Multiple Graph Abstractions for Parallel Routing over Virtual Topologies

Ahmet Soran
Computer Science and Engineering
University of Nevada, Reno
asoran@cse.unr.edu

Murat Yuksel
Electrical and Computer Engineering
University of Central Florida
murat.yuksel@ucf.edu

Mehmet Hadi Gunes
Computer Science and Engineering
University of Nevada, Reno
mgunes@unr.edu

***Abstract*—High throughput data transfers across the Internet has become a challenge with deployment of data centers and cloud platforms. In this paper, we propose to utilize the cores of a router to build multiple abstractions of the underlying topology to parallelize end-to-end (e2e) streams for bulk data transfers. By abstracting a different graph for each core, we steer each core to calculate a different e2e path in parallel. The e2e transfers can use the shortest paths obtained from each subgraph to increase the total throughput over the underlying network. Even though calculating shortest paths is well optimized in legacy routing protocols (e.g., OSPF), finding optimal set of subgraphs to generate non-overlapping and effective multiple paths is a challenging problem. To this end, we analyze centrality metrics to eliminate potentially highest loaded routers or edges in the topology without coordination and eliminate them from the subgraphs. We evaluate the heuristics in terms of aggregate throughput and robustness against failures.**

I. Introduction

The amount of data to be processed at computers and/or *transferred* across the Internet is growing at a pace of 50% per year [1]. As big data is emerging, the desire to centralize the processing of the data and transfer of it in more controlled settings have prevailed. Arguably, these have been the major causes for the advent of cloud computing and data center networking technologies. In line with these trends, a major problem emerged: How to effectively and efficiently transfer big data transfers exceeding 100Gbps across and within data centers? The legacy end-to-end (e2e) transfer techniques are not adequate to reach such speeds due to lack of performance on data connections over long-distance or high-bandwidth networks [2]. When single path is used for data transfers, the aggregate e2e transfer rate will be limited to the bottleneck on that path. Parallel data transfers can spread the data flow over the network in a non-overlapping manner, and hence improve the aggregate throughput. The downside is that such parallel transfers require *multi-path routing capability*. Availability of such e2e paths allows parallel data streams to be fed onto different paths and thereby attain a higher utilization of the underlying network which is not possible by legacy single shortest-path routers [3]. The approach is also proposed for anonymous communication [4].

According to recent studies [5], the multi-path parallel streaming approaches are successful in attaining better load balancing and network utilization. The key focus of these techniques is to diversify and spread e2e paths while satisfying constraints such as delay and loss. Since the problem is complex, most multi-path routing work boil down to heavy (pre)computations of the paths. However, practical protocols that can offer multiple short paths while effectively handling *network dynamics and failures* are missing.

In this paper, we explore a multi-path routing framework, *Multi-Core Parallel Routing (MCPR)*, that explicitly considers *multi-core routers* and employs shortest-path calculations and graph centrality metrics. Over the last two decades, the routers with multi-core CPUs have become widely deployed. MCPR leverages such multi-core 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 part to a separate CPU core.

Rather than solving the multi-path routing problem all at once, MCPR transforms it into two subproblems: (i) slicing subgraphs from the router topology so that the collection of the shortest paths on each subgraph is diverse and minimally-overlapping, and (ii) calculate shortest paths on each subgraph. Since the latter problem is already handled in legacy routers, MCPR can easily be adopted in practice if the former problem is solvable. To this end, we explore the utility of different centrality metrics in identifying potentially loaded routers and edges without coordination and categorically removing them from the subgraphs.

Section II covers related work on routing techniques and multi-core networking solutions. Section III presents the multi-path routing to improve data transfers using multi-cores. Section IV introduces centrality based heuristics for subgraph generation. Section V presents experimental setup and results. Finally, Section VI concludes the paper.

