Congestion-triggered Multipath Routing based on Shortest Path Information

Soonyong Sohn and Brian L. Mark
Dept. of Electrical and Computer Engineering
George Mason University
Fairfax, Virginia 22030
Email: {ssohn, bmark}@gmu.edu

John T. Brassil
Hewlett-Packard Laboratories
5 Vaughn Drive, Suite 301
Princeton Junction, New Jersey 08540
Email: jack.brassil@hp.com

***Abstract*— We present a multipath routing scheme that is designed to increase throughput and alleviate congestion in networks employing shortest path routing. The multipath routing scheme consists of an algorithm to determine a set of multiple disjoint or partially disjoint paths and a mechanism for distributing traffic over a multipath route to reduce the traffic load on a congested link. The algorithm for finding multipath routes is based on shortest path routing and does not require pre-establishment of paths or support for source routing. The mechanism for multipath traffic distribution is triggered at a node when the average load on an outgoing link exceeds a threshold. Our simulation results demonstrate that the proposed congestion-triggered multipath routing scheme can effectively improve network performance by exploiting routing redundancies inherent in the network topology.**

I. Introduction

Most computer networks currently employ routing protocols based on shortest path routing algorithms which determine a single path of minimum cost or length among all paths between a given pair of nodes. Representative shortest path routing protocols in the Internet include RIP [1], OSPF [2], and BGP [3]. In these routing protocols, the path length is typically taken as the hop count, i.e., the link cost is taken to be unity by default. Another link metric that is often used in conjunction with shortest path routing algorithms is link latency.

Under shortest path routing, all packets associated with a given source-destination pair generally traverse a single path of shortest length, even though other paths may be available. Consequently, shortest path routing can lead to network congestion and underutilized links. In multipath routing, packets belonging to a source-destination pair may be transmitted over multiple paths. Some of the potential benefits of multipath routing include load balancing [4], higher network throughput [4], [5], reduction of routing oscillation, the alleviation of congestion [6], [7], and improved packet delivery reliability [8].

In spite of the considerable attention that multipath routing has received in the research literature, it is rarely implemented in practical networks, most notably the Internet. Deployment of multipath routing involves two major challenges: (1) finding a suitable set of paths to form a multipath route; (2) distributing traffic over a multipath route. Finding a multipath route can be considerably more challenging than finding a shortest path. Moreover, multipath routing may require significant changes to the network routing infrastructure. Given a multipath route, the policy for traffic distribution is critical to end-to-end throughput performance. In fact, it is well-known that multipath routing can lead to *worse* performance than the conventional shortest path routing.

The main contributions of this paper are two-fold: (1) a simple algorithm to determine multipath routes, which can be implemented on top of a shortest path routing algorithms without major changes to the network infrastrcture; (2) a mechanism to distribute traffic over a multipath route based on link utilization measurements. The proposed algorithm for finding multiple paths between a source and destination node involves examinining and classifying the set of shortest paths to the destination from the neighbors of the source node. Ideally, the set of paths in a multipath route should be link disjoint to maximize routing diversity, but partially disjoint paths may be sufficient for the purposes of congestion avoidance. Multipath routing is triggered at a node when the average utilization of an outgoing link exceeds a certain threshold. Traffic is then distributed over a multipath route to reduce the load on this particular link.

The remainder of this paper is organized as follows. Section III develops an algorithm for finding multiple paths between a given pair of nodes, based on shortest path routing information. Section IV discusses a mechanism for congestion-triggered multipath traffic distribution. Section V presents simulation results showing the performance of the congestion-triggered multipath routing scheme in light vs. heavy traffic scenarios. Finally, Section VI concludes the paper.

II. Background and Related Work

A. Basic concepts from graph theory

A network can be represented by a directed graph $G = (V, E)$ with node set V and link or edge set E. The number of nodes and links in the network are denoted by $|V|$ and $|E|$, respectively. A link in E between nodes i and j is denoted as an ordered pair (i, j), where i is referred to as the *tail* of the link and j is the *head* of the link. Link (i, j) has an associated positive *cost* or *length* $D(i, j)$.

