Received July 10, 2017, accepted August 15, 2017, date of publication August 24, 2017, date of current version September 19, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2743697

CentFlow: Centrality-Based Flow Balancing and Traffic Distribution for Higher Network Utilization

RAJESH CHALLA[1], SEIL JEON[2], DONGSOO S. KIM[3], AND HYUNSEUNG CHOO[1]
[1]College of Software, Sungkyunkwan University, Suwon 16419, South Korea
[2]College of Information and Communication Engineering, Sungkyunkwan University, Suwon 16419, South Korea
[3]Indiana University–Purdue University Indianapolis, Indianapolis, IN 46202 USA

Corresponding author: Hyunseung Choo (choo@skku.edu)

This work was supported by the Korean Government (MSIT and MOE) through IITP under Grant B0101-15-1366, through G-ITRC under Grant IITP-2017-2015-0-00742, and through PRCP under Grant NRF-2010-0020210, respectively.

ABSTRACT Next-generation networks (NGNs) are embracing two key principles of software defined networking (SDN) paradigm functional segregation of control and forwarding plane, and logical centralization of the control plane. A centralized control enhances the network management significantly by regulating the traffic distribution dynamically and effectively. An eagle-eye view of the entire topology opens up the opportunity for an SDN controller to refine the routing. Optimizing the network utilization in terms of throughput is majorly dependent on the routing decisions. Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS) are well-known traditional link state routing protocols proven with operation over operator networks for a long time. However, these classical protocols deployed distributively fall short of expectation in addressing the current routing issues due to the lack of a holistic view of the network topology and situation, whereas handling enormous traffic and user quality of experience (QoE) requirements are getting critical. IP routing in NGN is widely expected to be supported by SDN to enhance the network utilization in terms of throughput. We propose a novel routing algorithm–CentFlow, for an SDN domain to boost up the network utilization. The proposed weight functions in CentFlow achieve smart traffic distribution by detecting highly utilized nodes depending on the centrality measures and the temporal node degree that changes based on node utilization. Furthermore, the frequently selected edges are penalized thereby augmenting the flow balancing and dispersion. CentFlow reaps greater benefits on an SDN controller than the classical OSPF due to its comprehensive view of the network. Experimental results show that CentFlow enhances the utilization of up to 62% of nodes and 49% of links, respectively, compared to an existing Dijkstra algorithm-based routing scheme in SDN. Furthermore, nearly 6.5% more flows are processed networkwide.

INDEX TERMS Software defined networks (SDN), flow balancing, network utilization, traffic distribution, congestion control.

I. INTRODUCTION

Software Defined Networking (SDN) enables simplified and efficient network management by implementing two fundamental principles; functional isolation of control and forwarding plane and centralized control of a distributed network. SDN is nowadays widely considered and recognized as one of the core enablers in major 5G standardization groups [1]–[4], as it can effectively handle the network resource and behavior with intelligence. The SDN paradigm is expected to improve the network utilization while reducing the capital expenditure (CAPEX) and operational expenditure (OPEX) considerably [4]. Due to the compelling features such as a global view of network topology and network programmability, SDN paradigm is quite promising for monitoring and traffic engineering in data networks [12], [14], [48].

Traffic Engineering (TE) is one of the critical domains to optimize the network management and performance by dynamically analyzing, regulating, and adapting the routing of current flows/traffic [5]–[8]. Traditional TE mechanisms such as Multiprotocol Label Switching (MPLS) have been deployed in the enterprise/operator networks for high-performance data delivery from one network node to another based on the label switched paths (LSPs). However, it poses major challenges in terms of intricate control plane, and scalability and robustness in managing the LSPs. These challenges are becoming a bottleneck in harnessing and deploying the full potential of SDN [6], [9].

A relatively new TE mechanism called Segment Routing (SR), touted as one of the most promising routing mechanisms in SDN, addresses these challenges [42], [43]. SR classifies the forwarding path into sub-paths called segments that simplify the control routing paths yielding enhanced network management. SR uses the classical Dijkstra shortest path algorithm based routing protocols like Intermediate System to Intermediate System (IS-IS) and Open Shortest Path First (OSPF) to advertise the segment information and to compute the routing paths [10], [11], [15]. The efficiency of these routing protocols lies with the weight function employed in them. Besides that, IS-IS and OSPF are expected to be integrated and harmonized with SDN in the next generation networks (NGN) as the incremental deployment of SDN is a natural progression due to the economical reasons [12]–[14], [49]. Coexisting with these routing protocols and leveraging them to exploit the SDN benefits are of high importance and can lead to a substantial improvement in network utilization along with the optimized TE performance. We, therefore, see a need to customize these distributed routing protocols by bringing in certain measures. Centrality is one of the most appropriate and viable measures that can impact routing path decision, however its usage and deployment have been limited so far due to the distributed routing environment [32], [36], [37], [46].

