MCPR: Routing Using Parallel Shortest Paths

Ahmet Soran, Murat Yuksel, and Mehmet Hadi Gunes

***Abstract*—Recent trends led to higher data volumes to be transferred and processed over the network. Legacy routing protocols, e.g., OSPF for intra-domain routing, send data from a source to destination on one of the shortest paths. We propose a novel approach to parallelize data transfers by leveraging the multi-core CPUs in the routers. We describe an end-to-end method to optimize data flows on multiple paths. Multi-core parallel routing (MCPR) generates new virtual topology substrates from the underlying router topology and performs the shortest path routing on each substrate. Even though calculating the shortest paths could be done with well-known techniques such as OSPF's Dijkstra implementation, finding optimal substrates and setting their link weights to maximize the network throughput over multiple end-to-end paths is still an NP-hard problem. In MCPR, we focus on designing heuristics for substrate generation from a given router topology. Each substrate is a subgraph of the router topology and each link on each substrate is to be assigned a weight to steer the shortest-path routing for maximal network throughput. Heuristics' interim goal is to generate substrates in such a way that the shortest path between a source-destination pair on each substrate minimally overlaps with the shortest paths calculated by the other substrates. Once these substrates are determined, we assign each substrate to a core in the router and employ a multi-path transport protocol, similar to MPTCP, to perform end-to-end data transfers. We designed heuristics that utilize node centrality, edge centrality, or flow patterns. We evaluated the MCPR heuristics on router-level ISP topologies and compared the network throughput against single shortest-path routing under extensive simulation scenarios including heterogeneous core count across the routers and network failures. The evaluations showed that MCPR heuristics can attain network throughput speedups reaching 2.6 while incurring only polynomial control overhead.**

***Index Terms*—Big data, load balancing, multi-core routers, network centrality, network management, routing protocols.**

I. Introduction

THE amount of data to be *processed* at computers and datacenters as well as *transferred* across the Internet are growing at an immense pace. 100 to 400 Gbps speeds for inter- and intra-datacenter transfers are becoming the norm [3]. As the datacenters keep their central role in the Internet's big data traffic, effective and efficient management and routing of big data transfers at large speeds across datacenters are critical challenges. The legacy end-to-end (e2e) transfer techniques are not adequate to reach such speeds of 100–400 Gbps [4]–[6]. This led to design of new e2e transport mechanisms that can leverage multiple paths, i.e., multi-path TCP (MPTCP) [7]–[9]. The basic concept is to split the e2e flow into parallel subflows to better utilize the underlying network paths. If these parallel data transfers can be spread in a non-overlapping manner, they can improve the aggregate throughput. Availability of such e2e paths allows parallel TCP streams or subflows to be fed onto different paths and thereby attain a higher utilization of the underlying network capacity which is not possible with legacy single path shortest-path routing algorithms [10], [11]. Hence, in order to successfully increase network throughput, these parallel transfers require *multi-path routing capability*.

The key focus of multi-path routing techniques is to diversify and spread the paths available to the e2e transport while satisfying constraints such as delay or loss. Since the problem is complex, most multi-path routing work resort to heavy precomputations of paths as an approach to find non-shortest paths. Although these multi-path routing techniques proved to be useful for scaling up the e2e reliable transfers, they require considerable updates to legacy routers which are designed and optimized for shortest path computations. Practical protocols that can offer multiple non-shortest paths while effectively handling *network dynamics and failures* are still of high need. Existing e2e transfer methods and legacy multi-path routing schemes are still yet to utilize multi-core CPUs available in most routers. Although, [12] shows that CPU utilization reach to 90% under heavy data transfers, consideration of multiple cores of the router CPUs as a first-class citizen in the network layer control-plane functions, like routing, has been missing.

In this paper, we propose a multi-path routing framework, multi-core parallel routing (MCPR), that explicitly considers *multi-core routers* in design while utilizing the highly optimized legacy *shortest-path calculations*. MCPR leverages the advent of multi-core CPUs in routers and eases the computational complexities of close-to-optimal multi-path routing algorithms *by dividing the overall multi-path routing problem into smaller parts and lending each to a separate CPU core*. The basic idea is to virtually slice the router topology into "substrate" topologies and assign them to separate router cores, which calculate the classical shortest paths on the assigned substrate.

Rather than solving the multi-path routing problem all at once, MCPR transforms it into two subproblems: (i) slicing out substrates from the router topology so that the collection of the shortest paths on each substrate is diverse and non-

Manuscript received December 26, 2023; approved for publication April 4, 2024; approved for publication by Han, Zhu Division 3 Editor, May 12, 2024.

This work was supported in part by U.S. National Science Foundation awards 1814086 and 2346681.

A. Soran is with TRK Technology, Turkey, email: sorantr@gmail.com.

M. Yuksel is with Electrical and Computer Engineering, University of Central Florida, Orlando, FL, USA, email: murat.yuksel@ucf.edu.

M. H. Gunes is with Akamai Technologies, New York City Area, USA, email: mgunes@stevens.edu.

M. Yuksel is the corresponding author.

Preliminary versions of this work appeared in [1], [2]. This paper extends the conference publication by adding (i) a theoretical analysis of the substrate generation problem, (ii) significantly more simulation results and comparisons for different removal techniques, (iii) results showing the effects of heterogeneous core deployments, and (iv) an exploration of performance against edge/node failures.

Digital Object Identifier: 10.23919/JCN.2024.000026

1229-2370/24/$10.00 © 2024 KICS

overlapping, and (ii) calculating shortest paths on each substrate. Since the latter problem is already handled in legacy routers, MCPR can easily be adapted to current routers if the former problem is solvable. From one point of view, our approach transforms the multi-path routing problem into a topology/substrate generation problem. Main contributions of our work include:

- An effective multi-path routing framework that utilizes shortest-path routing and multiple virtual topologies, which we call substrate topologies/graphs. The framework explicitly recognizes the multi-core routers and envisions assignment of multiple shortest-path routing daemons to separate cores on the routers.
- A novel divide-and-conquer approach to the multi-path routing problem that effectively transforms the multi-path routing problem to a substrate topology generation problem, which is aligned with multi-topology routing standards and can be deployed with minimal disruption to legacy protocols. The approach generates multiple substrate networks that are maintained in parallel and there is a shortest-path routing protocol running on each.
- Formal description of the substrate generation problem as an optimization and analysis showing the NP-hardness of the problem.
- Novel heuristic solutions to the substrate generation problem by using node or edge centrality measures of the underlying network as well as traffic flow patterns.
- Extensive simulation-based evaluation of the heuristics over router-level topologies by considering a number of different design dimensions, including heterogeneity of core counts on routers and network failures.

The paper is organized as follows: Section II covers related work on routing techniques and multi-core networking solutions. Section III explains basic design principles of MCPR and shows a canonical example. We, then, formulate the substrate generation problem in Section IV. Next, in Section V, we design graph-based and flow-based substrate generation heuristics, and make a qualitative comparison. Section VI presents an experimental setup and extensive evaluation of the heuristics under various scenarios (e.g., heterogeneous vs. homogeneous traffic and core distribution, and link failures) against the legacy single shortest-path routing. Lastly, we summarize MCPR approaches and insights gained from the evaluations, and discuss future work in Section VII.

II. Related Work

Attaining high data transfer speeds require simultaneous handling of load balancing (or traffic engineering (TE)) and congestion management. To reach 100+ Gbps throughput levels, a plethora of such efforts has been made in utilizing the network. Techniques explored include smart scheduling over single [13] or multiple [14] paths, getting congestion notifications from the network [15], running multiple TCP streams each with its own congestion control loop [16], [17], parallel sub-streams with a joint control loop such as MPTCP [9], having multi-core abilities to improve network performance [18], using software-defined networking (SDN) to manage traffic among geo-distributed inter-data centers during the data migration [19]. These approaches to increase the possible throughput out of a network are heavily vested on the availability of diverse, non-overlapping and robust end-to-end paths that are easy to compute and deploy. MCPR is complementary these methods, that manipulate e2e paths for higher network throughput, and focuses on devising an effective multi-path routing approach with the aim providing more diverse e2e paths.

A. Multi-Path Routing

Multi-path routing [10], [20]–[22] has been a popular topic for researchers to solve the TE problem over a network. The main purpose generally has been path diversity. In addition to general use of multiple paths as in MPTCP, there has been recent interest in multi-path routing in various specific settings such as wide-area transfers [23], dynamic provisioning in optical backbones [24], and data-centers [25]. Further, recent efforts have explored interesting directions such as the mix of clustering and multi-path routing for low-energy wireless sensor networks [26], and use of network coding to implement multiple paths within individual paths [27].

A major topic of multi-path routing research has been finding disjoint paths in a network [28]. Most of these efforts have been for mobile wireless networks where disjoint paths can be very useful for routing over the same radio frequency and attaining QoS guarantees [29]. More related to our work is disjoint multi-path routing in wireline networking which has been mostly for attaining better reliability when facing failures. Studies in this line of research included proactive setup of alternative disjoint paths for robustness against failures [30], centralized computation of disjoint multiple paths with minimal network cost where cost is broadly defined [31], two-phased computation of multiple paths that considers cost of all paths and devises a scalable heuristic to find disjoint paths [32], integrating the goal of finding disjoint paths with path quality guarantees [33], and efficient design of failure-resistant routing in data-center networks [34], [35]. These studies of disjoint multi-path routing assumed entirely centralized computation of the paths and did not consider the use of multiple (virtual) topologies. MCPR differs in its design by splitting the computation task to topology construction and shortest-path calculation and uses explicitly defined virtual topologies.

Both path generation and selection need to be performed in a short time period for deployability. Given a large set of paths, path selection [11] was also studied to find the best subset of paths. To generate dynamic multi-path routing, an overlay architecture is proposed by controlling the data traffic dynamically via selected nodes [36]. Thus, a key challenge with multi-path routing has been the adaptation to the real systems because of its computation costs and the lack of support from the existing routing infrastructure which is optimized for shortest-path calculations and rarely allows to send data through more than one paths with current protocols. MCPR considers both of these challenges in its design.

Potential impacts of multi-path routing on the underlying routing infrastructure can be very large. Recent efforts included customizing the infrastructure hardware so as to perform parallel computation of multi-path routing, e.g., by a custom circuit design [37]. More prominently, software-based solutions to the problem have attracted attention, e.g., multi-threaded designs to find paths for global routing between *netlists* [38], asynchronous pipelining for the sets of threads to attain high performance with multi-core systems [39], effective control plane management to better utilize link capacities for update messages [23], and faster lookup of IPv6 addresses with parallelized store/scan of routing tables over multiple cores [40]. A software-based approach [41] to optimize throughput and energy consumption of multi-core routers was explored as well. A common pattern of these developments is the use of multi-core CPUs for simplifying routing. The advent of multi-core routers presents an opportunity for networking and communications [42]. MCPR considers multi-core CPUs as a first-class citizen of its design and decides the number of available e2e paths according to the number of cores in routers.

B. Multi-Topology Routing

Early literature introduced the concept of multi-topology routing (MTR) as an extension to legacy intra-domain routing protocols [43], [44]. MTR enables the use multiple virtual topologies for routing on the underlying physical topology. The design of MTR includes the intra-domain routing protocol to work on setting links weights on the multiple virtual topologies and calculate paths on each of these virtual topologies. Then, the routing also has to determine what fraction of e2e flows are supposed to be fed to each virtual topology. The main goals of MTR are to enable more flexibility in e2e paths for attaining network-wide goals such as TE, multicasting, and fault tolerance. MCPR aligns with the main goals of MTR, which is to improve network performance by using virtual topologies and directing fractions of traffic to each virtual topology. Key differences of MCPR from MTR are two-fold. First, MCPR uses multiple independent shortest-path routing protocols each running on a different virtual topology. In MTR, there is one routing protocol that calculates paths on each virtual topology which can overload the routing daemon when facing failures or other disruptions. MCPR's approach can better utilize multiple cores on router CPUs as each routing protocol daemon can be assigned to a separate core. In this aspect, MCPR is a more general version of MTR. Second, MCPR strictly enforces shortest-path routing on each virtual topology while MTR allows non-shortest-path routing calculations. In this aspect, MCPR is a special case of MTR.