A *path* p is a sequence of nodes such that from each node in the path, there is a link to the next node in the sequence. The first node in the sequence is called the *source* node and the last node is called the *destination* node. The remaining nodes are known as *intermediate* nodes. As an example, the path

$$p = \{s, i_1, i_2, \cdots, i_n, d\}$$

consists of the source node s, intermediate nodes i_1 through i_n, and the destination node d. The path p can also be represented as a sequence of links as follows:

$$L(p) = \{(s, i_1), (i_1, i_2), \cdots, (i_{n-1}, i_n), (i_n, d)\},$$

where $L(p)$ is called the *link representation* of the path p. A *cycle* or *loop* is a path such that source and destination are the same. A path with no repeated nodes is called a *simple* or *loop-free* path.

Two paths are said to be *link disjoint* or simply *disjoint* if the link representations of the paths are disjoint, i.e., the two paths do not share a common link. A *multipath route* is a set of paths, each of which has the same source and destination node. We also refer to each path in a multipath route as an *alternative* path. In a network, a packet sent from the source node on any of the alternative paths will arrive at the same destination node. For a multipath route, link disjoint paths are desirable because the traffic distributed over the alternative paths in the multipath route do contend for common network resources.

B. Finding multipath routes

A number of algorithms have been proposed in the literature to find disjoint paths between source and destination nodes in a network [8]–[12]. The shortest pairs of disjoint paths problem (SPDP) can be defined as follows: Given a destination node d and for each node $s \neq d$, find a pair of disjoint paths from s to d of minimum total length. Ogier et al. [10] present a distributed algorithm to solve SPDP by reducing the problem to a shortest path problem on a modified graph. Most of the existing algorithms to find disjoint paths have significant communication and time complexity requirements and cannot easily be bootstrapped onto an existing shortest path routing infrastructure. Moreover, forwarding packets over a set of disjoint paths generally requires the support of source routing [13] or the pre-establishment of switched paths as in ATM (Asynchronous Transfer Mode) or MPLS (Multi-Protocol Label Switching). An interesting alternative approach for packet forwarding over disjoint paths is proposed in [10], which incurs significantly less overhead than source routing, but still requires modifications to the routing infrastructure.

Another approach to finding multipath routes is to exploit shortest path information derived from a shortest path routing protocol [14]–[19]. In this approach, the set of paths in a multipath route includes the shortest path, obtained from the shortest path routing protocol, plus alternative paths derived from shortest paths from each of the neighbors of the source node to the destination node. Here, the set of paths in the multipath route is not guaranteed to be disjoint. An alternative path via a neighbor node j is included in the multipath route only if the length of the shortest path from j to the destination node is strictly less than the length of the shortest path between the source and destination. Application of this rule avoids the formation of routing loops in the alternative path and ensures that the length of the alternative path is not significantly greater than that of the shortest path. In this paper, we propose an algorithm for finding multipath routes based on shortest path routing information, but, unlike the earlier approaches, it does not require the path from the neighbor node to the destination to be less than the length of the shortest path between the source and the destination. Thus, a larger set of paths is considered for possible inclusion in the multipath route.

C. Multipath traffic distribution

The Equal-Cost MultiPath (ECMP) protocol [2] is a multipath routing extension for Internet routing protocols such as OSPF and RIP [1]. In ECMP, a node implements multipath routing when it discovers two or more shortest paths (of equal length) to a destination node. These paths can be determined via relatively simple extensions of standard shortest path algorithms such as Dijkstra's algorithm or the Bellman-Ford algorithm. The set of paths making up a multipath route need not be disjoint. Under ECMP, once a multipath route is discovered, packets are forwarded in equal proportion over the set of paths in the multipath route. There are several drawbacks of this approach: (1) ECMP is not guaranteed to determine a multipath route for each source-destination pair; (2) The characteristics of the multipath route are not taken into account; (3) Packets are forwarded in equal proportion, on a packet-by-packet basis, over the paths in the multipath route, without considering network congestion. As a result, packets may arrive out-of-order within a flow at the destination. Moreover, ECMP may actually cause more congestion than single path routing in some scenarios.