Thanks to its global view and centralized control, an SDN controller can take better routing decisions that are optimal for a particular use case by deploying intelligent cost functions. However, routing decisions based on edge weight alone may undermine other important performance aspects such as traffic distribution, load on the individual network entity (link or node) and overall network utilization, which are currently becoming one of the crucial viewpoints in NGN. Secondly, it may be really hard to treat all network elements (nodes and edges) with the same priority and importance in the real world. Traffic steering without considering the importance of a network element may dilute the QoE considerably.

In this paper, we propose CentFlow, a new flow balancing approach to achieve better network utilization and traffic distribution in SDN environment. CentFlow emphasizes on intelligent traffic dispersion by employing two key factors betweenness centrality and temporal node degree (effective-degree, in short) for the routing path determination. The former one reflects the betweenness centrality of nodes and edges in a network topology [17]–[20], while the latter one represents a dynamic value associated to a node signifying the number of routable paths and is evaluated dynamically. These attributes constitute the novel node and edge weight functions which distinguish and penalize the over-utilized nodes and edges, thus reducing their chances of being chosen in the next shortest path computation. Our simulations show the impact of each factor, i.e. centrality and effective-degree towards load balancing, and evaluate the performance of CentFlow with a previously proposed Dijkstra algorithm based scheme [16] in terms of network utilization and traffic distribution. This article focuses on data plane traffic balancing. Control plane load distribution is beyond the scope of our current research. CentFlow helps in self-organizing the network traffic and defers the saturation point (congestion) without incurring additional monitoring overhead. More specifically, our contributions in this paper are:

- Proposing a customized link state routing algorithm for SDN environment. The intelligent flow-dispersion weight functions distribute the flow demands appropriately to relatively less central nodes/edges yielding higher network utilization in terms of the number of flows processed.
- Defining and dynamically evaluating the packet forwarding capacity of a node quantified by effective-degree, which contributes crucially towards deferring the saturation point of the network and assists in attaining better flow balancing in conjunction with centrality measures.
- Demonstrating the effectiveness of each metric individually in flow-dispersion. The edge-betweenness centrality has the least contribution among all. These results are topology dependent where we considered large scale graphs having average centrality values.
- Exhaustive evaluation of the proposed mechanism using different topology files from [21] and Python based simulator to confirm the trends and behavior of our chosen metrics and proposed mechanism on large scale graphs. We observed that edge-betweenness centrality has little impact on link-level traffic dispersion. The link utilization is comparatively less for very low ranked (rank with higher values) due to the effective-degree factor which disconnects heavily utilized nodes and all the edges connected to them.

We believe that our study can be useful in assisting the latest research directions with respect to efficient inter-domain SDN routing (considering autonomous systems as nodes) and probable placement of middleboxes or other virtual network functions (VNFs) based on betweenness centrality measures. The insights provided in this paper regarding the influence of deployed metrics can be used in designing a scalable SDN-driven cloud environment (assuming cloud resource instances as nodes) and efficient monitoring system that measures certain highly central nodes or edges to estimate about the network flows with better precision.

The remainder of this paper is structured as follows. Section II presents the basic concepts of SDN, centrality and related work on load balancing, traffic engineering and network utilization. The problem conceptualization and proposed scheme describing CentFlow algorithm are thoroughly discussed in Section III. In Section IV, we evaluate the performance of CentFlow with an extended Dijkstra algorithm in terms of better network utilization and traffic distribution.

Section V finally concludes the paper with pointers to our future research work.

II. BACKGROUND AND RELATED WORK

A. SOFTWARE DEFINED NETWORKING

Software Defined Networking (SDN) is an evolutionary paradigm to nurture TE and simplify network management in the future IP networks. Legacy IP networks route the traffic using proprietary, hardware dependent protocols. These routing decisions taken based on the limited view and awareness of network topology may lead to suboptimal network utilization. SDN presents a novel paradigm that network behaviors are controlled and managed by a centralized controller with a holistic view of the network, based on the control-plane and data-plane separation [22], [23], [48]. The benefit of such a paradigm is to facilitate network programmability with adequate abstraction, for effectively controlling and managing the network with flexible service (or application)-based policy.