II. Related Work

Attaining high data transfer speeds require handling of load balancing and congestion management jointly. To reach 100+ Gbps throughput levels, a plethora of efforts has been made in utilizing the existing networks. Techniques explored include smart scheduling over a single [6] or multiple [7] paths, getting congestion notifications from the network [8], running multiple TCP streams each with its own congestion control loop [9], and parallel sub-streams with a joint control loop such as MPTCP [5], having multi-core abilities to improve network performance [10], using SDN to manage traffic among geo-distributed inter-data centers during the data migration [11].

These approaches are heavily vested on the availability of diverse, non-overlapping and robust e2e paths that are easy to compute and deploy. Such multi-path routing [12] has been a popular topic for researchers to solve the load balancing problem over a network. 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 dynamic provisioning in optical backbones [13] and data-centers [14]. Given a large set of paths, path selection [3] was also studied to find the best subset of paths. A key challenge with multi-path routing has been the adaptation across the Internet because of computation costs and lack of infrastructure knowledge. MCPR considers both of these challenges in its design. In [15], we explored the routing problem as a set of smaller scale local problems without requiring a global coordination to enable flexible inter-domain routing.

III. Multi-Core Parallel Routing (MCPR)

Legacy routing systems 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 more throughput depends on fast, scalable and robust calculations of multiple paths between source-destination pairs. Since such multi-path calculations are too complex, MCPR takes a divide-and-conquer approach. It divides the multi-path routing problem into multiple shortest path calculation problems by abstracting the underlying network topology as different subgraphs, and uses legacy shortest path calculation algorithms to swiftly and efficiently calculate/conquer each subgraph.

The main idea of MCPR is that different slices (i.e., subgraphs) of the router topology are given to each router core and the e2e data transfer is split onto shortest paths calculated on each subgraph topology. Once assigned to a subgraph S_i, a flow will follow the shortest path calculated by that S_i. Note that all e2e flows will be using the same underlying physical topology regardless of which subgraph they are assigned to. MCPR leads to a system where "parallel" routes are produced on virtual subgraphs over the same physical topology. Many of the current routers have multi-CPU cores which can be utilized to execute the conquer phase of MCPR. Different than most of the 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 shortest paths calculated by each core. Then, it is up to the e2e transport protocols such as MPTCP to decide which one of these paths from the subgraph topologies to use and with what rate.

A. Motivating Scenario

Figure 1 illustrates a scenario where two virtual subgraphs are produced. Subgraph 0 is equivalent to the real router topology, whilst Subgraph 1 is generated by removing node 3 from Subgraph 0. Even though there are many other possible paths available over the network, current systems would carry

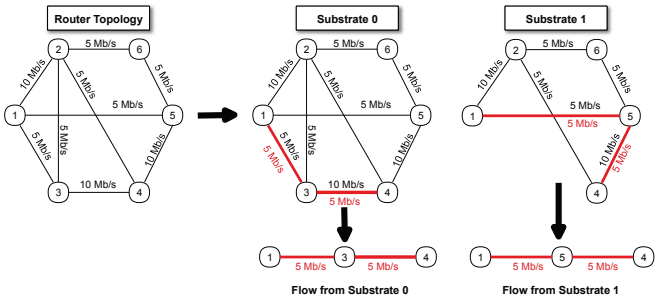

Fig. 1. Motivating scenario with two cores.

out 5 Mb/s on a single path. But, if these two topologies were given to a two-core router and *current path finding algorithms* were executed in parallel on different cores then each path could transfer 5 Mb/s data. Compared to the current systems, each subgraph's shortest path is different and both collectively yield a total of 10Mb/s throughput from node 1 to 4 for *two cores* scenario. In addition to removing node(s), some edge(s) can also be omitted, such as the edge connecting node 1 and 3, while creating a new subgraph. In that case, there would also be two different paths to reach the destination node 4, paths (1-3-4) and (1-2-4).