A natural use of MTR is to perform TE [45], [46]. In the same essence as MCPR, MTR enables balancing of traffic across the network by splitting e2e flows to each virtual topology. Early studies aimed to solve the TE problem for a given traffic matrix [47] or a varying traffic demand [48]. These efforts assumed to know what proportion of each e2e flow will land on each link of the virtual topologies. MCPR assumes no a-priori knowledge of traffic projections and builds its virtual topologies for generic demand by focusing on topological properties only. More recently, Mirzamany *et al.* [49] focused on finding disjointed topologies for MTR so as to improve TE performance and proved that finding the multiple disjointed logical topology is NP-hard. Though the insights from this study are useful for MCPR, MCPR does not require disjoint virtual topologies and searches for solutions from a larger design space.

From its start, a key motivation for MTR has been attaining fault-tolerance. The most common approach to using MTR for fault-tolerance involved utilizing the existence of multiple virtual topologies to form backup paths. Several approaches [50]–[52] were proposed to implement IP fast-reroute using MTR. The concept is to switch to a predetermined virtual topology when an adjacency in the intra-domain routing fails. In the same vein, Cicic [53] outlined a framework to design separated sets from a topology to minimize the possibility of multiple virtual topologies being affected by a failure. In these MTR-based fault-tolerance studies, minimizing the number of virtual topologies was also a major goal for reduced maintenance overhead. MCPR differs from these efforts as it focuses on maximizing throughput.

The concept of multiple virtual topologies has applications beyond intra-domain routing. Most recently, there have been proposals to use MTR for enhancing wireless resource management [54], access predictability [55], and latency minimization [56].

III. MCPR: A Parallel Routing Framework

Legacy routing systems, especially for the inter-datacenter connections, employ shortest path algorithms to find e2e paths. However, due to its greedy nature, such shortest path routing causes congestion, and hence limits the amount of data that can be sent over the network. Attaining better load balancing and higher throughput heavily depends on fast, scalable and robust calculations of multiple paths between source-destination pairs. Since such multi-path calculations are complex, MCPR takes a divide-and-conquer approach [2]. It divides the multi-path routing problem into multiple shortest path calculation problems by abstracting the underlying network topology as different substrates, and uses legacy shortest path calculation algorithms to swiftly and efficiently calculate/conquer each substrate.

The main idea of MCPR is that different slices (i.e., substrates) of the router topology are given to each core and the e2e data transfer is split onto shortest paths calculated on each substrate topology. Once assigned to a substrate i, a flow will follow the shortest path calculated by the substrate i. However, all the flows will be using the same underlying physical topology regardless of which substrate they are assigned. Hence, MCPR can lead to a system where "parallel" routes are produced on virtual substrates over the same physical topology.

Many of the current routers have multi-cores which can be utilized to execute the conquer phase of MCPR. Different from most of multi-threaded solutions for routing, *MCPR runs a separate instance of a legacy shortest path algorithm on each*

core and does not need to concatenate the results coming from each instance/core. The resulting set of multiple paths is the combination of the shortest paths calculated by each core. It is then up to the e2e transport protocols such as MPTCP to decide which one of these paths from the substrate topologies to use and with what rate.

A. Design Approach and Principles

The following principles summarize the main design features of MCPR:

i- Shortest path is abundant and efficient. Legacy shortest path routing solutions are very much optimized and designed into the fabric of routers. Multi-path calculations utilizing them will be straightforward to deploy.

ii- Multi-core CPUs are readily available. It is possible to execute multiple shortest path routing daemons in parallel on the existing routers with multiple cores. Each core can run a separate instance of legacy routing protocols such as OSPF [57].

iii- Robustness against network dynamics is achievable via shortest-path (re)-calculations. A critical challenge of multi-path routing is its brittleness against network dynamics such as failures or demand spikes. Such network changes may require re-calculation of the entire multi-path set, which can be prohibitively costly in routing timescales. MCPR delegates the path re-calculation to each substrate and lets the shortest path routing algorithm on each substrate perform the re-calculation using the existing optimization approaches.

iv- Substrate generation can be centralized. The "divide" part of the MCPR is the most challenging as it requires finding the best set of substrate topologies so that their shortest paths minimally overlap. This is a heavy computation task (see Section IV). On the other hand, such heavy computation can be done in SDN controllers or other centralized locations with high computation power. Additionally, the substrate generation could be done at larger timescales without any major sub-optimality. Failures or network dynamics do not necessitate the substrate generation to be done within the routers themselves since the re-calculations of shortest paths can be independently performed by each core. Finally, centralizing substrate generation allows goals like multi-path traffic engineering which requires a global and centralized view of the network.

B. A Motivating Scenario

If (i) different virtual routing topologies based on the actual topology are assigned to a separate core of multi-core routers, (ii) data transfers could be distributed over these virtual topologies, and (iii) the current well-known shortest-path calculation techniques are executed on each core; then data load could potentially be better distributed over the network.

Fig. 1 illustrates a motivating scenario where two virtual substrates are produced. substrate 0 is equivalent to the genuine router topology, whilst substrate 1 is generated by removing node 3 from substrate 0. Even though there are many other possible paths available over the network, current systems would carry 5 Mb/s on a single path. But, if these two

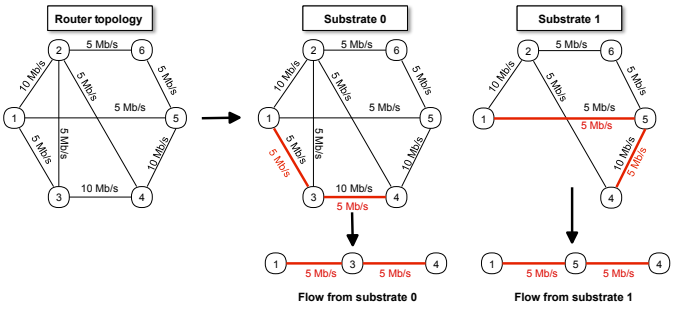

Fig. 1. Motivating scenario with two cores.

TABLE I
IMPORTANT SYMBOLS.

Symbol	Meaning
$G = \{V, E\}$	Graph representation of physical network with V vertex set and E edge set
w_{uv}	Weight (or cost) of the edge from vertex u to vertex v
$G_i = \{V_i, E_i\}$	A subgraph of G with $V_i \subseteq V$ and $E_i \subseteq E$
$S_i = \{G_i, W_i\}$	A substrate of G, composed of subgraph $G_i = \{V_i, E_i\}$ with edge weights W_i
S	The set of all possible substrates of G
Q	A subset of S, all possible substrates of G
$P(Q)$	The set of shortest paths generated from substrates in Q
$T(Q)$	Total throughput of the shortest paths in $P(Q)$
V_S	The subset of vertices in G that generate a flow
V_D	The subset of vertices in G that are flow destinations
$f_{s \to d}$	A flow from source s to destination d
$\mathcal{F}$	The set of all flows
f_{uv}	A flow on edge $(u, v) \in E$
f^i_{uv}	A flow on en edge (u, v) in S_i, i.e., $(u, v) \in E_i$

topologies were given to a two-core router and *current shortest path finding algorithms* were executed in parallel on each core, then each path could transfer 5 Mb/s data. Compared to the current systems, each substrate's shortest path is different, and the two collectively yield a total of 10 Mb/s throughput from node 1 to 4 for the *two cores* scenario.

While creating a new substrate, in addition to removing node(s), some edge(s) can also be omitted, such as the edge connecting node 1 and node 3. In that case, similar to the previous example, there would be two different paths to reach the destination node 4, i.e., path 1-3-4 and path 1-2-4. Each substrate is able to find the shortest path that can carry 5 Mb/s and a total of 10 Mb/s capacity can be transferred. This is double the capacity of current systems that calculate single shortest-path over the topology, e.g., OSPF.

IV. Substrate Generation Problem

We provide a formal definition of the substrate graph generation problem of MCPR. Table I shows the notation we use for the formulation. Consider an underlying network topology as a graph $G = \{V, E\}$ with a set of vertices V and edges $(u, v) \in E$ where $u, v \in V$. Assuming there are n nodes and m links in the topology, $|V| = n$ and $|E| = m$. Given G, MCPR's substrate graph generation involves several decision parameters: (i) *number of substrate graphs* to generate and (ii) *edge weights* on each substrate graph. Let w_{uv} be the weight

(or cost) of the edge from u to v where $u, v \in V$. Suppose the weights can be set to $k-1$ different integer values or an infinitely large number, i.e., $w_{uv} \in \{1, 2, \cdots, k-1\} \cup \{\infty\}$ for all $u, v \in V$. Let S be the set of all possible substrates of G. Since each edge in E can have k different weight values, the number of possible substrate graphs from G is $|S| = k^m$. Accordingly, a substrate $S_i \in S$, can be expressed as $S_i = \{G_i, W_i\} = \{\{V_i, E_i\}, W_i\}$ where $V_i \subseteq V$, $E_i \subseteq E$, and W_i is the weights assigned to the edges in E_i.

To formally define our problem, we further define the followings: The capacity of an edge is a mapping $c : E \to \mathbb{R}^+$ denoted by c_{uv} where $(u, v) \in E$. It represents the maximum amount of flow that can pass through an edge. Let $V_S \subseteq V$ and $V_D \subseteq V$ be the set of sources and destinations, respectively. We define a flow on an edge as a mapping $f : E \to \mathbb{R}^+$ denoted by f_{uv} where $(u, v) \in E$ and $f_{uv} < c_{uv}$. Without loss of generality, we further define throughput (or rate) of a flow $f_{s\to d}$ from a source $s \in V_S$ to destination $d \in V_D$ as:

$$|f_{s\to d}| = \sum_{v:(s,v)\in E} f_{sv} = \sum_{v:(v,d)\in E} f_{vd}. \tag{1}$$

This approach models the flow throughput as a fluid flowing through the network, following the capacity constraints and directions of each edge.

A. Maximum Substrate Set Problem

The overall goal of MCPR is to maximize the throughput of the network by using the shortest paths from a subset of the substrates. Let $\mathcal{F}$ be the set of all flows $f_{s\to d}$ where $s \in V_S$ and $d \in V_D$. Further, let $Q \subseteq S$ represent a set of substrates $Q = \{S_1, S_2, \cdots, S_q\}$ and $P(S_i)$ be the collection of shortest paths generated from substrate S_i. Further, let $T(S_i)$ be the total throughput attained from the shortest paths in $P(S_i)$. We can formulate MCPR's problem of generating a set of substrates that maximizes the throughput as follows:

MAXSUBSTRATESET(G,$\mathcal{F}$):

$$Q^* = \arg\max_{Q\subseteq S} \sum_{S_i\in Q} T(S_i) \tag{2}$$

subject to

$$\sum_{v:(s,v)\in E} f_{sv} = \sum_{v:(v,d)\in E} f_{vd}, \qquad f_{s\to d} \in \mathcal{F}, \tag{3}$$

$$\sum_{u:(u,v)\in E} f_{uv} = \sum_{w:(v,w)\in E} f_{vw}, \qquad v \in V \setminus \{s, d\} \ \ \forall f_{s\to d}, \tag{4}$$

$$f_{uv} \le c_{uv}, \qquad \forall (u, v) \in E, \text{ and} \tag{5}$$

$$\exists\, u \to v \in \bigcup_{S_i\in Q} P(S_i), \qquad u, v \in V, \tag{6}$$

where $s \in V_S$ and $d \in V_D$ are source and destinations of the flows to be carried by the network, and $u \to v$ is a path from node u to node v. The constraints (3) make sure each flow departs its source and arrives its destination in its entirety. Equation (4) assures the conservation of each flow at all vertices other than its source and destination. Equation (5) is the capacity constraints for all edges of G. Finally, (6) assures the resulting multi-path routing provides at least one path between all source-destination pairs.