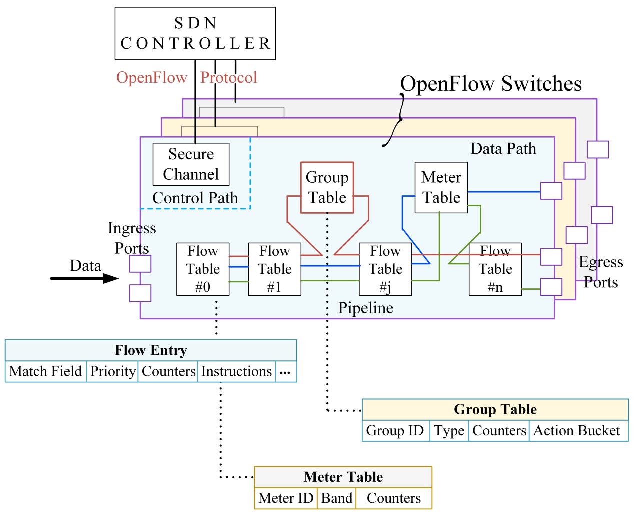

FIGURE 1. Internals of an OpenFlow switch supporting OpenFlow v1.3.

The internals of an OpenFlow (OF) switch supporting OF protocol v1.3 is depicted in Fig.1. OF is an open standard communication protocol for SDN, being available in many commercial products and research projects [23], [24]. In OpenFlow, the forwarding functions and operations reside in switch while control functions run on a centralized entity, a controller. The central SDN controller supervises a flow by installing a flow entry in the switches on route. A flow table of an SDN switch that holds the flow entries serves the same purpose as the forwarding information base (FIB) in legacy switches. The flow entries govern the routing path and are computed by the controller with its holistic view of the topology. Subsequently, each incoming packet at the SDN switch is matched against with the stored flow entries and is routed accordingly.

There are some ingress and egress ports in each OF switch through with the packet data comes in and goes out, respectively. As illustrated in Fig. 1, there exist two communication paths viz. a secure control path through which each switch performs control plane communication with an SDN controller using OF protocol; and a data path through which the data packet traverses. The data path consists of multiple flow tables called pipeline, a group table and a meter table per flow table. A flow table contains flow entries and each flow entry is composed of *match field*, *priority*, *counters*, and a set of *instructions* to apply to the filtered packets [54]. The *counters* field holds the number of packets which matched a criteria are transmitted or received. Once any flow entry matches the criteria, the corresponding instructions are executed that modify the pipeline processing or action set. For example, it may direct to pass the matched flow through a meter table that acts as a rate limiter i.e., sets an upper bound for the data rate what this flow can use utmost. This is how an OF switch realizes the QoS. A group table that contains group entries, provides an abstraction to manage a group of ports as a single entity. It applies a set of actions (*action bucket*) to a matched flow entry before sending it out through the egress port. These features, i.e. *priority*, *counters* in each table and QoS related meters are the some of the fundamental elements of the OF protocol that are required to realize traffic demand distribution based on utilization or bandwidth.

B. CENTRALITY

A data network can be represented in the form of a graph with switches and ports depicted as nodes and edges, respectively in a network. There are various measures in data networks to define the importance of network elements (NEs) i.e. nodes or edges. The most commonly used measure is centrality, which has been a predominant research topic in social networks [17], [19], [20], [25]. Out of various centrality measures described in literature [18]–[20], betweenness centrality and its variants are one of the widely used measure to identify the important nodes. Centrality indicators in a data network can ascertain the most important and highly utilized NEs. Employing centrality measures in the SDN routing policy is expected to improve the network utilization through minimizing the selection of highly-central elements steering to a well-distributed network traffic and defer the congestion saturation point.

Let $G = (V, E)$ is a directed graph, where V is a set of nodes and $E \subseteq V \times V$ is a set of directed edges. In a directed edge $\langle u, v\rangle \in E$, u and v are respectively called head and tail of the edge. A path between u and v is a sequence of alternating nodes and edges $u, \langle u, v_1\rangle, v_1, \langle v_1, v_2\rangle, v_2 \ldots \langle v_i, v\rangle, v$ to reach from u to v and the number of edges required to reach v is called the length of the path. The shortest path between any pair of distinct nodes is called geodesic and the length between them is known as the geodesic distance. A directed edge can be replaced by two opposite links to convert a graph into an equivalent undirected graph. In this paper, we assume the edges to be undirected unless stated explicitly as directed edges.