The Optimized MultiPath (OMP) protocol [20] is an improved version of ECMP for link state routing protocols such as OSPF, which allows unequal traffic distribution. Multipath routes are found using the approach based on shortest path routing. The paths in the multipath route need not be of equal length. Traffic is distributed over a multipath route in inverse proportion to the utilizations of the constituent paths. Path utilization information is inferred from link state information. Unlike ECMP, OMP ensures that packets belonging to a flow are always forwarded on the same path. Thus, traffic is distributed over a multipath route at the granularity of a flow, which avoids the out-of-order packet problem of ECMP. Since OMP is triggered by path utilization information, multiple nodes sharing common subpaths may simultaneously begin distributing traffic over multipath routes sharing common links. Although congestion may be avoided over the original path that triggered the OMP protocol, other paths may become congested as an unwanted side effect of the OMP traffic distribution policy. This could lead to the triggering of OMP at further nodes, which may eventually result in network instability.

III. Multipath Routes Based on Shortest Path Routing

In this section, we describe a new algorithm to find a set of paths forming a multipath route, assuming an underlying shortest path routing infrastructure.

A. Shortest path routing

For the network $G = (V, E)$, a shortest path routing algorithm determines a unique *shortest path*, p_{sd}, from each node s to every other node d in the network. Shortest paths determined by a shortest path routing algorithm possess the following important property [21].

Definition 1: Shortest path property: Let p_{sd} be the shortest path in G from node s to node d. Then any subpath of p_{sd} is also the shortest path between its two end nodes.

Consider the shortest path p_{sd} from node s to node d, determined by a routing algorithm. Let $\mathcal{N}(s)$ denote the set of neighbor nodes of node s. To obtain alternative paths from s to d, we consider paths of the form

$$p_{sd}^j = \{s\} \oplus p_{jd}, \tag{1}$$

where '$\oplus$' denotes the concatenation of two (finite) sequences. Thus, the path p_{sd}^j begins at node s, proceeds to neighbor node j, and then follows the shortest path route p_{jd} from node j to node d.

However, p_{sd}^j might form a routing loop by having node s as a intermediate node. To prevent routing loops, we impose a condition on establishing p_{sd}^j as stated in the following lemma[1].

Lemma 3.1: If $s \notin p_{jd}$ then p_{sd}^j is loop-free.

We define a set, $\mathcal{A}_{sd}$, of alternative paths from node s to node d via neighbor nodes as follows:

$$\mathcal{A}_{sd} = \{p_{sd}^j : j \in \mathcal{N}(s), s \notin p_{jd}\}. \tag{2}$$

Thus, $\mathcal{A}_{sd}$ is the set paths from s to d via a neighbor node j that is not in the path p_{sd}. To form a multipath route from node s to d, we shall consider the set of alternative paths in the set $\mathcal{A}_{sd}$. To make use of an alternative path $p_{sd}^j \in \mathcal{A}_{sd}$, node s merely forwards a packet to node j. Under conventional single path routing, node j will then forward the packet along the shortest path p_{jd} to the destination node d.

An important consequence of the shortest path property is:

Proposition 3.2: Two alternative paths from a source node s to a destination node d are disjoint if and only if the only common nodes are s and d, i.e., the paths do not have any common intermediate nodes.

In particular, any two paths in $\mathcal{A}_{sd}$ are disjoint if and only if the only common nodes are s and d.

[1]For brevity, we omit all proofs in this paper. For further details, the reader is referred to [22], available from the authors

B. Disjoint alternative paths

To maximize routing diversity and resilience, it is often desirable to form a multipath route on a set of pairwise disjoint. Define the set of paths

$$\mathcal{P}_{sd} = \{p_{sd}\} \cup \mathcal{A}_{sd}.$$

The following lemma gives a condition for two shortest paths to a common destination node to be disjoint.

Lemma 3.3: Consider two shortest paths p_{sd} and p_{jd} from two distinct nodes s and j, respectively, to a common destination node d. If the two paths do not have a common penultimate (second to last) hop k, then they must be disjoint. Conversely, if paths p_{sd} and p_{jd} are not disjoint, they must share a common subpath p_{ad}, which is a shortest path from node a to d.

Corollary 3.4: Consider path p_{sd} and path $p_{sd}^j \in \mathcal{A}_{sd}$. If p_{sd} and path p_{sd}^j do not share a common penultimate hop k, then they must be disjoint.

Using Lemma 3.1 and Corollary 3.4, we obtain the following results.