We now further detail MAXSUBSTRATESET by expanding $T(Q)$. To do so, we denote a flow on substrate S_i with f^i_{uv} where $(u, v) \in E_i$. Also, we denote the shortest path from vertex u to vertex v as an ordered set of edges $P(u \to v) \subseteq E$. With that, we rewrite MAXSUBSTRATESET(G,$\mathcal{F}$) as follows:

$$Q^* = \arg\max_{Q\subseteq S, f^i_{sv}, i=1\cdots q} \sum_{s\in V_S} \sum_{S_i\in Q} \sum_{v:(s,v)\in E_i} f^i_{sv} \tag{7}$$

subject to

$$(3) - (6),$$

$$\sum_{S_i\in Q} \sum_{v:(s,v)\in E_i} \frac{f^i_{sv}}{|f_{s\to d}|} = 1, \forall f_{s\to d}, \tag{8}$$

$$\sum_{u:(u,v)\in E_i} f^i_{uv} = \sum_{w:(v,w)\in E_i} f^i_{vw}, \quad v \in E_i \setminus \{s, d\} \ \forall f_{s\to d} \ \forall S_i \in Q, \tag{9}$$

$$\exists v : f^i_{sv} = \sum_{v:(s,v)\in E_i} f^i_{sv}, \qquad v \in S_i, \forall f_{s\to d} \ \forall S_i \in Q, \text{ and} \tag{10}$$

$$\sum_{(u,v)\in E_i \setminus P(s\to d)} f^i_{uv} = 0, \qquad \forall f_{s\to d} \ \forall S_i \in Q. \tag{11}$$

The objective function now details that the sum of the throughput of the flows starting at all sources is to be maximized. The constraint (8) makes sure the fractions of each flow assigned to each substrate graph sum to 1. Equation (9) is the set of constraints assuring that flows in each substrate graph are conserved. Equation (10) assures that there is only one flow from each source inside a substrate graph. Finally, (11) assures that only the edges on the shortest paths inside a substrate graph are utilized.

Formulations of MAXSUBSTRATESET above look at the problem in a black box manner. It is possible to express the substrate generation problem from the network's point of view in a white box style. In particular, network throughput is maximized when routing calculates paths with minimal overlap. Next, we will express the substrate generation problem in terms of minimizing overlap.

B. Minimum Substrate Set Problem

For a graph g=$\{v, \epsilon\}$, let R_g be the routing vector such that $R_g(l)$ is the number of shortest paths traversing link $l \in \epsilon$. Note that $R_g(l)$ is the edge betweenness centrality of a node. Given a set of substrate graphs $Q = \{S_1, S_2, \cdots, S_j\}$ in MCPR, we can express the number of shortest paths traversing l as

$$t(l, P) = \sum_{S_i\in Q} \sum_{g\in P(S_i)} R_g(l). \tag{12}$$

The substrate generation problem of MCPR is, then, to make the use of each link as even as possible, which also implies a minimal overlap among shortest paths. To factor in the varying number of substrates, we can aim to minimize the difference between the minimum and the maximum $R_g(l)$. Hence, we write MCPR's substrate generation problem as a minimization of the unevenness in the usage of links, MINSUBSTRATESET(G,$\mathcal{F}$), in G=$\{V, E\}$:

$$Q^* = \arg\min_{Q\subseteq S} \quad (\max_{l\in V} t(l, Q) - \min_{l\in V} t(l, Q)) \tag{13}$$

$$\text{subject to } (3)-(6).$$

MINSUBSTRATESET provides a clear guidance on how heuristics should be designed for the substrate generation problem. In the next section, we will use this guidance to minimize the maximum load on individual links while trying to maximize the aggregate throughput. Producing a new substrate dynamically to reach the optimum result is hard to determine in a tractable time cost. On each step, we remove one or more network element(s). In MCPR, we determine the elements by using some heuristics, based on network centrality metrics, detailed later in Section V.

C. Analysis

The MAXSUBSTRATESET(G,$\mathcal{F}$) problem is quite similar to the multi-commodity flow (MCF) problem, which is the basis for quite a lot of multi-path routing problems [58]–[61]. The goal of MAXSUBSTRATESET(G,$\mathcal{F}$) is the same as MCF, i.e., maximizing the network flow. However, MAXSUBSTRATESET(G,$\mathcal{F}$) differs from MCF as it requires the paths to be shortest in a subgraph of G. Instead of finding the optimal fraction of flows on each edge as in MCF, it tries to find the optimal set of subgraphs (i.e., $Q \subseteq S$) as well as the fraction of flows on each subgraph (i.e., f_{sv}^i) to maximize the network flow. To be compliant with routing practices, MAXSUBSTRATESET(G,$\mathcal{F}$) assigns weights to each edge (i.e., w_{uv}) when generating a substrate. These are virtual values and allows MAXSUBSTRATESET(G,$\mathcal{F}$) to produce different shortest paths in each substrate so that the overall network flow can be maximized.

Corollary 1: MAXSUBSTRATESET*'s search space is* $\mathcal{O}(2^{k^m})$.

Proof: Let Q^* be the Q that maximizes $T(Q)$. Finding Q^* requires scanning of all possible Qs. Let $\hat{Q}$ be the set of all possible $Q \subseteq S$. Then, the search space size for MAXSUBSTRATESET is the size of $\hat{Q}$, which is:

$$|\hat{Q}| \quad = \quad \sum_{i=1\cdots|S|} \binom{|S|}{i} = 2^{|S|} - 1. \tag{14}$$

Substituting $|S| = k^m$ in (14), we find the number of possible substrate sets $|\hat{Q}| = 2^{k^m} - 1$, which is clearly not polynomial in terms of the number of edges m. □

MCF is known to be NP-Complete when the flow rates are integer, but can be solved in polynomial time if fractional flows are allowed. Since MAXSUBSTRATESET(G,$\mathcal{F}$) requires setting of the link weights w_{uv} within each substrate, it is a more complex version of the network link weight setting (NLWS) problem. For a single network (or substrate), optimal setting of link weights to maximize throughput of shortest path routing for a given set of flows was shown to be NP-hard [62], [63]. Let NETLINKWEIGHT(G,$\mathcal{F}$) be the problem of finding the optimal link weights for graph G such that the network throughput is maximized for the flow set $\mathcal{F}$.

Corollary 2: *The maximum network throughput for graph G with optimal solution to NLWS is greater than or equal to the maximum network throughput of any of its subgraph G' with optimal solution to NLWS, i.e.,* NETLINKWEIGHT*(G',$\mathcal{F}$)* $\leq$ NETLINKWEIGHT*(G,$\mathcal{F}$) for* $G' \subseteq G$.

Algorithm 1 A Solution to MAXSUBSTRATESET using NETLINKWEIGHT

```
1: procedure MAXSUBSTRATESET(G,F)
2: Inputs
3: G = {V, E}: Bidirectional graph representation of a network
4: F: A set of flows f_{s→d} where s, d ∈ V
5: Output
6: Q: A set of substrate graphs maximizing flow of F on G
7:     Q ← ∅
8:     while |E|! = 0 and Σ_{f_sd ∈ F} F! = 0 do
9:         S ← NETLINKWEIGHT(G, F)
10:        Q ← Q ∪ S
11:        F̂ ← SHORTESTPATHFLOW(S, F)
12:        F ← F − F̂
13:        Remove all edges of G that are maxed out by F̂
14:    end while
15: return Q
16: end procedure
```

Proof: G' is either equivalent to G or it does not have one or more edge of G. Removing any edge (or vertex) from G will eliminate the possibility of utilizing the capacity of that edge (or the edges connected to that vertex) when trying to maximize the network throughput over the subgraph G'. Regardless of how the routing of flows will be on G', the total network throughput will not be greater than what is possible with G. □

Lemma 1: *When the number of substrates is limited to 1 (i.e., $|Q^*| = 1$),* MAXSUBSTRATESET*(G,$\mathcal{F}$) is equivalent to the NLWS problem, which is NP-hard.*

Proof: A substrate $S_i = \{G_i, W_i\}$ of G has two components: The set of vertices V_i and edges E_i from the original graph G and the weights W_i of the edges in E_i. Intuitively, removing any edge (or vertex) from G when generating the only substrate in Q will eliminate the possibility of utilizing the capacity of that edge (or the edges connected to that vertex) when trying to maximize the network throughput using that substrate. In other words, NETLINKWEIGHT(G_i,$\mathcal{F}$) $\leq$ NETLINKWEIGHT(G,$\mathcal{F}$), which follows from Corollary 2. □

Beyond setting link weights in a substrate, MAXSUBSTRATESET(G,$\mathcal{F}$) requires optimal selection of (i) a substrate set Q from the original network G and (ii) the fraction of flows f_{sv}^i assigned to each substrate. These factors further complicate the problem.

Lemma 2: MAXSUBSTRATESET*(G,$\mathcal{F}$) is polynominal time reducible to the NLWS problem, i.e.,* MAXSUBSTRATESET$(G, \mathcal{F}) <_P$ NETLINKWEIGHT$(G, \mathcal{F})$.

Proof: We prove by devising an iterative algorithm to reduce MAXSUBSTRATESET to NETLINKWEIGHT. Algorithm 1 shows the steps of the algorithm assuming an $\mathcal{O}(1)$ solution to NETLINKWEIGHT$(G, \mathcal{F})$ is available.

In each of its iteration, the reduction algorithm calls (line 9) NETLINKWEIGHT with the input graph, G, representing the physical network and the set of flows, $\mathcal{F}$, to be laid on the

network. This returns the optimum weight assignment to the edges on G such that the flows in $\mathcal{F}$ are maximally satisfied. This weight assignment and the original graph G becomes the first substrate to be added (line 10) to the solution Q. Then, in line 11, we apply shortest-path routing of $\mathcal{F}$ on S and obtain what fraction of the flows in $\mathcal{F}$ are satisfied by this routing. Some of these flows will compete on the same edges, in which case an arbitrary mechanism to share the contended edge capacity can be employed. Then, in line 12, the satisfied portion of the flows are subtracted from the original set of flow demands. Finally, at end of the iteration, the edges that have no capacity left are removed from the graph G. The algorithm iterates until either there are no edges left in G or there are no flows left in $\mathcal{F}$ to satisfy.

Each iteration of the algorithm has $\mathcal{O}(n \log n + m)$ complexity due to the shortest-path calculation in line 11. A careful inspection of the algorithm reveals that it will not iterate more than m times. This is due to the assumption that NETLINKWEIGHT find the optimum link weights for carrying the remaining flows on the remaining graph. Since it is finding the optimum weights, it must find the maximum flow for at least one flow in $\mathcal{F}$. From the max-flow min-cut theorem [64], we can deduce that the outcome of line 9 will cause at least one edge to be maxed out. This means that line 13 will remove at least one edge at every iteration of the algorithm, which guarantees that the while loop will not iterate more than m times. Hence, the overall complexity of the reduction algorithm is $\mathcal{O}(mn \log n + m^2)$, which is polynomial. □

Theorem 1: MAXSUBSTRATESET *is NP-hard.*
Proof: It follows from Lemma 1 and Lemma 2. □

V. MCPR HEURISTIC DESIGN

MCPR creates virtual slices, i.e., *substrates*, from the entire router topology to run those substrates on each core for getting more diversified paths for e2e data transfers. In the entire set of possible substrates, some topologies could be incapable of producing better results, e.g., that could include a partitioned set of nodes in a substrate which can not route within the substrate. Considering such constraints, our design goal of MCPR is to find optimal substrates that yield diverse and non-overlapping shortest paths. Thus, we develop intuitive heuristics to create new substrates which can improve total aggregate throughput.