While there are many variants of betweenness centrality out there in graph theory, the focus of this paper is on node betweenness centrality and edge betweenness centrality [26]. The betweenness centrality of a node u is defined as the number of geodesics passing through the node u divided by the total number of geodesics between all pair of nodes [17], [18], [25]. Suppose $\sigma(u, v)$ is the number of geodesics from node u to v and $\sigma_n(u, x, v)$ are the number of geodesics passing through the node $x \in V$. Then, the betweenness centrality of the node x, denoted as $c_{nb}(x)$, is given by

$$c_{nb}(x) = \sum_{\substack{u\in V, v\in V,\\ u\neq x\neq v}} \frac{\sigma_n(u, x, v)}{\sigma(u, v)} \tag{1}$$

As the best case scenario, if x is on all the geodesics from u to v then its $c_{nb}(x)$ will be maximum $((|V| - 1)(|V| - 2))/2$. The normalized betweenness centrality $\bar{c}nb(x)$ can be obtained by dividing the betweenness centrality value with the total number of geodesics like

$$\bar{c}nb(x) = \frac{c_{nb}(x)}{\binom{|V|-1}{2}} \tag{2}$$

The betweenness centrality of edge $c_{eb}(x, y)$ is an extension of node-betweenness centrality. The edge betweenness centrality is defined as the ratio of the number of geodesics passing through the edge $\langle x, y\rangle$ to the total number of geodesics between nodes u and v [28]. The term link and edge are used interchangeably in this paper. $c_{eb}(x, y)$ and $\bar{c}eb(x, y)$ can be simply computed from Eq. (2) and Eq. (3), by replacing node x with edge $\langle x, y\rangle$, and is given by

$$c_{eb}(x, y) = \sum_{\substack{u\in V, v\in V,\\ u\neq\langle x,y\rangle\neq v}} \frac{\sigma_e(u, \langle x, y\rangle, v)}{\sigma(u, v)}, \tag{3}$$

where, $\sigma(u, v)$ is the number of geodesics between nodes u and v. The normalized edge betweenness centrality $\bar{c}eb(x, y)$ can be obtained by dividing the betweenness centrality value with the total number of geodesics like

$$\bar{c}eb(x, y) = \frac{c_{eb}(x, y)}{\binom{|V|-1}{2}}, \tag{4}$$

As shown in Fig. 2 , a network graph composed of 7 nodes ($|V| = 7$), where the distance between any node is equally given as 1. From Eq. (2), the node betweenness centrality $c_{nb}(g)$ of node g can be computed by,

$$\begin{aligned} c_{nb}(g) &= [\sigma_n(a, g, c)/\sigma(a, c) + \sigma_n(a, g, d)/\sigma(a, d) \\ &\quad + \sigma_n(b, g, d)/\sigma(b, d) + \sigma_n(a, g, e)/\sigma(a, e) \\ &\quad + \sigma_n(b, g, e)/\sigma(b, e) + \sigma_n(c, g, e)/\sigma(c, e) \\ &\quad + \sigma_n(b, g, f)/\sigma(b, f) + \sigma_n(c, g, f)/\sigma(c, f) \\ &\quad + \sigma_n(d, g, f)/\sigma(d, f)] \\ &= [1/2+1+1/2+1/2 + 1 + 1/2 + 1/2 + 1 + 1/2] \\ &= 6 \end{aligned}$$

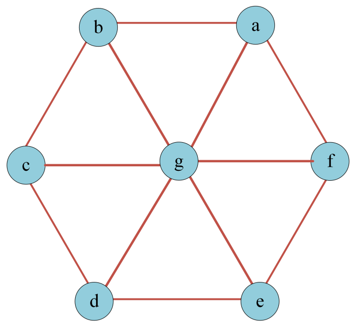

FIGURE 2. An undirected graph with unit distance between any two nodes. Clearly, node *g* has highest betweenness centrality.

$\bar{c}nb(g)$, the normalized value of $c_{nb}(g)$, is obtained by $6/15 = 0.4$. For all other nodes ($x = a, b, ..., f$), the $\bar{c}nb(x) = 1/15 = 0.07$ It infers that g is the most central node in this given graph.