Proposition 3.5: Let p_{sd} be the shortest path from node s to node d and let p_{jd} be the shortest path from node j to node d. If p_{sd} and p_{jd} are disjoint, then p_{sd} and p_{sd}^j are disjoint and p_{sd}^j is loop-free.

Proposition 3.6: Consider paths $p_{sd}^j, p_{sd}^l \in \mathcal{A}_{sd}$, where $j \neq l$. If p_{sd}^j and p_{sd}^l do not share a common penultimate hop, then p_{sd}^j and p_{sd}^l are disjoint.

Proposition 3.5 is used to choose a path p_{sd}^j that is loop-free and disjoint from the shortest path p_{sd}. Proposition 3.6 is used to ensure that two alternative paths p_{sd}^j and p_{sd}^l are disjoint from each other.

C. Class-c paths

To form a multipath routes, we shall define a set $\mathcal{P}_{sd}^{(c)}$ of *class-c paths* between s and d, which have the property that any two paths $p, q \in \mathcal{P}_{sd}^{(c)}$ have at most c common links, i.e.,

$$|L(p) \cap L(q)| \leq c. \tag{3}$$

The class $\mathcal{P}_{sd}^{(c)}$ is defined by Algorithm 1. In particular, the set of class-0 paths consists of pairwise disjoint paths from s to d. The set of class-1 paths has the property that every pair of paths shares at most a single common link, i.e., a link (k, d), where k is a neighbor of node d. The set of class-2 paths has the property that every pair of paths shares at most two common links, say (k, d) and (k', k), which form a subpath $\{k', k, d\}$. The sets of class-3, class-4, etc., can be characterized similarly. Note that the set of class-c paths is contained in the set of class-$(c+1)$ paths for $c = 0, 1, 2, \cdots$. Thus, we have

$$P_{sd}^{(0)} \subseteq P_{sd}^{(1)} \subseteq P_{sd}^{(2)} \subseteq \cdots \subseteq P_{sd}^{(c)} \subseteq P_{sd},$$

for $c > 2$.

In Algorithm 1, the set $P_{sd}^{(c)}$ is initialized to contain the shortest path p_{sd}. The other paths in $P_{sd}^{(c)}$ are selected from the set of alternative paths $\mathcal{A}_{sd}$ in increasing order of path

Network Topology	Parameters			Class-c paths		
	N	L	d	$c=0$	$c=1$	$c=2$
Six-node	6	16	2.66	1.93	2.13	2.20
Smallnet	10	44	4.4	2.77	3.78	3.88
LATA	11	46	4.18	2.38	3.37	3.58
NSFNET	14	42	3	1.89	2.12	2.23
Citi Multi-ring	15	40	2.67	1.38	1.67	1.83
Bellcore	15	54	3.6	1.72	2.43	2.73
EON	19	64	3.89	1.76	2.61	3.07
ARPA	20	62	3.1	1.66	1.90	2.12
ARPA2	21	50	2.38	1.31	1.41	1.46
US IP backbone	24	86	3.58	1.59	2.05	2.38
Average	15.5	51.4	3.346	1.84	2.35	2.55

TABLE I

NUMBER OF CLASS-c PATHS FOR DIFFERENT NETWORK TOPOLOGIES.

length. The **while** loop iterates over all paths in the set $\mathcal{A}_{sd}$. In lines 4 and 5, the shortest path $\tilde{p}$ in the (current) set $\mathcal{A}_{sd}$ is removed from the set. In lines 6-13, the candidate path $\tilde{p}$ is tested against all of the paths in the (current) set $\mathcal{P}_{sd}^{(c)}$ to check whether the condition (3) is satisfied. If $\tilde{p}$ satisfies (3) for all paths $p \in \mathcal{P}_{sd}^{(c)}$, then $\tilde{p}$ is added to the set $\mathcal{P}_{sd}^{(c)}$ (lines 14-16). The computational complexity of Algorithm 1 is $O(|\mathcal{N}(s)|^2)$ where $|\mathcal{N}(s)|$ is the number of neighbors of node s.

Algorithm 1 Finding class-c paths from s to d.