When generating the substrates, a crucial challenge is to assure all-to-all connectivity. To address this issue, we define **substrate 0** as the actual topology. When generating the subsequent substrates, however, some nodes or edges are going to be omitted based on some criteria. A simple heuristic step could be to omit the nodes/edges that are being used the most by the shortest paths in substrate 0.

While generating subsequent substrates, some network elements, such as nodes or edges, will be omitted from the network. Our heuristics analyze the genuine topology to predict which routers/edges could be maxed out (i.e., utilized) earlier than others. These routers/edges are removed from some of the subsequent substrates to balance the load in the underlying network. Thus, the primary design parameter is to decide which nodes/edges are going to be omitted on the generated substrate. Our first step of heuristics could be to omit the most utilized nodes or edges that might be more centralized as these elements can be used more than the others.

Algorithm 2 Graph-based removal

procedure GRAPH-BASED($amount, coreCount$)
  $substrateList.Add(actualTopology)$
  $metrics \leftarrow [NBC, NCC, NDC, etc.]$
  $sortedList \leftarrow nodes.Sort(metrics)$
  $numOfSubstrates \leftarrow 1$
  while $numOfSubstrates < coreCount$ **do**
    $elements \leftarrow sortedList.Select(amount)$
    $newTopology \leftarrow formerT.Remove(elements)$
    $substrateList.Add(newTopology)$
    $numOfSubstrates \leftarrow numOfSubstrates + 1$
  end while
end procedure

Depending on the timescale of changes to the best solution, we use two different heuristic criteria for determining the "central elements" of the network: Graph-based or flow-based. The former suits to large timescale multi-path calculations while the latter focuses on short timescale network traffic dynamics. In the *graph-based* technique, the whole topology is analyzed and all the edges connected to the central elements are expelled on the next generated substrate. Similarly, in the *flow-based* method, the traffic pattern at the particular time is inspected, and selected edges with highest utilization will not be on the new substrate. Therefore, our heuristics are concentrated on figuring out the possible network dynamics in a feasible way, and finding the central network elements that can be omitted to generate new substrates.

As substrate 0 is the original topology to guarantee the connectivity for other substrates, the generating process could be done in a cumulative approach or an independent approach. In the *cumulative* method, all substrates are generated from the previous substrate that is already produced. This process ensures that network elements removed in the previous substrates, cannot be on the newly created substrate. For instance, in order to calculate the 4^{th} substrate for a 4-core router, 2^{nd} and 3^{rd} substrates should have already been calculated for that router. If an edge was removed in the 2^{nd} substrate, it would not be on the 3^{rd} or subsequent substrates. Whereas, in the *independent* method, all substrates are calculated from the original topology, and so, all of the other substrates can be generated at the same time.

A. Graph-Based MCPR Heuristics

Graph-based heuristics are purely based on the graph properties of the genuine topology. In graph-based heuristics, network centrality metrics are used to find the most 'central' nodes/edges as they are most likely to be maxed out by traffic flows. The main purpose of these methods is to increase the number of non-overlapping paths for the newly generated substrate. For the graph-based approaches, *the only*

consideration will be the topology information, regardless of the traffic pattern, to determine the order of nodes/edges to be removed. Therefore, the selected nodes/edges for removal can be pre-calculated and remain unchanged until the topology changes. On the other hand, the graph-based methods might not be robust with the dynamic traffic flow patterns because they determine the potential congestion areas only by using the topology information. Algorithm 2 shows the steps for graph-based heuristics.

The graph-based heuristics are based on the centrality characteristics of the nodes/edges in the network topology. Central nodes/edges are calculated and selected as potential congestion areas without analysis of the actual network flow.

Node centrality. We utilized the following centrality metrics to determine which nodes to remove:

Node degree centrality (NDC) measures the number of in/out connections between the nodes. If a node has a higher degree, that node can potentially carry more traffic than others around it as it connects to a greater number of nodes. In NDC, the highest degree nodes are eliminated first.

Node betweenness centrality (NBC) is a metric that measures the number of shortest paths among all pair of nodes traversing through a particular node. NBC calculates all shortest paths between every possible node pair to find how many shortest paths pass through a particular node. NBC reduces possible intersection of the paths that is most commonly used on the shortest paths. NBC, in essence, captures the very basic notion of overlapping shortest paths. Thus, eliminating the nodes with high NBC is in direct intuition with the goal of graph-based heuristics, i.e., to increase non-overlapping shortest paths in the subsequent substrates.

Node closeness centrality (NCC) measures the distance from a node to all others to determine which node is the closest to every node. This heuristic would identify nodes that are at the center of the graph. NCC selects the nodes for removal when they have the average shortest distance to all other nodes, with the intuition that such a node would be on more of the shortest paths.

Harmonic closeness centrality (HCC) is a modified version of the NCC where the sum of distances is inverted. In particular, HCC calculates the sum of the inverted distances. We try to see the effect of this little change on MCPR.

Eigen vector centrality (EVC) measures the centrality of a node based on the centralities of its neighbors in a recursive manner. This measure accounts for nodes that are not only high degree but are also connected to other high-centrality nodes. Hence, this heuristic would identify nodes that are not only central themselves but also connected to other central nodes.

Page rank centrality (PRC) is an extension of EVC that provides a default centrality to all nodes independent of their links. PRC provides a centrality score to all nodes as EVC gives a centrality of zero to nodes that are not in the strongly connected component. We used a damping factor of 0.85 as suggested by [65].

Random node (RN) provides a greedy solution with the minimal computation cost as baseline. RN randomly picks nodes to remove until the removal percentage is fulfilled. This heuristic allows us to observe the performance of an uninformed removal process.

Edge centrality. We utilized the following centrality metrics to determine which edges to remove:

Edge betweenness centrality (EBC) is the edge version of NBC. Instead of counting the number of shortest paths through a given node, EBC calculates how many shortest paths use a given edge. Removing the most used edges can be helpful to obtain higher aggregate throughput over the network rather than losing the most central node with all of its edges.

Edge closeness centrality (ECC) measures closeness of an edge to other edges (similar to NCC) and removes the edge at the center of the graph from the substrates.

B. Flow-based MCPR Heuristics

As current systems typically use the shortest path for deciding a path from a source to a destination, some edges can be shared by multiple paths. Hence, overlaps of e2e paths causes congested spots on the network. Intuitively, the bandwidth of the most used edges are going to be maxed out earlier than the other edges. As each substrate calculates its own shortest paths, load on the shared edges of different substrates could be high. As we remove the most utilized edges from the previous substrate(s), we try to obtain short paths that avoid congestion spots. Thus, our main goal in flow-based heuristics is predicting which edges carry more data than other edges, and balancing the load across all edges of the topology.

The graph-based properties might not capture the dynamism of network traffic. Designing heuristics that consider the current utilization of edges could be favorable to adapt the substrate generation process. We also employ flow-based heuristics as a dynamic approach that depends on the traffic patterns in addition to the underlying topology. The generated substrates change with the estimated data flows. For each flow set, a new substrate set must be generated periodically, which increases the computational overhead. While flow-based heuristics are adaptive to the traffic dynamics, they are computationally intensive. We consider the following flow-based heuristic:

Highest flow (HF) focuses on the current flow of edges. To generate a new substrate, HF counts existing flows traversing each edge and omits the highest load edges in substrate i to produce subsequent substrate i+1. In each step of the substrate generation, we remove a number of edges according to the target removal percentage. If $s = |Q|$ represents the number of substrates and $f = |\mathcal{F}|$ is the number of flows, the time complexity for HF is $\mathcal{O}(sfn^2 \log n)$. Although the method can be faster than some of the graph-based heuristics, HF should be performed whenever the traffic load changes.

C. Removal Methods

Both of the graph-based and flow-based heuristics produce an ordered list of network elements (i.e., nodes or edges) to

TABLE II
SUBSTRATE GENERATION HEURISTICS AND REMOVAL METHODS.

	Highest flow	Graph-based (edge removal)	Graph-based (node removal)	Random removal
Block	+	+	+	-
Bucket	-	+	+	-
Region	-	+	+	-
Individual	-	+	+	-

be removed based on how central they are to the network. In MCPR a measure of aggressiveness, "removal percentage", is given as the percent of elements to be removed from the topology for generating the next substrate. To complete the substrate generation step, selected heuristic removes network elements up to the a given *removal percentage*. We consider four different selection methods to remove elements from the centrality list and apply heuristics in Table II.

(a) Block remove chooses a block of elements, starting from the most central node to the least, filling up the removal percentage. In the first substrate, it removes the highest ranged block; in the second one, the elements in the second highest block are removed, and so on. This method removes the most central nodes at the same time, and tries to offload the load to the less central nodes in the subsequent substrates.

(b) Bucket remove creates the blocks as described in the block remove, but, instead of clearing the whole block at once it uses blocks as different clusters of the items on the ordered list. It uses a round robin to choose one element from each group into the bucket and then clears different buckets in each substrate. Hence, this method does not remove the most central components at the same time.

(c) Region remove performs breadth first search (BFS) [66] to discover the neighbors of the most central item (i.e., the most central element in the given ordered list) and remove the network element along with its BFS tree. For the next substrate, it choose the subsequent most central element, and so on to clear the neighboring region. We do not impose a depth limit while creating the BFS tree and continue until it reaches the given removal percentage target. We applied both edge-based and node-based regional remove in our experiments.

(d) Individual remove focuses on one element from the top of the centrality list. Note that this method is not applicable to heuristics and does not use the "removal percentage" notion since it omits only one element in each step.

D. Handling Heterogeneity in Core Distribution

There are different kinds of routers manufactured by various companies with multiple CPU cores. Since upgrading a large network with new routers is not possible in a short period of time, the homogeneity of the routers cannot be assumed. As a result, we need to find MCPR heuristics that can work across heterogeneous routers with different number of core.

Each router core will have a different topology as MCPR empowers each core to handle one of the generated substrates. The substrates may not be efficiently hosted by some of the routers if their core count is fewer than the number of generated substrates.For example, when four substrates (i.e., substrate 0–3) are created, substrate 2 and substrate 3 cannot be assigned to a dedicated core at a 2-core router. When the 2-core router has a flow coming from a substrate that is not assigned to a separate core (e.g., substrate 3), the forwarding of the data packets for that flow will have to be handled by one of the two cores that are already assigned a substrate (e.g., substrate 0 or substrate 1). This means that the flows on these "unassigned" substrates will have to share the cores with other substrates. Such sharing of the cores will slow down the control plane and, in turn, forwarding actions. Hence variance in the number of cores deteriorate the overall performance.

To design MCPR heuristics that can run over routers with the heterogeneous count of cores, we take two different design approaches:

(a) Preventing excessive substrates (PES) removes excessive substrates. In this approach, after deciding which elements will be removed to generate a new substrate, routers are also omitted if their core count is fewer than the substrate count. Because such routers will not able to process the data coming from the excessive substrates, the nodes with a core count fewer than i are omitted while generating the substrate j where $j > i$. Note that this is a preventive approach since it eliminates the possibility of having substrate count exceeding the core count of routers.