B. Design Approach and Principles

Rather than solving the multi-path routing problem all at once, our approach transforms it into two subproblems: (i) slicing out subgraphs from the router topology so that the collection of the shortest paths on each subgraph is diverse and produces non-overlapping e2e paths, and (ii) calculate shortest paths on each subgraph. Since the latter problem is already handled in legacy routers, our approach can easily be adapted to current routers if the former problem is efficiently solved. The MCPR design principles and main features are:

- *Shortest path calculation 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 easy to deploy as well.
- *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.
- *Robustness to network dynamics.* 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 subgraph and lets the shortest path routing algorithm on each subgraph perform the re-calculation.
- *Subgraph generation can be centralized.* The divide part of MCPR is the most challenging as it requires finding the best set of subgraphs so that their shortest paths minimally overlap. This potentially requires a heavy computation task. The advantage is that such heavy computation can be done in software-defined networking (SDN) controllers or other centralized locations with high computation power. Centralizing subgraph generation allows optimizations such as multi-path traffic engineering which require a global and centralized view

of the network. Furthermore, failures or network dynamics do not necessitate the subgraph generation to be done within the routers themselves since re-calculations of shortest paths can be independently done by each core. The subgraph generation could be done at larger timescales without any major sub-optimality.

C. Subgraph Generation Problem

In this subsection, we provide a formal definition of the subgraph generation problem of MCPR. Given an underlying network topology as a directed graph $G = \{V, E\}$ with a set of vertices $|V| = n$ and edges $|E| = m$, MCPR's subgraph generation involves several decision parameters: (i) *number of subgraphs* to generate and (ii) *edge weights* on each subgraph. Let $w_{uv} = 0..k-1$ be the weight of the edge from vertex u to v in G, where w_{uv} can be set to k different integer values. Further, let S be the set of all possible subgraphs of G. Subgraph $S_i \in S$, can be expressed as $S_i = \{V_i, E_i\}$ where $V_i \subseteq V$ and $E_i \subseteq E$. Then, the number of possible subgraphs is $|S| = k^m$.

The overall goal of MCPR is to maximize the throughput of the network by using the shortest paths from a subset of the subgraphs. Let $q \subseteq S$ represent a group of subgraphs and $P(q)$ be the collection of shortest paths generated from the subgraphs in q. Further let $T(q)$ be the total throughput attained from the shortest paths $P(q)$. Then, we can formulate MCPR's problem of generating a group/set of subgraphs that maximizes the throughput, MAX_SUBGRAPH_SET, as follows: $\max_{q \subseteq S} \quad T(q)$ subject to $\exists \quad (u \rightarrow v) \in P(q) \quad \forall \quad u, v \in V$ where $(u \rightarrow v)$ is a path from node u to node v.

It is possible to express the 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 subgraph generation problem in terms of minimizing overlap.

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 subgraphs $q = \{G_1, G_2, .., G_j\}$ in MCPR, we can express the number of shortest paths traversing l as $t(l, P) = \sum_{g \in P} R_g(l)$

The subgraph generation problem of MCPR is, then, to make the usage of each link as evenly as possible, which also implies a minimal overlap among shortest paths. To factor in the varying number of subgraphs, we can aim to minimize the difference between the minimum and the maximum $R_g(l)$. Hence, we write MCPR's subgraph generation problem as a minimization of the unevenness in the usage of links, MIN_SUBGRAPH_SET, in G=$\{V, E\}$: $\min_{q \subseteq S} \quad (\max_{l \in V} t(l, q) - \min_{l \in V} t(l, q))$ subject to $\exists \quad (u \rightarrow v) \in P(q) \quad \forall \quad u, v \in V$.

MIN_SUBGRAPH_SET provides guidance on how heuristics should be designed for the subgraph generation problem. In subgraph generation, we will use this guidance to minimize the maximum load on individual links while trying to maximize the aggregate throughput.