1: **Input:** $\mathcal{A}_{sd}$, c, s, d; **Output:** $\mathcal{P}_{sd}^{(c)}$
2: $\mathcal{P}_{sd}^{(c)} \leftarrow \{p_{sd}\}$
3: **while** $\mathcal{A}_{sd} \neq \emptyset$ **do**
4: $\quad \tilde{p} \leftarrow \text{argmin}\{D(q) : q \in \mathcal{A}_{sd}\}$
5: $\quad \mathcal{A}_{sd} \leftarrow \mathcal{A}_{sd} \backslash \{\tilde{p}\}$
6: $\quad$ size $\leftarrow |\mathcal{P}_{sd}^{(c)}|$
7: $\quad$ **for** each $p \in \mathcal{P}_{sd}^{(c)}$ **do**
8: $\quad\quad$ **if** $|L(\tilde{p}) \cap L(p)| \leq c$ **then**
9: $\quad\quad\quad$ size $\leftarrow$ size - 1
10: $\quad\quad$ **else**
11: $\quad\quad\quad$ **break**
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: $\quad$ **if** size = 0 **then**
15: $\quad\quad \mathcal{P}_{sd}^{(c)} \leftarrow \mathcal{P}_{sd}^{(c)} \cup \{\tilde{p}\}$
16: $\quad$ **end if**
17: **end while**

We applied Algorithm 1 to various network topologies (see [22]). The results are shown in Table I. Each row in the table indicates the average number of class-0, class-1, and class-2 paths found by the Algorithm 1 for a given network topology. Each network topology is characterized by the number of nodes N, the number of (unidirectional) links L, and the average node degree d. The bottom row of the table shows the average numbers of class-c paths averaged over all ten topologies. Since the average number of class-1 paths is 2.35, a given node has, on average, at least two class-1 paths to every other node.

IV. CONGESTION-TRIGGERED MULTIPATH ROUTING

The basic idea of our proposed *Congestion-Triggered MultiPath routing (CTMP)* scheme is that when a node s detects congestion on local link (i.e., link (s, j)), it distributes traffic over class-c paths according to link utilization LU and path utilization PU so as to resolve the local link congestion. We first describe a method to exchange routing and network information at section IV-A and then discuss an approach to alleviate network congestion at section IV-B.

A. Routing and Network information exchange

We modify Path Vector (PV) routing to compute class-c paths and to include additional congestion-related information in the routing control messages. PV routing is similar to Distance Vector (DV) routing, except that the shortest path information is maintained along with the distance to each destination. In addition to the path vector for PV routing, the CTMP scheme requires the storage and exchange of additional network state information as follows:

- ECI (Explicit Congestion Indication) and MPI (Multipath Indication).
- utilization $PU(p)$ and capacity $PC(p)$ along path p,

The ECI and MCI bits are one-bit flags stored in the routing table for each destination. The ECI bit is set to 0 by default to indicate no congestion and 1 to indicate congestion on the path to the destination. The MPI bit has a default value of 0 and is set to 1 if a class-c multipath route is established to the corresponding destination. We assume that each node s in network estimates the utilization, $LU(l)$, of each local outgoing link l. Let $LC(l)$ denote the capacity of link l. The capacity and utilization of path p are defined, respectively, as

$$PC(p) = \min\{LC(l) : l \in p\}, \tag{4}$$

$$PU(p) = \min\{LU(l) : l \in p\}. \tag{5}$$

The path capacity and utilization can be computed by a given node in a similar way as the path vector in PV routing. For example, a given node s estimates $PC(p_{sd})$ and $PU(p_{sd})$ by using routing/network information sent by its neighbors as follows:

$$PC(p_{sd}) = \min\{LC(s, j), PC(p_{jd})\}, \tag{6}$$

$$PU(p_{sd}) = \min\{LU(s, j), PU(p_{jd})\}. \tag{7}$$

Thus, node s does not need to exchange its local link capacity and utilization information to every other node in the network.

B. Local congestion trigger and multipath traffic distribution

We develop an approach to resolve the network congestion by distributing the traffic over class-c multipath routes obtained using Algorithm 1. When node s detects congestion on a local outgoing link l, it computes multipath routes to destinations d for which the path p_{sd} contains link l. Some portion of the traffic to node d is then shifted to alternative paths $p \in \mathcal{P}_{sd}^{(c)}$. Congestion is detected on a local link if its utilization exceeds a *local congestion threshold* β, e.g., $\beta = 95\%$. The objective is to decrease the utilization to a more acceptable level by

shifting a portion of the traffic to the alternative paths. We refer to this portion of traffic as *detour* traffic.