(b) Data loss model (DLM) generates substrates as if there is a homogenous core count across the routers, and then create a data loss model by considering the additional processing delay at the routers with excessive substrates. This allows the possibility that some routers will serve more substrates than their core count. In these cases, we estimate the additional load on the routers due to the excessive substrates. Following the observations on the performance of MPTCP, we proportionally reduce the achievable data flow speeds for the overloaded routers [67], [68]. In order to evaluate the performance of this approach, we reduce the data flow speed in proportion to how many times the router's interfaces were used. We calculate the scaling factor of nodes according to their overload, i.e., scaling factor for a 2-cored router with 4 substrates will be 2/4, and compute an aggregate data loss for each flow by considering all nodes traversed by the data flow. When computing the aggregate data loss on a path, we consider delay-tolerant and loss-tolerant transfers since application traffic may have different preferences. In loss-tolerant transfers, we multiply scaling factors of the nodes on the path, which we call DLM-product (DLM-P). In delay-tolerant transfers, we use the arithmetic average to determine the portion of data loss, which we call DLM-arithmetic (DLM-A). We also use the harmonic average to mix both delay and throughput tolerant scenarios, which we call DLM-harmoni (DLM-H). At the end, the calculated portion of the data is determined as data loss because of the excessive substrates.

TABLE III
CHARACTERISTICS OF ROCKETFUEL NETWORK TOPOLOGIES.

Network	Nodes	Edges	Max deg	Avg. deg	Avg. path len	Cluster. coeff.	Assortativity
AboveNet	141	922	40	13.1	3.62	0.269	0.698
Ebone	87	403	51	9.3	3.90	0.299	0.357
Exodus	79	352	24	8.9	3.94	0.286	0.749
SprintLink	315	2333	90	14.8	3.89	0.331	0.387
Telstra	108	368	92	6.8	2.89	0.171	0.006
Tiscali	161	874	406	10.9	2.31	0.072	-0.063

VI. EVALUATION OF MCPR HEURISTICS

A. Experimental Setup

We performed static analysis of the MCPR heuristics in C++ by creating a new environment for ***calculating the amount of data throughput*** over a given network. In this analysis, we create substrates with the heuristic methods, find the end-to-end paths for data flows on each substrate, carry all data on the substrates by using max-min allocation method for sharing capacities of edges/links. Then, aggregate throughput can be calculated for a given scenario.

We tested our heuristics on six Rocketfuel topologies that have different characteristics [69]. Table III presents the number of nodes, number of edges, maximum degree, average degree, average path length, clustering coefficient, and assortativity of the topologies we have used.

We compared the MCPR against the -currently used- single shortest path routing to analyze the performance based on the total throughput across the network. Since the information provided by Rocketfuel is not enough to see all aspects of the given network, we created a new *network model* that also includes flow patterns, link capacities, flow demands of the end nodes. We generated network flows between all node pairs based on the gravity model (the product of populations divided by the square of the geo-distance) between the point of presences (PoPs) of Rocketfuel topologies. We assumed the link capacities are inversely proportional to the link weights that were provided by Rocketfuel, since the edges with lower weights will more likely be on the shortest paths. Then, we used max-min allocation to determine the e2e rates the flows will attain. We normalized the flow rates based on the smallest flow rate, and, so our throughput results are shown in normalized units.

Topology: Rocketfuel topologies have information on merely nodes and links. In our analysis, we determine link capacities inversely proportional to the delay-based edge weights provided by Rocketfuel.

Flow patterns: In our analysis, the gravity-based flow pattern, which uses actual population and geo-location of the cities where nodes are located, is applied to generate the data traffic for each pair of nodes. Flow demands (or rates) are calculated with the product of populations divided by the square of the geo-distance between two locations. We also normalized the flow demands by the minimum flow rate attained by any flow in the network.

Traffic rate and link capacity unit compability: Flow demands and the link capacities are both estimated and given in normalized units. Thus, unit compatibility between these two factors is needed to measure the total throughput. We perform this compatibility during the max-min allocation when flows share link capacities. Instead of equal distribution, we share link capacities based on flow demands which are flown through a particular edge. So, each flow gets their portion based on their demands.

Load balancing on substrates: Our proposed method reduces multi-path calculation problem to another problem: *Finding optimal subsets of possible substrates to achieve maximum total throughput sent over the network.* In this model, it is needed to determine the load on each substrate for a specific flow. In our analysis, in order to calculate the throughput attained by substrates, we assign flows' traffic loads incrementally. First, we send maximal traffic load on the first substrate, re-arrange the link capacities and use the next substrate(s) to send the rest of the traffic. For instance, let us assume that we have two cores and one flow. We try to send the whole flow on substrate 1. If there is still flow traffic needed to be sent, we carry it on substrate 2. A finer grained model could look for better ways of splitting the flows' traffic load on to the underlying substrates.

Distribution of the core count for heterogenous scenarios: The core count of the routers has not been observed in the Rocketfuel topologies. We create a new estimation model for core numbers on the routers to test the MCPR with heterogeneous scenarios. According to our assumption(s), the most used router(s) should be able to process more data passing through them than the other routers. To calculate the core-numbers, we use geo-location and population data for a router. We assign 2, 4, 8, 16, and 32 core counts to routers, and we assume an equal amount of each router category in the topology. For example, when a topology has 25 routers, we assume that the top 5 most-used, based on the population of their geo-locations, routers have 32 cores and following top 5 routers have 16 cores and so on.

B. Speedup Against Single Shortest Path

We compare MCPR heuristics against the single shortest path according to the total throughput achieved from the generated substrates in total. We test both the *cumulative* and *independent* approaches during the substrate generation process. In the independent generation, all substrate graphs are generated from the actual topology. In the latter one, each substrate is forked from the previously created one and hence later graphs have a larger portion of the network removed. We also test the performance in different percentage of node removals and different removal approaches (Section V-C). To analyze the speedup of heuristics, we use the single shortest path performance as a baseline. Note that the flow-based heuristic HF is expected to provide the best speedup as it dynamically adopts to the network flow in substrates. Similarly, the random node removal heuristic RN is expected to provide the lower bound of speedup as it randomly removes nodes from the substrates without any additional calculation or estimation.

Regardless of the choice of substrate generation heuristic, MCPR outperforms single shortest-path routing, with average throughput speedup ranging from 1.3 with 2 cores to 2.6 with

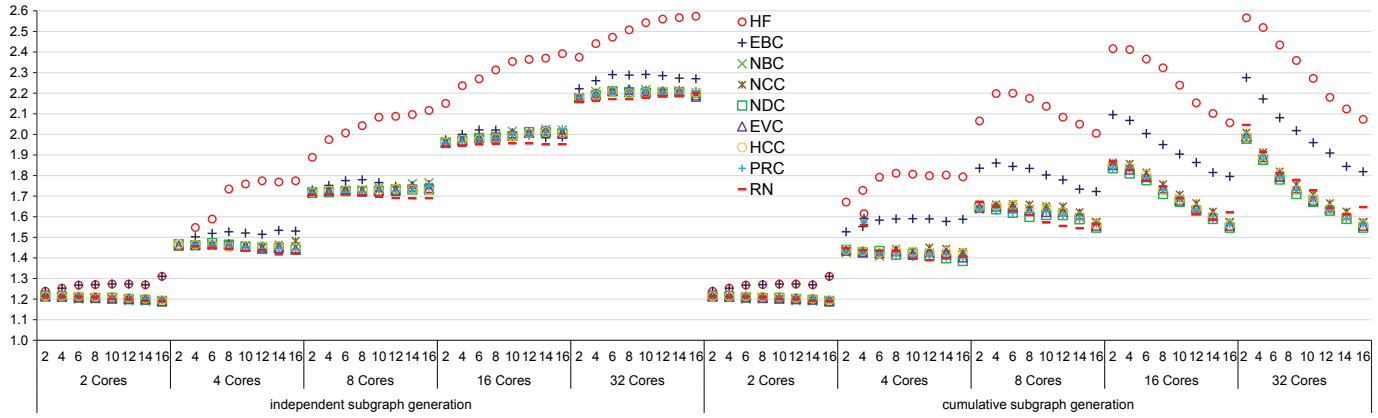

Fig. 2. Average performance of MCPR heuristics over all topologies.
x-axis indices: The first line shows the removal percentage of network nodes/edges (between 2% and 16% by increments of 2). The second line shows the number of cores (i.e., 2, 4, 8, 16, 32 cores). The third line indicates the substrate creation approach.

32 cores. Fig. 2 presents the performance gain of the heuristics compared to the single shortest path for the average of Rocketfuel topologies. In the figure, x-axis has three different indicators. First line on the top shows the removal percentage of network nodes/links (which is varied between 2% and 16% by increments of 2). Second line gives the number of cores (i.e., 2, 4, 8, 16, 32 cores) that will accommodate the substrates. The last line indicates the substrate creation approach, i.e., independent or cumulative. Each heuristic yields a higher performance than the single shortest path indicating even with a poorly chosen heuristic (such as the random node removal), MCPR can produce better throughput than traditional single shortest path routing. We observe that MCPR heuristics improve with the number of cores for independent substrate generation. While, in some instances of independent substrate generation (e.g., 16% removal with 32 cores), node centrality heuristics perform slightly worse than the random substrate generation, centrality heuristics overall yield better results than RN. As expected, HF produces best throughput speedup in majority of the instances as it adapts the substrate generation based on network flows.

Having more cores in the routers consistently improves the MCPR performance. In both independent and cumulative methods, a specified percent of the nodes/edges are removed in the next substrate. The independent method removes fewer elements in generating the next substrate as it always uses the actual topology. In Fig. 2, we observe that the cumulative method performs worse with higher node/edge removal rates in subsequent substrates. As there are greater number of nodes/edges ignored in the later substrates, they are not able to yield viable e2e paths. Hence, node centrality metrics occasionally perform worse than RN. On the other hand, with independent substrate generation, higher removal rates yield better performance for the HF approach that adjusts the substrates to the networks' flows. Thus, in general with higher cores, one should utilize the independent edge removal to obtain paths that would provide higher throughput in the network.

Edge-based MCPR heuristics offer a good balance between high throughput and high computational complexity. In the HF approach, small percentage of removals with higher number of cores provides better throughput because HF removes hot spot(s) from the subsequent substrates. However, HF needs to recalculate the substrates after each flow change, which incurs larger computation cost. Hence, it is viable to use EBC that produces the best among centrality heuristics, which incur lower computational complexity as it only recalculates the substrates after a topology change. Overall, for a large topology, edge centrality heuristics, particularly EBC, offer a good trave-off when the number of cores is high and the independent removal percentage within 8% to 10% is used.

TABLE IV
MAXIMUM SPEEDUP ATTAINED ON ROCKETFUEL TOPOLOGIES.

Network	HF independent	HF cumulative	Graph-based independent	Graph-based cumulative
SprintLink	4.26	4.21	3.56	3.13
AboveNet	3.80	3.83	3.41	3.42
Tiscali	2.45	2.70	1.80	1.43
Telstra	2.01	1.95	1.73	1.43
Ebone	1.50	1.49	1.48	1.43
Exodus	1.42	1.40	1.45	1.37

The speedup attained by MCPR is in high correlation with the node count, edge count, and average degree of the underlying network topologies. When analyzing individual topologies, in Table IV, we get from 1.45 to 4.26 times better results than the single shortest path baseline. We observe that the best overall performance is achieved with SprintLink and AboveNet, two largest (in terms of node and edge counts) networks in the data set. We observe smaller speedup on small topologies compared to bigger ones since there are not enough elements to remove from the topology. In our experiments, Exodus is the smallest topology with 79 nodes and 352 edges with the highest assortativity coefficient, which means that the topology is very balanced, unlike a star topology. Single shortest path routing attains a well balanced traffic routing leaving little room for improvement. MCPR attains 1.45 times better throughput than the shortest path approach on Exodus. However, some of the graph-based heuristics perform better than HF because of the assortative characteristics and the size of the network. When we analyze the next smallest topology,

TABLE V
QUALITATIVE COMPARISON OF MCPR HEURISTICS.