IV. Heuristics

MCPR produces new virtual subgraphs from the underlying router topology. The main aim is to get most diversified e2e paths on each subgraph. For a given network with E edges, 2^E different subgraphs can be generated without any constraints. However, some of the subgraphs would yield non-connected graphs. To assure connectivity, our heuristic design keeps the actual topology as the first subgraph and aims to find the best possible set of additional subgraphs that yield the most diverse and non-overlapping shortest paths.

Instead of searching all possible sets to find the best solution, parallel routing utilizes centrality metrics to generate subgraphs. Our heuristics analyze the given topology to predict which routers/edges could be maxed out earlier than others. These routers/edges will be removed from some subgraphs to balance the load in the underlying topology. Intuitively, one expects to max out the nodes and edges that are typically more centralized. In Graph-based heuristics, we use network centrality metrics to find the most central nodes/edges in the network. These nodes will be removed from the next subgraph to distribute load over the network. In Flow-Pattern-Based heuristics, we omit the most used edges first to produce a new subgraph.

Further, we use two different approaches while creating a new subgraph. In *Cumulative approach*, all subgraphs are generated from the previous one sequentially. In this method, subgraphs get smaller with each generation. In *Independent Approach*, all subgraphs are generated from the given network topology. As each subgraph will calculate its own shortest paths independently, some edges could be shared by multiple paths determined by different subgraphs.

Our heuristics are based on the centrality characteristics of the nodes in the network topology. Central nodes are calculated and selected as potential congestion areas without detailed analysis of the actual network flow. They can be calculated once and recomputed only when the topology changes. We utilized following centrality metrics to determine which nodes to remove:

- **EBC**: Edge Betweenness Centrality
- **NBC**: Node Betweenness Centrality
- **DC**: Degree Centrality
- **EVC**: Eigen Vector Centrality
- **PRC**: Page Rank Centrality
- **CC**: Closeness Centrality
- **HCC**: Harmonic Closeness Centrality

V. Results

In this section, we evaluate the performance of centrality based heuristics to solve subgraph generation for MCPR.

A. Experimental Setup

We performed a static analysis that calculates the total throughput for a given network. We compared the MCPR against the currently used single shortest path routing to analyze the throughput across the network. We generated network flow between all node pairs based on the gravity model between point of presences (PoPs) of AS topologies.

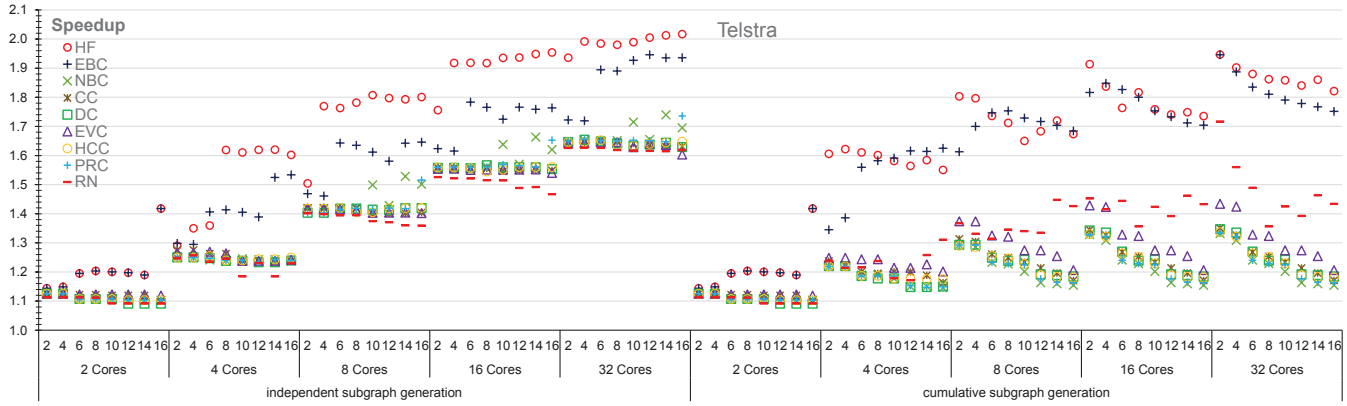

Fig. 2. Performance of MCPR heuristics for the Telstra topology

TABLE I
CHARACTERISTICS OF NETWORK TOPOLOGIES

Network	Nodes	Edges	Max Deg	Avrg Deg	Avrg 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

