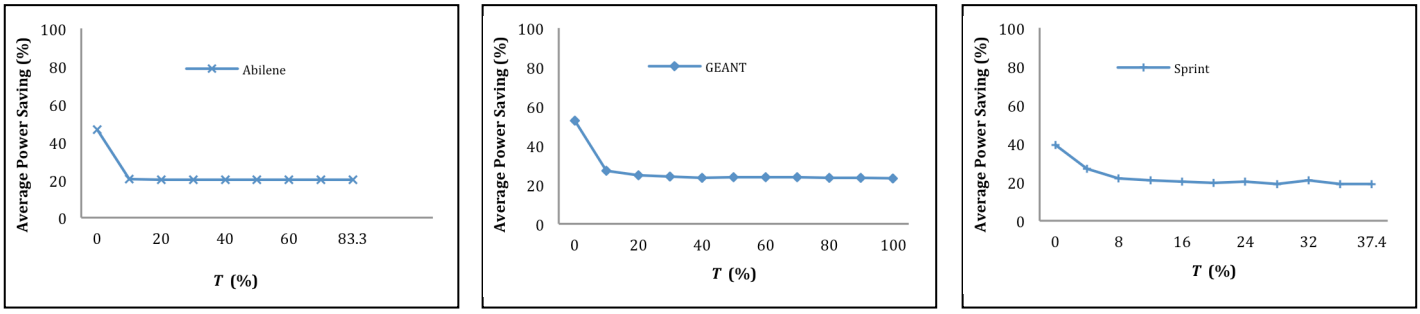
Figure 7. Energy saving and average delay on Sprint with different MLUs under SP





(a) Abilene (T=T_{max}=83.3% and U_T≥0.5) (b) GÉANT (T=T_{max}=100% and U_T≥0.3)

Figure 8. CDF of link utilization and path delay on Abilene and GÉANT



(a)Abilene (b) GÉANT (c) Sprint

Figure 9. Energy saving for different thresholds of T

V. CONCLUSION

We have presented a new energy-aware routing problem. To reduce energy expenditure, we aim to maximally switch off unnecessary links during off-peak periods such that the remaining powered on links are sufficient to route the given traffic demands under the constraints that the ratio of using TLDP is not less than a given threshold T and the maximum link utilization is not greater than a threshold U_T. We have proposed an efficient and effective heuristic technique to solve the problem. Through extensive simulations on both real and synthetic network topologies and traffic demands, we have shown its benefits in reducing a network's energy consumption.

REFERENCES

[1] K. Xiong, Z. D. Qiu, Y. Guo, and H. Zhang. "Multi-constrained shortest disjoint paths for reliable QoS routing. " ETRI Journal, 31(5): 534-544, 2009.

[2] G. Apostopoulos, R. Guerin, S. Kamat, and S.K. Tripathi. "Quality of service based routing: A performance perspective." In ACM SIGCOMM 1998, pp. 17-28.

[3] J. Chen, R. Sundaram, M. Marathe, and R. Rajaraman. "The confluent capacity of the Internet: congestion vs. dilation." In IEEE ICDCS 2006.

[4] W. Zhang, J. Tang, C. Wang, and S. D. Soysa. "Reliable adaptive multipath provisiong with bandwidth and differential delay constraints." In IEEE INFOCOM 2010.

[5] M. Kodialam, T. V. Lakshman, "Restorable dynamic quality of service routing". In IEEE Communications Magazine 2002;72‑81.

[6] Y. Guo, F. Kuipers and P. V. Meeghem. "Link-disjoint paths for reliable QoS routing." In Int. J. Commun. Syst., 26:779-798, 2003.

[7] Y. Bejerano, Y. Breitbart, A. Orda, R. Rastogi and A. Sprintson, "Algorithms for computing QoS paths with restoration". In IEEE INFOCOM 2003.

[8] M. Zhang, C. Yi, B. Liu, and B. Zhang, "GreenTE: Power-Aware Traffic Engineering". In ICNP, 2010.

[9] W. Fisher, M. Suchara, and J. Rexford, "Greening backbone networks: reducing energy consumption by shutting off cables in bundled links". In Green Networking, 2010.

[10] M. Kodialam, T. V. Lakshman, J. B. Orlin and S. Sengupta, " Pre-configuring IP-over-optical networks to handle router failures and unpredictable traffic". In IEEE INFOCOM 2006.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, "Introduction to algorithms (Second Edition)". MIT Press, 2005.

[12] J. Y. Yen, "Finding the K Shortest Loopless paths in a network". In Management Science, 17(11), 1971.

[13] Multipath Issues in Unicast and Multicast Next-Hop Selection, available at http://www.ietf.org/rfc/rfc2991.txt.

[14] "Yin Zhang's Abilene TM". http://www.cs.utexas.edu/~yzhang/ research/ AbileneTM/.

[15] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing Public Intradomain Traffic Matrices to the Research Community". ACM SIGCOMM Computer Communication Review, Vol. 36, no. 1, pp. 83-86, January 2006.

[16] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP Topologies with Rocketfuel" In IEEE/ACM Transactions on Networking, vol. 12, no. 1, pp.2-16, 2004.

[17] "Power Management for the Cisco 12000 Series Router." [Online]. Available:http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12spower.html.

[18] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran and C. Diot, "Measurement and Analysis of Single-Hop Delay on an IP Backbone Network". In IEEE Journal on Selected Areas on Communications, vol. 21, No. 6, Aug 2003.