Graph-based	Flow-based
Depends on topology	Depends on flows
Pre-computed	Dynamic
Coarse granularity	Fine granularity

Ebone, the maximum speedup that MCPR attained is 1.5 with HF and 1.48 with the graph-based heuristics. Exodus and Ebone topologies have the same average path length; however, the maximum degree is 51 for Ebone which makes the topology disassortative. Thus, HF performs slightly better than the graph-based heuristics.

Topological properties of the underlying network plays a significant role in MCPR's performance. Telstra and Tiscali are star-like topologies with a high-degree core. MCPR's throughput speedup is 2.01 and 2.45 for Telstra and Tiscali, respectively. Tiscali has a slightly disassortative behavior where highest degree nodes form a core of the network. Hence, the centrality-based removals are not able to produce viable substrates, and so, HF attains better throughput. Thus, Tiscali shows the most significant difference between the HF and other centrality metrics. Fig. 3 presents the effect of the number of cores in the networks with 8% independent edge removal for AboveNet, Tiscali, and Telstra.

C. Control Plane Overhead

Implementation of the MCPR heuristics will add new overhead compared to the legacy shortest-path routing. This overhead includes both the computation of the virtual topologies as well as the messaging needed to install and maintain them.

1) Computational complexity: Graph-based heuristics are based on the underlying topologies and infer which nodes would be congested first by analyzing network centrality metrics. Graph-based techniques are performed if the topology is changed such as addition or failure of edges or nodes. On the other hand, flow-based heuristics are based on data flows and adapt to the current flows. Hence, they require additional computations with each flow change. On the positive side, the flow-based heuristics (e.g., HF in Fig. 2) achieve higher aggregate throughput as they balance the load based on the actual flows. Table V summarizes these tradeoffs between the two heuristic approaches.

Overall, flow-based heuristic periodically adapts to the flow patterns and is sensitive to the current network traffic during the new substrate generation process. Hence, the HF gives an upper bound for MCPR heuristics. Although HF is the upper bound, it should be calculated each time when the flow pattern changes and will increase control plane communication to disseminate changes. Graph-based heuristics, however, give pre-computed substrates and are independent of current flow status. Table VI compares the computational complexities of the heuristics where n is the number of nodes and m is the number of edges in the underlying network, and $s = |Q|$ is the number of substrates being generated.

2) Messaging and memory complexity: The amount of control messages that have to be transferred to routers will depend on (i) the size of the virtual topology information and (ii) the frequency of updates on the virtual topologies. Since each virtual topology will include link-state information, its size is going to be $\mathcal{O}(m)$ in bits. For all the MCPR heuristics, the memory overhead for each router will be $\mathcal{O}(sm)$ bits as they have to store the link-state information for each virtual topology. Assuming the underlying topology changes at every t_t seconds, the messaging overhead of the node- or edge-based MCPR heuristics is $\mathcal{O}(sm/t_t)$ in bps. Likewise, assuming that the traffic flows experience a sizable change at every t_f seconds, the messaging overhead of the flow-based MCPR heuristics is $\mathcal{O}(sm/t_t+sm/t_f)$ in bps. These overhead bounds are shown in Table VI for each MCPR heuristic. Effective messaging overhead will depend on the specifics of protocol implementation and, more importantly, balancing of the trade-off between the proactiveness of routing protocol and the amount of network capacity spent on carrying the control messages. Most legacy protocols, such as OSPF, handle this trade-off by utilizing fixed timers to suppress the amount of routing control messages in the network.

TABLE VI
CONTROL OVERHEAD OF MCPR HEURISTICS.

Heuristic	Computation complexity (s)	Memory complexity (bits)	Messaging complexity (bps)
NDC	$\mathcal{O}(n+m)$	$\mathcal{O}(sm)$	$\mathcal{O}(sm/t_t)$
NBC	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(sm)$	$\mathcal{O}(sm/t_t)$
NCC	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(sm)$	$\mathcal{O}(sm/t_t)$
HCC	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(sm)$	$\mathcal{O}(sm/t_t)$
EVC	$\mathcal{O}(n^3)$	$\mathcal{O}(sm)$	$\mathcal{O}(sm/t_t)$
PRC	$\mathcal{O}(n^2)$*	$\mathcal{O}(sm)$	$\mathcal{O}(sm/t_t)$
RN	$\mathcal{O}(1)$	$\mathcal{O}(sm)$	$\mathcal{O}(sm/t_t)$
EBC	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(sm)$	$\mathcal{O}(sm/t_t)$
ECC	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(sm)$	$\mathcal{O}(sm/t_t)$
HF	$\mathcal{O}(sfn^2 \log n)$	$\mathcal{O}(sm)$	$\mathcal{O}(sm/t_t + sm/t_f)$

*Can be approximated in $\mathcal{O}(n+m)$.

D. Graph Analysis of Substrates

Analyzing the changes in the graph properties of the substrates being generated by MCPR heuristics sheds light into the performance outcomes. We find that node centrality heuristics impact the graph properties significantly more than the edge centrality and flow-based heuristics. Fig. 4 presents the average and maximum node degrees of substrate graphs generated by the MCPR heuristics. The node degree statistics are averages over all Rocketfuel topologies. We present the node degree statistics for independent as well as cumulative substrate generation methods. Cumulative substrate generation significantly changes graph characteristics and reduces connectivity in the graph. We compare the best of the three heuristic groups: NBC, EBC, and HF. As it is expected that the node centrality heuristics disrupt the substrate graphs more than edge centrality and HF heuristics, NBC reduces the node degree much faster than the other two heuristics. Node centrality heuristics have almost the same effect in terms of topology disruption. We observe that small removal percentages have small effects on substrate characteristics. However, in larger removal percentages MCPR is not able to generate different substrates and repeats itself after a threshold.

In node centrality heuristics, the first substrate removes the most centralized nodes, which causes significant decrease in

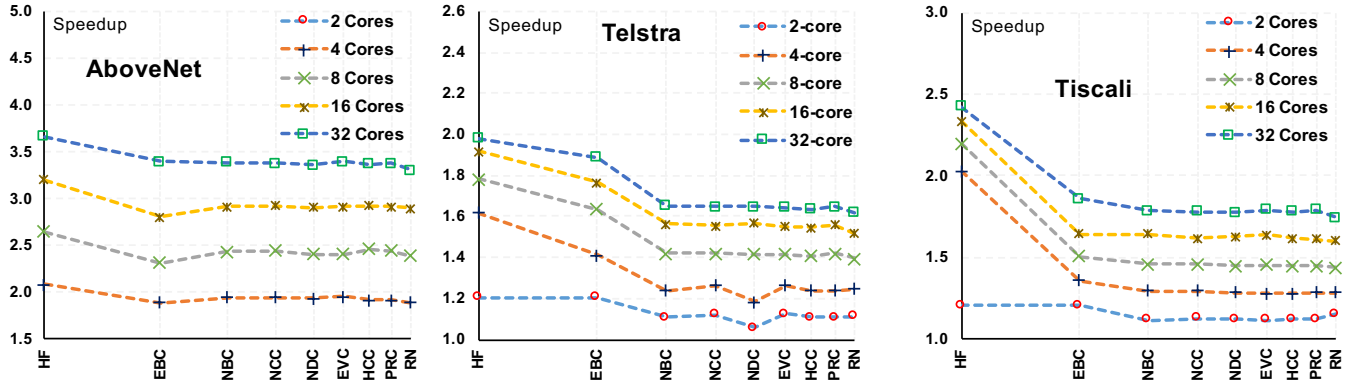

Fig. 3. Effect of the number of cores with 8% independent removal for different topologies.

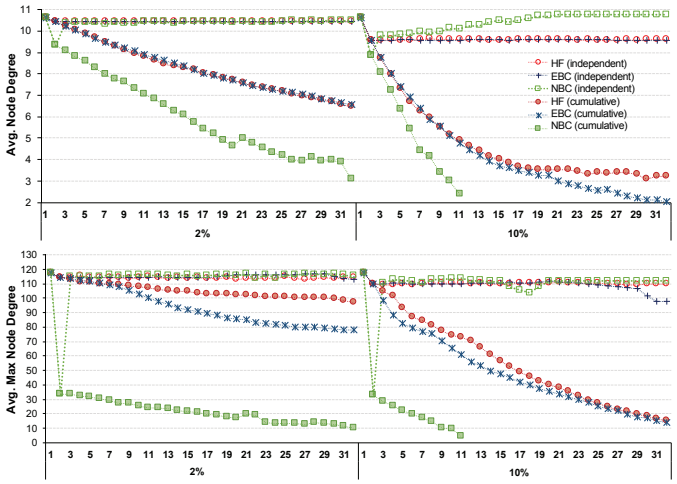

Fig. 4. Average and maximum node degree of the substrates across all topologies.
x-axis indices: The first line shows the number of substrates generated from 1 to 32, with 1 corresponding to the original (single shortest-path) network. The second line shows the removal percentage of network nodes/edges (2% and 10%).

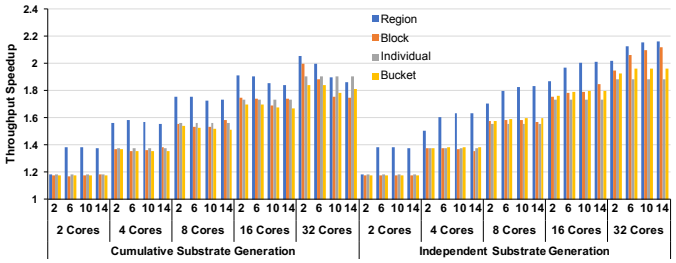

Fig. 5. Comparison between the removal approaches (block, bucket, region, individual) for EBC.
x-axis indices: The first line shows the removal percentage of network nodes/edges (between 2% and 14% by increments of 4). The second line shows the number of cores (i.e., 2, 4, 8, 16, 32 cores). The third line indicates the substrate creation approach.

the average and maximum node degrees. After a point, MCPR heuristics start to remove periphery nodes from the network and this increases the average node degree. This indicates that later substrates might not improve the performance as removing periphery nodes does not contribute to balancing the load. Average node degree, however, is not affected much by the edge removal heuristics. As a result, edge removal heuristics perform better than node centrality heuristics (in Fig. 2). We can observe the same pattern for maximum node degree.

E. Comparing Removal Approaches

Until this subsection, we presented the results for block removal approach. We now compare the performance of the removal techniques, i.e., bucket, region, individual, and block removal approaches. Since EBC outperforms the other centrality-based heuristics, we only show the comparison for EBC in Fig. 5.

When we generate substrates cumulatively, the new substrates are being disrupted rapidly which sets back the total throughput. However, in the individual removal method, we remove only one element at each step that lessens deformation on the substrates but improves the performance slightly. Bucket removal performs better and reduces the topology disruption for the cumulative method, but it does not entirely clear congested spots. So, it does not sufficiently contribute to the throughput speedup. Block removal omits a block of nodes/edges to generate a new substrate. This removes the most important elements in the first one, however, in later substrates, it removes low-ranking elements which reduces the performance significantly. Finally, region removal outperforms all other heuristics because, in each substrate, it clears a neighborhood of elements corresponding to hot spots. This spatially correlated removal proves to be effective since traffic congestion in networks tend to be highly localized.