In the gravity model, flow demands are calculated as the product of populations divided by the square of the geo-distance between two PoP locations. We tested our heuristics on six Rocketfuel topologies that have different characteristics [16]. Table I presents the number of nodes, number of edges, maximum degree, average degree, average path length, clustering coefficient, and assortativity of the network graphs.

In Rocketfuel topologies, link capacities are not provided but delay-based edge weight is provided. In our analysis, we determine link capacities inversely proportional to the link weights in the Rocketfuel data set. In order to calculate the throughput attained by subgraphs, we assign data loads incrementally. First, we send maximal data on the first subgraph, rearrange capacities and use the next subgraph(s) to send the rest of the data. A finer grained model could look for maximum flow in the underlying subgraphs.

In order to asses centrality based metrics, we added two heuristics (i) Highest Flow (**HF**) to determine potential maximal flow, and (i) Random Network (**RN**) to determine whether centrality heuristics are any better than random edge removals. In the Highest Flow, try to adopt to the dynamism in the network traffic by considering the current utilization of edges. HF heuristic determines the active flows and omits the most used edges in the topology to generate subgraph. This approach provides short paths that avoid congestion spots. Hence, HF would achieve higher aggregate throughput with a better load balancing performance. In HF, however, subgraphs need to be recalculated periodically when a new flow is introduced. While it is adaptive to the network dynamics, it is computationally intensive.

B. *Speedup Against Single Shortest Path*

To analyze the speedup of heuristics, we analyze the performance of each against the single shortest path as a baseline. Note that, *highest flow* heuristic is expected to provide the best speedup as it dynamically adopts to the network flow in subgraphs. Similarly, *random network* is expected to provide the lower bound of speedup as it randomly removes nodes from the subgraphs.

Figure 2 presents the performance gain of subgraph generation heuristics compared to the single shortest path for Telstra topology. In the figure, x-axis has two different indicators. First line on the top shows the removal percentage of network nodes/links (which is incremented between 2% and 16% by 2). Second line gives the number of cores (i.e., 2, 4, 8, 16, 32 cores) that will generate subgraphs. The last line indicates the subgraph creation approach, i.e., independent and cumulative. Test results are given for gravity-based flow patterns which is proper for inter-data center traffic flows. From Figure 2, we observe that MCPR performs much better than the single core routing (ranging from 1.2 times with 2 cores to 2.0 times with 32 cores).

Figure 3 presents the average performance gain of subgraph generation heuristics compared to the single shortest path for all topologies. Each heuristic yields a higher performance than the single shortest path approach indicating even with a poorly chosen heuristic (such as random node removal), MCPR can produce better throughput than traditional routing. We observe that MCPR heuristics improve with the number of cores. While in some instances of independent subgraph generation (e.g., 16% removal with 32 cores), node centrality heuristics perform slightly worse than random network generation, centrality heuristics overall yield better results than random removals. Additionally, highest flow heuristic produces best throughput speedup in majority of the instances as it adopts the subgraph generation to network flows.