Whenever a node detects local link congestion or receives an ECI bit from a neighbor, it computes a set of alternative paths and distributes the detour traffic over these paths. If a node cannot resolve the congestion in this way, it signals its neighbors using the ECI bit. The procedure for congestion-triggered multipath traffic distribution to resolve congestion at the *local link* is given as follows:

1) When node s detects local congestion on outgoing link l (i.e., $PU(l) > \beta$), or receives an ECI=1 bit from a neighbor node on link l, it tries to move detour traffic onto alternative paths that avoid this link. The alternative paths are class-c paths determined using Algorithm 1.
2) Let $\mathcal{M} = \{p \in \mathcal{P}_{sd}^{(c)} : PU(p) < \gamma\}$. The parameter γ is called the *path availability threshold*.
3) Distribute traffic over the path set $\mathcal{M}$ according to the *traffic splitting function* $\phi(\cdot)$ (see below).
4) Send ECI to neighbor nodes if the congestion cannot be resolved by Step 3.

When congestion is detected on link l in Step 1, node s considers all of its paths which contain link l. Node s also generates the set of class-c paths, $\mathcal{P}_{sd}^{(c)}$, to a given destination node, in case these paths have not already been computed (i.e., the MPI bit is set to zero). Let $p \in \mathcal{P}_{sd}^{(c)}$ be a path that contains link l. The traffic splitting function is defined as a mapping $\phi : \mathcal{P}_{sd}^{(c)} \rightarrow [0, 1]$ as follows:

$$\phi(p) = \frac{LC(l) \cdot \eta}{\sum_{q \in \mathcal{P}_{sd}^{(c)} \setminus \{p\}} PC(q)[\eta - PU(q)]^+ + LC(l) \cdot \eta}, \quad (8)$$

and for $\tilde{p} \in \mathcal{P}_{sd}^{(c)} \setminus \{p\}$,

$$\phi(\tilde{p}) = \frac{PC(\tilde{p})[\eta - PU(\tilde{p})]^+}{\sum_{q \in \mathcal{P}_{sd}^{(c)} \setminus \{p\}} PC(q)[\eta - PU(q)]^+ + LC(l) \cdot \eta}, \quad (9)$$

where $[x] \triangleq \max\{0, x\}$ and η is a parameter called the *target path utilization*. The traffic splitting function $\phi(\cdot)$ is a probability function over the set of paths $p \in \mathcal{P}_{sd}^{(c)}$, i.e.,

$$\sum_{p \in \mathcal{P}_{sd}^{(c)}} \phi(p) = 1.$$

The three parameters β, η, and γ are related as follows:

$$0 < \gamma \leq \eta < \beta < 1.$$

V. Simulation Results

The performance of CTMP is compared using simulation with that of conventional PV routing and ECMP. As in ECMP, the CTMP scheme routes multipath traffic at the packet level. Routing at the flow level could be carried out by means of hash functions as in OMP [20], but is not considered here. The simulations are carried out using the ns-2 network simulator on the NSFNET network topology shown in Fig. 1. The topology consists of 14 different nodes and 42 unidirectional links, each with a capacity of 45 Mbps.

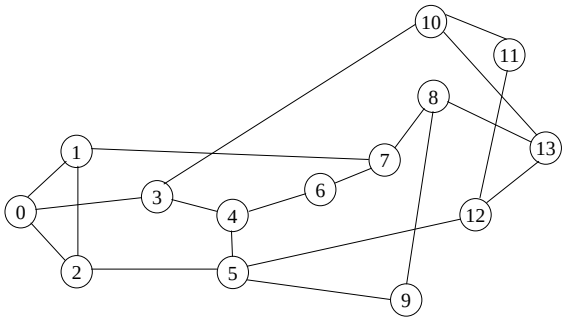

Fig. 1. NSFNET network topology.

link	PV	ECMP	CTMP
(5,2)	65.10	57.18	65.10
(7,8)	56.42	66.52	56.42
(8,7)	56.42	66.49	56.42
(2,5)	60.76	57.15	60.76
(3,10)	60.76	56.42	60.76
(10,3)	60.76	56.42	60.76
(4,5)	60.76	52.08	60.76
(5,4)	56.42	52.11	56.42
(5,12)	52.08	56.42	52.08
(12,5)	52.08	56.35	52.08
Average	40.30	40.30	40.30

TABLE II

Top 10 link utilizations under light traffic.