In the independent removal substrate generation, region and block perform better as in the cumulative process. The Individual approach has moderate improvement, and Bucket does not have significant contribution to the performance. In bucket removal, we omit some elements in each block which is unable to balance the load across the blocks. When we remove the whole block, in block removal, we reduce the load on most used nodes synchronously attains better load balancing. Hence, block performs better than bucket. Besides, region clears the neighborhood, so the overloaded nodes and the neighborhoods around them will not exist on the newly generated substrates. This results in higher throughput for the early substrates. When we increase the number of cores, for the later substrates, region will remove the area of the low-ranking nodes as well and hence degrade the performance severely. In

Fig. 5, we also show that region performs better than block, but it is a dynamic approach, and the implementation is costly. Even though region performs better in our later tests, we show block results in the previous sections since it might be easily adapted without additional implementation costs.

F. Effect of Heterogeneous Core Count

We test all topologies for heterogeneous scenarios with a selected set of parameters for core and removal percentages under independent substrate generation since it has a better performance. We only show the performance of EBC and NBC in Fig. 6 to demonstrate how MCPR performs under heterogeneous core distribution on average among all Rocketfuel topologies.

We show the performance of PES and DLM methods. We indicate the upper and lower bounds with a thick line. The throughput of the homogeneous scenario is the upper limit of the heuristics' performance when all of the nodes have enough capability to process all of the assigned substrates without any data loss. However, in real environments, there could be processing delay that causes significant data loss. The PES method eliminates the nodes with fewer cores than the number of substrates being assigned to them. Since this elimination is done after the substrate is created, the technique removes additional elements after all and this reduces the total throughput. The DLM method with *arithmetic* (DLM-A), *harmonic* (DLM-H), and *product* (DLM-P) features show the speedup with a loss of data according to the loss calculation. Although the DLM-P (and in some cases PES for node removal) is the lower bound of the MCPR, it still has an improvement against the single shortest path baseline. Note that, in DLM, we add some additional processing delay harshly which causes data loss for those flows going through the nodes with fewer cores than the number of substrates running on them. So, MCPR may give better results in real settings.

G. Effect of Failures

Failure analysis helps to determine the robustness of the approach under dynamic network conditions. We also analyze MCPR's performance when the network has changed unexpectedly because of traffic spikes or sudden failures. When network failures occur, multiple e2e paths should be recalculated due to topology changes. Re-computation process means extra overhead on both control-plane communication and CPU usage. Also, most of these failures are short-term and failed edge or node could be back online in a while. In that case, all paths will be recalculated twice to determine the paths between source and destination pairs. However, in MCPR, shortest path calculation is enough to modify the paths on each substrate when node or edge failures occur. To show the robustness performance of MCPR, we calculate the total throughput over the network when one node or edge is down, and we compare the average throughput achieved. We compare the average results to understand the *single node failures and single edge failures*. We assume that the substrate graphs stay still even though a node/edge fails. This allows us to see how robust the substrate graphs are against single failures.

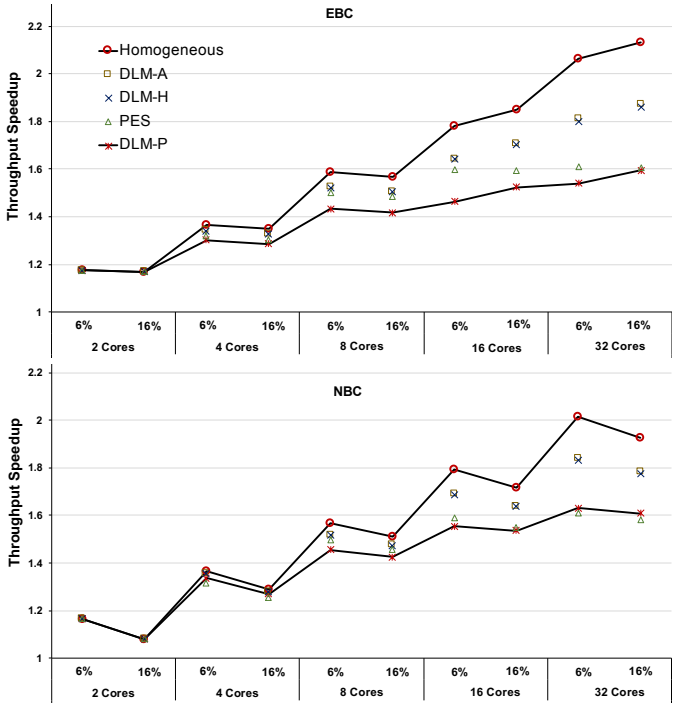

Fig. 6. MCPR performance with EBC (top) and NBC (bottom) under the heterogeneous core distribution for independent substrate generation.
x-axis indices: The first line shows the removal percentage of network nodes/edges (6% and 16%). The second line shows the number of cores (i.e., 2, 4, 8, 16, 32 cores).

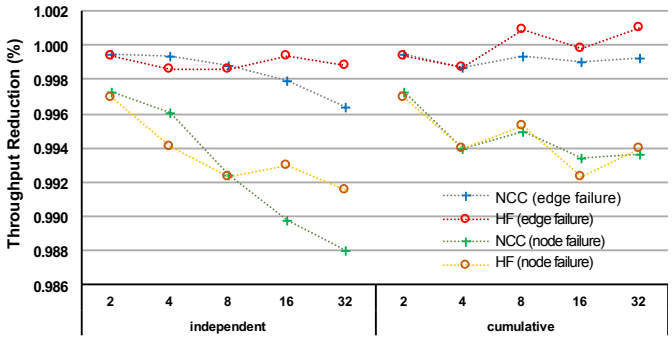

Fig. 7. Performance of AboveNet with node/edge failures when 8% removal is applied.
x-axis indices: The first line shows the number of cores (i.e., 2, 4, 8, 16, 32 cores). The second line indicates the substrate creation approach.

MCPR heuristics handle both edge and node failures quite well. We consider the best and worst performing heuristics, that are HF and NCC respectively. Fig. 7 shows the average decrease in throughput when 6% and 8% removal rate is used with NCC and HF approaches in the AboveNet topology. Overall, we observe that performance of MCPR is reduced by 0.4% when a node or edge failure occurs. When considering the node failure, both NCC and HF approaches loose 0.7% and 0.6% throughput on average for 6% and 8% removal rates, respectively. When considering the edge failures, NCC looses 0.2% throughput on average while HF looses 0.1% throughput with 6% removal rate. Similarly, with 8% removal rate, the average throughput loss is 0.1%. Note that HF even improves the throughput with edge failures under independent substrate

graph generation with 8, 16 and 32 cores. This is due to the fact that HF adjusts to the network condition and, with failure of an edge, it was able to find alternative paths that slightly improved the overall throughput.

VII. Summary and Future Work

We presented a new multi-path routing framework, MCPR, that uses graph abstraction of the network topology and employs network centrality calculations to generate subgraphs for multi-core routers. The basic idea is to virtually slice the router topology into different subgraphs and assign each to a separate router core, which calculates the classical shortest paths on the assigned subgraph. This eases the computational complexity of multi-path routing by dividing the overall problem into smaller ones and lending each subgraph to a separate CPU core with traditional shortest path algorithms. We analyzed MCPR's theoretical background, and showed that MCPR is an NP-hard problem.

Experimental results show that centrality based heuristics are able to increase overall throughput in the network more than twice with 8-core routers compared to the current single shortest path approach. The throughput speedup attained by MCPR heuristics range from 1.3 with 2-core routers to 2.6 with 32-core routers, all while incurring polynomial control overhead. In general, the results showed that the throughput speedup is in strong correlation with the size (node/edge count) and connectivity (average node degree) of the underlying network. MCPR framework solves the multi-path calculation problem with well-known techniques that easily adapts to the current systems. Normally, multi-path routing algorithms need to calculate e2e paths when topology changes. However, in MCPR, we solely calculate shortest paths, which is a well-known algorithm already built in the routers. We also analyzed the effects of various selection methods after finding the central nodes to remove in substrates. We showed the performance of MCPR when routers have different number of cores considering delay-tolerant and loss-tolerant applications. MCPR attains higher throughput against the shortest path baseline under heterogeneous distribution of cores across network routers. When compared to homogeneous distribution of cores across routers, MCPR heuristics experienced small reductions in the throughput speedup, e.g., from $\approx$1.6 to $\approx$1.4 when the average core count is 8. The MCPR framework is robust to network failures because of its divide and conquer design. The experiments showed that MCPR heuristics suffer from less than 1% reduction in throughput when they face with edge/node failures.

Possible future work includes to create subgraphs with other search algorithms such as genetic algorithms, and improving heuristics with dynamic and static solutions. MCPR can also be run on a test-bed or real environment to show easy adaptation and real performance. MCPR heuristics might be enhanced and kernel level implementation can be done as a new protocol. MCPR's control overhead can be studied by utilizing legacy multi-topology routing standards. Understanding the trade-off between control messaging overhead and efficacy of MCPR in attaining better throughput would be revealing. Optimizing the messaging formats for multi-core routers while compatibility with existing shortest-path routing protocols is a worthy effort to pursue. MCPR framework can be analyzed with different aspects of networking such as by exploring its SDN implementations and the performance of MCPR on centralized and de-centralized systems. The comparison between MCPR and other multi-path routing algorithms can be investigated. Finally, MCPR attempts to solve both multi-path routing and parallelization of multi-path routing algorithms on multi-core routers at the same time. Studying the latter problem on a suite of multi-path routing algorithms on multi-core routers is a worthy future direction.

References

[1] A. Soran, M. Yuksel, and M. H. Gunes, "Multiple graph abstractions for parallel routing over virtual topologies," in *Proc. IEEE INFOCOM Workshops*, 2017.

[2] A. Soran, F. M. Akdemir, and M. Yuksel, "Parallel routing on multi-core routers for big data transfers," in *Proc. ACM CoNEXT Student Workshop*, 2013.

[3] A. Russeler, "Data centre network trends for 2023," https://dcnnmagazine.com/data-centres/data-centre-network-trends-2023, Dec. 2022.

[4] V. Jacobson *et al.*, "RFC 1323: TCP extensions for high performance," 1992.

[5] Y. Kim, S. Atchley, G. R. Vallee, S. Lee, and G. M. Shipman, "Optimizing end-to-end big data transfers over terabits network infrastructure," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 1, pp. 188–201, 2017.

[6] Z. Liu, P. Balaprakash, R. Kettimuthu, and I. Foster, "Explaining wide area data transfer performance," in *Proc. ACM HPDC*, 2017.

[7] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, "TCP extensions for multipath operation with multiple addresses," IETF RFC 8684, Mar. 2020.

[8] S. Barré, O. Bonaventure, C. Raiciu, and M. Handley, "Experimenting with multipath TCP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 443–444, 2011.

[9] C. Raiciu *et al.*, "Improving datacenter performance and robustness with multipath TCP," in *Proc. ACM SIGCOMM*, 2011.

[10] S. Vutukury and J. J. Garcia-Luna-Aceves, "MDVA: A distance-vector multipath routing protocol," in *Proc. IEEE INFOCOM*, 2001.

[11] S. Nelakuditi and Z. L. Zhang, "On selection of paths for multipath routing," in *Proc. IEEE/IFIP IWQoS*, 2001.

[12] B. Tierney, E. Kissel, M. Swany, and E. Pouyoul, "Efficient data transfer protocols for big data," in *Proc. IEEE e-Science*, 2012.

[13] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," in *Proc. ACM SIGCOMM*, 2011.