In both independent and cumulative methods, a specified percent of nodes/edges are removed in the next subgraph. As a result, the independent method removes fewer elements in generating the next subgraph. In Figure 3, we observe that the cumulative method performs worse with higher node/edge

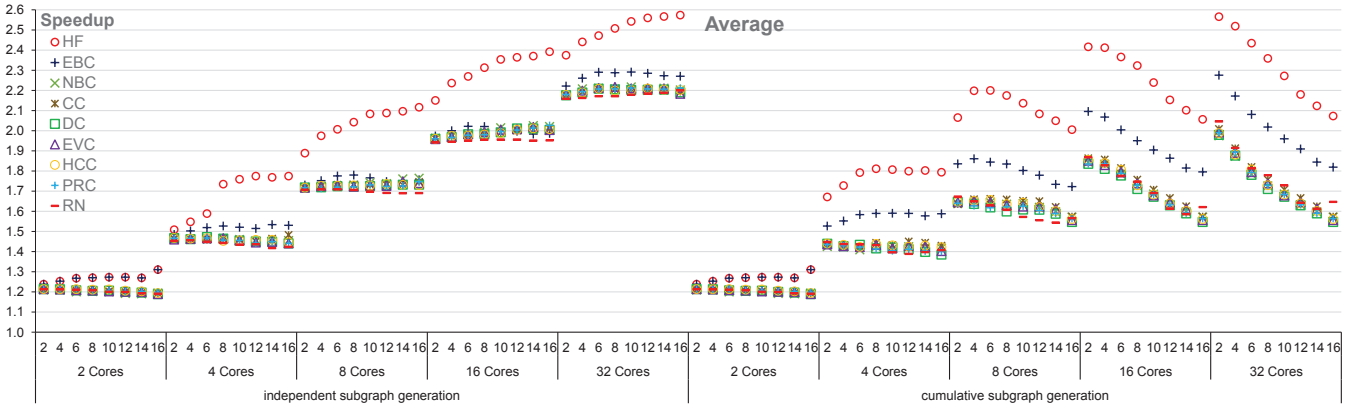

Fig. 3. Average performance of MCPR heuristics for all topologies

removal rates in subsequent subgraphs. As there are greater number of nodes/edges ignored in the later subgraphs, they are not able to yield viable e2e paths. Hence, node centrality metrics occasionally perform worse than random node removal. On the other hand, with independent subgraph generation, higher removal rates yield better performance for the HF approach that adjusts the subgraphs to the networks' flow. Hence, in general with higher cores, one should utilize the independent edge removal in the subgraphs to obtain viable paths that would provide higher throughput in the network.

In the highest flow approach, small number of removals with higher number of cores provides better throughput because HF removes hot spot(s) from the subsequent subgraphs. Note that, however, highest flow approach needs to recalculate subgraphs after each flow change. Hence, it is viable to use edge betweenness centrality that produces the best among centrality heuristics. Overall, for a large topology, edge centrality heuristics are better with high number of cores and independent removal percentage within 8% to 10%.

When analyzing individual topologies, we observe that the best overall performance is achieved with the SprintLink and AboveNet, two of the largest networks in the data set. However, the third largest network, Tiscali shows the largest difference between highest flow and other centrality metrics. We believe this is in part due to Tiscali graphs' slightly dissassortative behavior where highest degree nodes form a core of the network. Hence, centrality based removals are not able to produce viable subgraphs.

Figure 4 presents the effect of the number of cores in the networks with %8 independent edge removal. As observed in both the Telstra and overall average results, the speedup performance improves with the number of cores. However, this improvement is sub logarithmical.

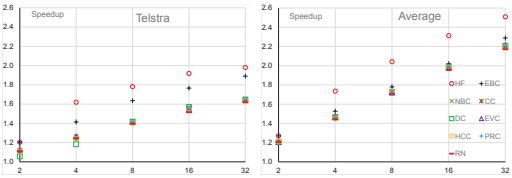

Fig. 4. Effect of the number of cores with 8% independent removal (log-scale)

C. Analysis of Subgraphs

In this subsection, we perform a detailed analysis of the subgraphs generated by each of the heuristic. Figure 5 presents the average of the average node degree, maximum node degree, and clustering coefficient of all graphs. We present the average of the independent removal method on all Rocketfuel topologies. Cumulative subgraph generation significantly changes graph characteristic and removes connectivity in the graph. In general, node centrality heuristics disrupt the network in subgraphs more than edge centrality and highest flow heuristics. Node centrality heuristics have almost the same effect in a manner of topology disruption. We observe that small removal percentages have low effects on subgraph characteristics. However, in larger removal percentages MCPR is not able to generate different subgraphs and repeats itself after a threshold. Note that the removal percentage should be less than $100/coreNumber$ as after this point, there are no more nodes to remove in the subgraphs.