To evaluate the network performance, we consider link utilizations and the number of packet drops under two traffic scenarios: light traffic and heavy traffic. The light traffic scenario does not cause the network to be congested, whereas the heavy traffic scenario causes a significant degree of congestion in the network. For the CTMP scheme, the parameters β, η, and γ are set to 95%, 90%, and 90%, respectively.

A. Light traffic scenario

In the light traffic scenario, 2 Mbps traffic flows are sent from each node to every other node. Table II shows the top 10 link utilizations, as well as the average link utilizations over all links. As can be seen in Table II, the highest link utilization is 65.1% on link (5,2) under PV routing, 66.52% on link (7,8) under ECMP, and 65.1% on link (5,2) under CTMP. In this scenario, there are no packet drops on any of the links. Thus, we see that there is no congestion. Also, we can see that the average link utilizations for the three routing schemes are the same. Since there is no congestion to trigger multipath routing in CTMP, the performance of CTMP is the same as that of PV routing. On the other hand, we can see that ECMP provides different link utilizations from PV routing and CTMP, since it applies equal cost multipath routing [2].

B. Heavy traffic scenario

In the heavy traffic scenario, each node transmits at a rate of 3 Mbps traffic to every other node in order to cause network congestion on some links. From Table III and IV, we can see that there is congestion on link (5,2) under PV routing

link	PV	ECMP	CTMP
(5,2)	100	87.75	90.46
(7,8)	86.72	100	86.72
(8,7)	86.65	100	90.96
(2,5)	93.37	87.78	93.37
(3,10)	93.37	86.64	93.37
(10,3)	93.37	86.72	93.37
(4,5)	93.37	80.03	93.37
(5,4)	86.72	80.03	90.53
(5,12)	80.07	86.75	82.19
(12,5)	80.07	86.68	80.07
Average	61.95	61.78	61.96

TABLE III

TOP 10 LINK UTILIZATIONS UNDER HEAVY TRAFFIC.

link	PV	ECMP	CTMP
(5,2)	48	0	0
(7,8)	0	131	0
(8,7)	0	126	0
Others	0	0	0

TABLE IV

NUMBER OF PACKET DROPS UNDER HEAVY TRAFFIC SCENARIO.

and both on link (7,8) and link (8,7) under ECMP. While PV routing causes only one link to be congested, ECMP induces congestion in two links. ECMP employs multipath routing but distributes traffic equally over a multipath route, without taking into account link condition. This results in a poor traffic distribution, which causes other links to be congested. As traffic volume increases, the network congestion can become more severe under ECMP.

The CTMP scheme reduces the traffic volume through the congested links by distributing traffic over class-1 multipath routes. This action is triggered only when the local link is congested. Under CTMP, the utilization of link (5,2) is greater than β so that node 5 establishes a multipath route for any traffic flowing through link (5,2) and distributes the traffic over class-1 multipath routes according to the traffic splitting function $\phi(\cdot)$ as given by (9) and (8).

As can be observed from Tables III and IV, the CTMP scheme alleviates network congestion by balancing the traffic load over multipath routes. Note from Table III that the average link utilizations for the three different schemes are different from each other. This is due to the packet drops that occur on congested links. By alleviating network congestion, CTMP is able to improve the overall network performance.

VI. CONCLUSION

We propose a multipath routing scheme consisting of two components: (1) an algorithm to determine multipath routes over shortest path routing; (2) a congestion-triggered scheme to distribute traffic over a multipath route to avoid network congestion. The multipath route finding scheme does not require the pre-establishment of paths or source routing. Our simulation results demonstrate the ability of the multipath routing scheme to relieve network congestion and improve overall network utilization. In ongoing work, we are evaluating the congestion-triggered routing multipath routing algorithm under more complex network and traffic scenarios.