[14] Y. Wang, S. Su, A. Liu, and Z. Zhang, "Multiple bulk data transfers scheduling among datacenters," *Comput. Netw.*, vol. 68, pp. 123–137, 2014.

[15] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 63–74, 2010.

[16] E. Yildirim, D. Yin, and T. Kosar, "Prediction of optimal parallelism level in wide area data transfers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 12, pp. 2033–2045, 2011.

[17] T. J. Hacker, B. D. Noble, and B. D. Athey, "Improving throughput and maintaining fairness using parallel tcp," in *Proc. IEEE INFOCOM*, 2004.

[18] N. Hanford *et al.*, "Improving network performance on multicore systems: Impact of core affinities on high throughput flows," *Future Generation Comput. Syst.*, vol. 56, pp. 277–283, 2016.

[19] Y. Wu *et al.*, "Orchestrating bulk data transfers across geo-distributed datacenters," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 112–125, 2017.

[20] J. Raju and J. J. Garcia-Luna-Aceves, "A new approach to on-demand loop-free multipath routing," in *Proc. IEEE ICCCN*, 1999.

[21] I. Gojmerac, T. Ziegler, F. Ricciato, and P. Reichl, "Adaptive multipath routing for dynamic traffic engineering," in *Proc. IEEE GLOBECOM*, 2003.

[22] Y. Ganjali and A. Keshavarzian, "Load balancing in ad hoc networks: Single-path routing vs. multi-path routing," in *Proc. IEEE INFOCOM*, 2004.

[23] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM*, 2013.
[24] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *IEEE/OSA J. Lightw. Technol.*, vol. 31, no. 1, pp. 15–22, 2013.
[25] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies," in *Proc. USENIX NSDI*, 2010.
[26] S. Sharma and S. K. Jena, "Cluster based multipath routing protocol for wireless sensor networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 2, pp. 14–20, 2015.
[27] F. du Pin Calmon, J. M. Cloud, M. Medard, and W. Zeng, "Multi-path data transfer using network coding," Jan. 3 2017, US Patent 9,537,759.
[28] S. K. Singh, T. Das, and A. Jukan, "A survey on Internet multipath routing and provisioning," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2157–2175, 2015.
[29] M. Z. Hasan, H. Al-Rizzo, and F. Al-Turjman, "A survey on multipath routing protocols for QoS assurances in real-time wireless multimedia sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1424–1456, 2017.
[30] Y. O. Lee and A. L. N. Reddy, "Disjoint multi-path routing and failure recovery," in *Proc. IEEE ICC*, 2010.
[31] R. Balasubramanian and S. Ramasubramanian, "Minimizing average path cost in colored trees for disjoint multipath routing," in *Proc. IEEE ICCCN*, 2006.
[32] D. Lopez-Pajares, E. Rojas, J. A. Carral, I. Martinez-Yelmo, and J. Alvarez-Horcajo, "The disjoint multipath challenge: Multiple disjoint paths guaranteeing scalability," *IEEE Access*, vol. 9, pp. 74422–74436, 2021.
[33] B. H. Shen, B. Hao, and A. Sen, "On multipath routing using widest pair of disjoint paths," in *Proc. IEEE HPSR Workshop*, 2004.
[34] W. Fan, F. Xiao, H. Cai, X. Chen, and S. Yu, "Disjoint paths construction and fault-tolerant routing in BCube of data center networks," *IEEE Trans. Comput.*, vol. 72, no. 9, pp. 2467–2481, 2023.
[35] Z. ALzaid, S. Bhowmik, and X. Yuan, "Multi-path routing in the jellyfish network," in *Proc. IEEE IPDPSW*, 2021.
[36] N. M. Jones, G. S. Paschos, B. Shrader, and E. Modiano, "An overlay architecture for throughput optimal multipath routing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2615–2628, 2017.
[37] T. Thorsen, S. J. Maerkl, and S. R. Quake, "Microfluidic large-scale integration," *Science*, vol. 298, no. 5593, pp. 580–584, 2002.
[38] Y. Shintani, M. Inagi, S. Nagayama, and S. Wakabayashi, "A multithreaded parallel global routing method with overlapped routing regions," in *Proc. IEEE DSD*, 2013.
[39] T. Li, Y. Ren, D. Yu, and S. Jin, "Ramsys: Resource-aware asynchronous data transfer with multicore systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 5, pp. 1430–1444, 2017.
[40] T. Ganegedara and V. Prasanna, "100+ Gbps IPv6 packet forwarding on multi-core platforms," in *Proc. IEEE GLOBECOM*, 2013.
[41] R. Bolla, R. Bruschi, and P. Lago, "Energy adaptation in multi-core software routers," *Comput. Netw.*, vol. 65, pp. 111–128, 2014.
[42] C. Partridge, "Forty data communications research questions," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 5, pp. 24–35, 2011.
[43] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault, "Multi-topology (MT) routing in OSPF," IETF RFC 4915, Jun. 2007, https://www.rfc-editor.org/rfc/rfc4915.html.
[44] T. Przygienda, N. Shen, and N. Sheth, "M-ISIS: Multi topology (MT) routing in intermediate system to intermediate systems (IS-ISs)," IETF RFC 5120, Feb. 2008, https://www.rfc-editor.org/rfc/rfc5120.html.
[45] S. Bae and T. R. Henderson, "Traffic engineering with OSPF multi-topology routing," in *Proc. IEEE MILCOM*, 2007.
[46] A. Kvalbein and O. Lysne, "How can multi-topology routing be used for intradomain traffic engineering?," in *Proc. ACM SIGCOMM Workshop INM*, 2007.
[47] X. Wang, S. Wang, and L. Li, "Robust traffic engineering using multi-topology routing," in *Proc. IEEE GLOBECOM*, 2009.
[48] N. Wang, K.-H. Ho, and G. Pavlou, "Adaptive multi-topology IGP-based traffic engineering with near-optimal network performance," in *Proc. Netw.*, 2008.
[49] E. Mirzamany, A. Lasebae, and O. Gemikonakli, "An efficient traffic engineering based on multi-topology routing for future Internet," *Comput. Netw.*, vol. 70, pp. 170–178, 2014.
[50] S. Cevher, M. Ulutas, S. Altun, and I. Hokelek, "Multi-topology routing based IP fast re-route for software defined networks," in *Proc. of IEEE ISCC*, 2016.
[51] M. T. Gardner, R. May, C. Beard, and D. Medhi, "Using multi-topology routing to improve routing during geographically correlated failures," in *Proc. DRCN*, 2014.
[52] S. S. W. Lee, K.-Y. Li, and A. Chen, "Energy-efficient multi-topology routing configurations for fast failure reroute in IP networks," in *Proc. IEEE GLOBECOM*, 2012.
[53] T. Čičić, "On basic properties of fault-tolerant multi-topology routing," *Comput. Netw.*, vol. 52, no. 18, pp. 3325–3341, 2008.
[54] N. Huin, S. Martin, J. Leguay, and S. Cai, "Network slicing with multi-topology routing," in *Proc. IFIP Networking*, 2021.
[55] A. Mihailovic, B. Abrishamchi, M. Farhoudi, and A. H. Aghvami, "A comprehensive multi-topology minimum set cover link-state routing approach for emerging random all-IP access network topologies," *Comput. Netw.*, vol. 219, pp. 109418, 2022.
[56] O. K. Demir and S. Cevher, "Multi-topology routing based traffic optimization for IEEE 802.1 time sensitive networking," *Real-Time Syst.*, vol. 59, pp. 123–159, 2023.
[57] J. Moy, "OSPF version 2," *IETF RFC 2328*, Apr. 1998.
[58] D. Banerjee and B. Mukherjee, "A practical approach for routing and wavelength assignment in large wavelength-routed optical networks," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 903–908, 1996.
[59] A.E. Ozdaglar and D.P. Bertsekas, "Routing and wavelength assignment in optical networks," *IEEE/ACM Trans. Netw.*, vol. 11, no. 2, pp. 259–272, 2003.
[60] M. Saad and Z.-Q. Luo, "On the routing and wavelength assignment in multifiber WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 9, pp. 1708–1717, 2004.
[61] B. Mukherjee, *Optical WDM Networks*, Springer, 2006.
[62] M. Kodialam and T.V. Lakshman, "Network link weight setting: A machine learning based approach," in *Proc. IEEE INFOCOM*, 2022.
[63] B. Fortz and M. Thorup, "Increasing Internet capacity using local search," *Comput. Optimization Appl.*, vol. 29, pp. 13–48, 2000.
[64] G. B. Dantzig and D. R. Fulkerson, "On the max-flow min-cut theorem of networks," Numerical Analysis Report, DAMTP Technical Report P-826, The RAND Corporation, Santa Monica, CA, 1955.
[65] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," Tech. Rep., Stanford InfoLab, 1999.
[66] L. Bolc and J. Cytowski, *Search Methods for Artificial Intelligence*, Academic Press, 1992.
[67] C. Raiciu *et al.*, "How hard can it be? designing and implementing a deployable multipath TCP," in *Proc. USENIX NSDI*, 2012.
[68] J. Duan, Z. Wang, and C. Wu, "Responsive multipath TCP in SDN-based datacenters," in *Proc. IEEE ICC*, 2015.
[69] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. ACM SIGCOMM*, 2002.

Ahmet Soran is the Chief Technology Officer at TRK Technology, Turkey. He received B.S. and M.S. degrees in Computer Engineering from TOBB University of Economics and Technology, Ankara, Turkey, in 2009 and 2012, respectively, and Ph.D. degree in Computer Science and Engineering from the University of Nevada, Reno (UNR) in 2017. He was an Assistant Professor in the Computer Engineering Department at Abdullah Gul University, Kayseri, Turkey from 2018 to 2021. His research interests include graph theory applications, communications and networking, device-to-device protocols, network and traffic management, network architecture, cybersecurity, and privacy. He is a Member of IEEE.

Murat Yuksel (SM'15) is a Professor at the ECE Department of the University of Central Florida (UCF), Orlando, FL, and a Visiting Scientist at MIT Lincoln Labs. He served as the Interim Chair of ECE at UCF from 2021 to 2022. Prior to UCF, he was a Faculty Member at the CSE Department of the University of Nevada, Reno, NV. From 2002 to 2006, he was a Member of Adjunct Faculty and a Postdoctoral Associate at the ECSE Department of Rensselaer Polytechnic Institute (RPI), Troy, NY. He received M.S. and Ph.D. degrees in Computer Science from RPI in 1999 and 2002, respectively. He worked as a Software Engineer at Pepperdata, and a Visiting Researcher at AT&T Labs and Los Alamos National Lab. His research interests are in the areas of networked, wireless, and computer systems with a recent focus on wireless systems, optical wireless, spectrum sharing, network economics, network architectures, and network management. He has been on the editorial boards of Computer Networks, IEEE Transactions on Communications, IEEE Transactions on Machine Learning in Communications and Networking, and IEEE Networking Letters. He has published more than 200 papers at peer-reviewed journals and conferences, and is a co-recipient of five Best Paper, one Best Paper Runner-up, and one Best Demo Awards. He is a Senior Member of IEEE and ACM.

Mehmet H. Gunes is a Senior Software Engineer at Akamai Technologies, USA. He was an Associate Professor at Stevens Institute of Technology. He was a Faculty at the University of Nevada, Reno after obtaining a Ph.D. degree in Computer Science from the University of Texas at Dallas in 2008. His research expertise includes complex networks, cybersecurity, and Internet measurements. His research has been funded by the National Science Foundation, National Institute of Justice, Department of Defense, Amazon AWS, Argonne Lab, Nevada Homeland Security, and Cincinnati Center for Pediatric Genomics.