In node centrality heuristics, first subgraph removes the most centralized nodes, which causes significant decrease in the average and maximum node degree. After a point, MCPR start to remove periphery nodes from the network and this increases the average node degree. This indicates that later subgraphs might not improve the performance as removing periphery nodes does not contribute to balance the load. Average node degree, however, is not effected much by the edge removal heuristics. As a result, edge removal heuristics performs better than node centralities (in Figure 3). We can observe the same pattern for maximum node degree.

As highest flow is dynamically generating new subgraphs, it tends to remove the bridge edges in the network. This causes an increase in the clustering coefficient of new subgraphs. As edge betweenness centrality removes the highest centrality edges in initial subgraphs, the bridges remain in the subsequent subgraphs. This leads to a decrease in clustering as central group of nodes are removed in subsequent subgraphs.

On the other hand, in cumulative approach, average and maximum node degrees reduce significantly. Clustering increases with small removal percentages but when removal percentage is increased clustering also plummets.

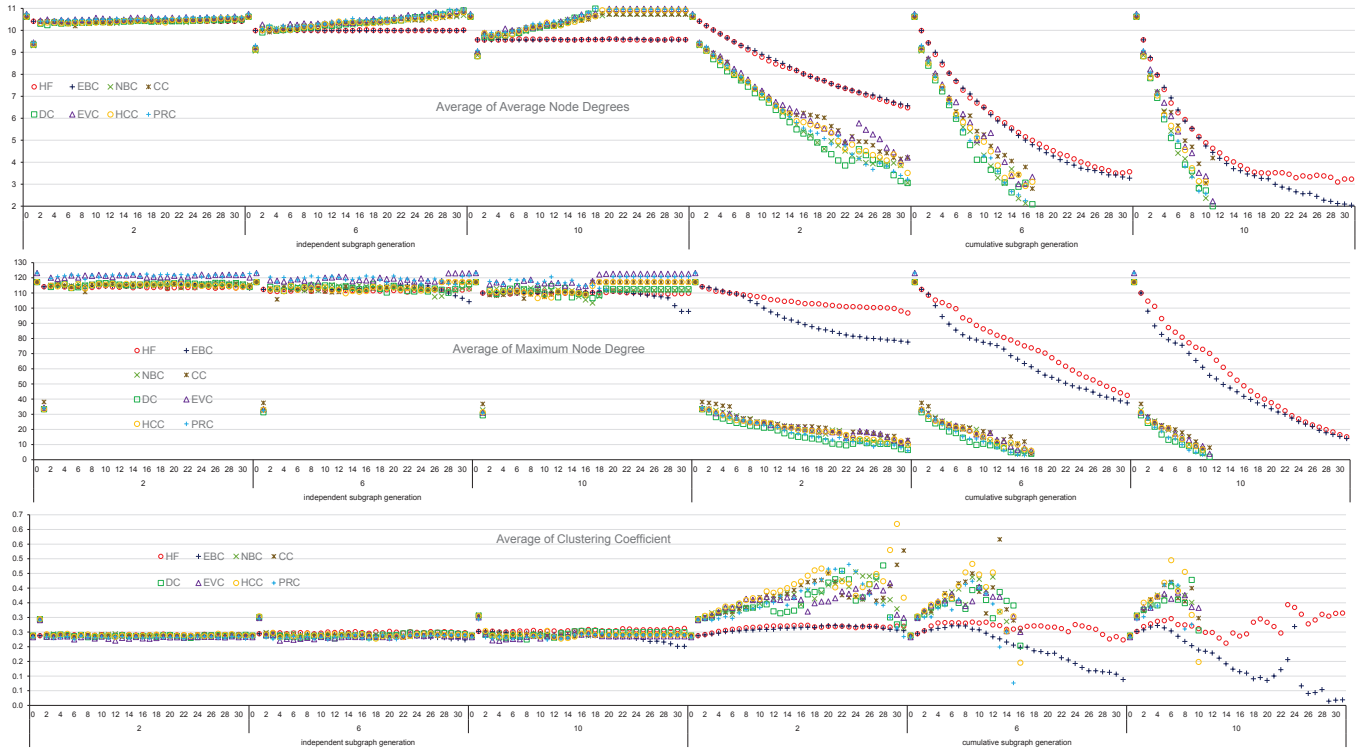

Fig. 5. Average of the graph characteristics of the subgraphs for 2%, 6%, and 10% removal

VI. Summary and Future Work

In this paper, we presented a new multi-path routing framework 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 performed a detailed graph analysis of subgraphs on multiple topologies to determine best centrality heuristics to utilize in Multi-Core Parallel Routing (MCPR). Experimental results show that centrality based heuristics are able to increase overall throughput in the network compared to the current shortest path approach.

Possible future work includes theoretical proof of the NP Completeness of finding the optimal subgraphs, extending heuristics for heterogenous core deployments and to create subgraphs with search algorithms such as genetic algorithms, and exploring performance under node/edge failure scenarios.

Acknowledgment