ACKNOWLEDGEMENT

This work was supported in part by DARPA Contract N66001-05-9-8904 and by the National Science Foundation under Grant CCF-0133390.

REFERENCES

[1] G. Malkin, "RIP version 2 protocol analysis," *IETF Internet RFC 1721*, November 1994.
[2] J. Moy, "OSPF version 2," *IETF Internet RFC 2328*, 1998.
[3] P. Traina, "BGP-4 protocol analysis," *IETF RFC Internet 1774*, October 1995.
[4] H. Suzuki and F. A. Tobagi, "Fast bandwidth reservation scheme with multi-link and multi-path routing in ATM networks," in *Proc. IEEE INFOCOM*, 1992.
[5] R. Krishnan and J. Silvester, "Choice of allocation granularity in multi-path source routing schemes," in *Proc. IEEE Infocom*, vol. 1, 1993, pp. 322–329.
[6] S. Bahk and M. E. Zarki, "Dynamic multi-path routing and how it compares with other dynamic routing algorithms for high speed wide area networks," in *Proc. ACM SIGCOMM*, 1992, pp. 53–64.
[7] P. Georgatsos and D. Griffin, "A management system for load balancing through adaptive routing in multiservice ATM networks," in *Proc. IEEE Infocom*, 1996, pp. 863–870.
[8] K. Ishida, Y. Kakuda, and T. Kikuno, "A routing protocol for finding two node-disjoint paths in computer networks," in *Proc. IEEE International Conference on Network Protocols (ICNP)*, 1992, pp. 340–347.
[9] D. Sidhu, R. Nair, and S. Abdallah, "Finding disjoint paths in networks," in *Proc. ACM SIGCOMM*, August 1991, pp. 885–896.
[10] R. Ogier, V. Rutenburg, and N. Shacham, "Distributed algorithms for computing shortest pairs of disjoint paths," *IEEE Transactions on Information Theory*, vol. 39, pp. 443–455, March 1993.
[11] D. Eppstein, "Finding the k shortest paths," in *Proc. Foundations of Computer Science*, November 1994, pp. 154–165.
[12] R. Bhandari, "Optimal physical diversity algorithms and survivable networks," in *Proc. Second IEEE Symposium on Computers and Communications*, Alexandria, Egypt, July 1997, pp. 433–441.
[13] D. Estrin, T. Li, Y. Rekhter, K. Varadhan, and D. Zappala, "Source demand routing: Packet format and forwarding specification," *IETF Internet RFC 1940*, May 1996.
[14] S. Murthy and J. Garcia-Luna-Aceves, "Congestion-oriented shortest multipath routing," in *Proc. IEEE Infocom*, March 1996.
[15] W. Zaumen and J. J. Garcia-Luna-Aceves, "Loop-free multipath routing using generalized diffusing computations," in *Proc. IEEE Infocom*, March 1998.
[16] S. Vutukury and J. Garcia-Luna-Aceves, "An algorithm for multipath computation using distance-vectors with predecessor information," in *Proc. IEEE IC3N*, Boston, Massachusetts, October 1999.
[17] P. Narvaez and K. Y. Siu, "Efficient algorithms for multi-path link state routing," in *Proc. ISCOM*, Kaohsiung, Taiwan, 1999.
[18] S. Vutukury and J. Garcia-Luna-Aceves, "MDVA: A distance-vector multipath routing protocol," in *Proc. IEEE Infocom*, Anchorage, Alaska, USA, April 2001, pp. 319–327.
[19] ——, "MPATH: a loop-free multipath routing algorithm," *Journal of Microprocessors and Microsystems*, pp. 319–327, 2001.
[20] C. Villamizar, "OSPF Optimized MultiPath," *IETF Internet draft*, February 1999.
[21] R. K. Ahuja, T. L. Magnatti, and J. B. Orlin, *Network flows*. Prentice Hall, Englwood Cliffs, 1993.
[22] S. Sohn, B. L. Mark, and J. T. Brassil, "Congestion-triggered multipath routing based on shortest path information," Network Architecture and Performance Laboratory, Dept. of Electrical and Computer Enginering, George Mason University, Fairfax, VA, Tech. Rep., July 2006.