This material is based upon work supported in part by the National Science Foundation grant number CNS-1321069.

References

[1] S. Lohr, "The age of big data," *New York Times*, vol. 11, 2012.

[2] S. Molnár, B. Sonkoly, and T. A. Trinh, "A comprehensive tcp fairness analysis in high speed networks," *Computer Communications*, vol. 32, no. 13, pp. 1460–1484, 2009.

[3] S. Nelakuditi and Z. Zhang, "On selection of paths for multipath routing," in *IEEE/IFIP IWQoS*, 2001, pp. 170–186.

[4] H. T. Karaoglu, M. B. Burak, M. H. Gunes, and M. Yuksel, "Multi path considerations for anonymized routing: Challenges and opportunities," *IEEE NTMS*, pp. 1–5, 2012

[5] C.Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath tcp," in *ACM SIGCOMM*, 2011, pp. 266–277.

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

[7] Y. Wang, S. Su, A. X. Liu, and Z. Zhang, "Multiple bulk data transfers scheduling among datacenters," *Computer Networks*, 2014.

[8] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *ACM SIGCOMM*. New York, NY, USA: ACM, 2010, pp. 63–74.

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

[10] N. Hanford, V. Ahuja, M. Farrens, D. Ghosal, M. Balman, E. Pouyoul, and B. Tierney, "Improving network performance on multicore systems: Impact of core affinities on high throughput flows," *Future Generation Computer Systems*, vol. 56, pp. 277–283, 2016.

[11] Y. Wu, Z. Zhang, C. Wu, C. Guo, Z. Li, and F. C. Lau, "Orchestrating bulk data transfers across geo-distributed datacenters," 2015.

[12] I. Gojmerac, T. Ziegler, F. Ricciato, and P. Reichl, "Adaptive multipath routing for dynamic traffic engineering," in *IEEE GLOBECOM*, San Francisco, CA, November 2003.

[13] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *Lightwave Technology, Journal of*, vol. 31, no. 1, pp. 15–22, 2013.

[14] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "Spain: Cots data-center ethernet for multipathing over arbitrary topologies." in *NSDI*, 2010, pp. 265–280.

[15] H. T. Karaoglu, M. Yuksel, and M. H. Gunes, "On the scalability of path exploration using opportunistic path-vector routing," *IEEE ICC*, pp. 1–5, 2011

[16] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," in *ACM SIGCOMM*, 2002, pp. 133–145.