The betweenness centrality helps in identifying the heavily utilized network elements for a shortest path algorithm and is an effective metric to determine the degree of load balancing in a network graph.

C. RELATED WORK

This section covers a contemporary literature review of traffic engineering for SDN and IP/SDN hybrid networks, especially from routing and network performance perspective. Firstly, we present recently published surveys on this topic followed by one of the hottest topics - QoS-aware routing algorithms that may distribute flows over sub-optimal shortest paths also, to meet service level agreements. Secondly, routing technologies and solutions in the incremental SDN deployment scenario are discussed where the Segment Routing (SR) is becoming a natural choice. Finally, we review how centrality measures are being harnessed in recent times for SDN networks.

With the advent of SDN to simplify the forwarding plane by moving the routing related intricacies and decision making to the centralized controller, experts are now focusing on better mechanisms to leverage the centralized and hybrid routing protocols for enriched TE in SDN [6], [47], [49], [50]. A recent literature introduces state-of-the-art results and trends of TE in SDN/OpenFlow networks [3], [55]. It focuses primarily on flow routing and management, load-balancing, and traffic analysis along with other TE areas. In [6], a reference framework for TE in SDN is proposed that includes traffic measurement and management, and enumerates recent studies on traffic management technologies and solutions in hybrid IP/SDN network. A new TE algorithm for information-centric networks allocates the traffic flows to those paths that can minimize the total response time of the flows [29]. Such an allocation of paths is based on the bandwidth estimation for each flow, performed by an SDN controller. An SDN based intelligent network measurement framework iSTAMP, aiming for higher network utilization, optimally aggregates and samples the most informative flows to accurately estimate the network flows [30]. Furthermore, iSTAMP selects most rewarding flows with high influence to improve the estimation accuracy. In yet another SDN based TE approach, a flow utilization perspective is considered as a measure of network utilization and evaluated the performance of this perspective with the real backbone datasets [34].

The impact of incremental SDN deployment for TE is explored in [14], which addresses the SDN controller optimization problem in the partial SDN deployment scenario to gain significant improvements in network utilization with reduced packet loss and delays. SDN partitioning [13] is a novel solution to tackle hybrid IP/SDN networks that deploys both distributed and centralized routing protocols such as OSPF/IS-IS and SDN in the same routing domain to facilitate smooth transition to SDN and for the optimized total cost of ownership (TCO). It proposes SDN switches to be the border nodes to remodel the distributed OSPF routing domains into sub-domains. SDN controller manages the inter-sub-domain routing. In [31], a novel heuristic algorithm HEATE for IP/SDN hybrid networks considers two entities: i) an SDN controller that decides the traffic splitting ratio to achieve multi-path routing and determines optimized link weights for shortest paths, and ii) IP routers that compute the shortest paths based on those optimized link weights. The flow demands are aggregated on partial links and the underutilized links are disconnected to conserve energy. All these research articles still exploit OSPF/IS-IS and hence Cisco proposed Segment Routing (SR) technology that clubs the best of MPLS and OSPF to create a better fit routing protocol for SDN or hybrid IP/SDN networks [44].

SR simplifies the control plane operation by classifying the entire routing domain into multiple sub-paths called segments [55], [56]. Routing within each segment is managed using OSPF/IS-IS protocols to route the data packets within each segment [35], [45], [57]. For 5G networks, a segment can be mapped to a network slice [58], where a slice is a logical network created to fulfill a telecom service provider business use case. Therefore, an efficient weight function is critical for routing within the next generation network slice. An effective heuristic algorithm for SR in SDN is proposed with an aim to increase the network throughput and construct a bandwidth-satisfying path [35]. To achieve its aim within each segment, the algorithm uses a weight function comprising of two attributes, namely, link criticality and residual bandwidth of the link. This weight function is applied to OSPF within each segment resulting in an optimized intra-segment network throughput. Link criticality that is used to predict the future traffic load by determining the link's importance also referred as *centrality* in graph theory and network analysis.

Of late, centrality has contributed notably to TE with SDN [27], [28], [32], [33], [37]. Centrality measures were more prominently studied in social networks than in data networks [17]–[20]. However, SDN has invigorated the data networks research from the viewpoint of centralized control and its benefits. A novel metric Quality of Alternative Paths (QAP) centrality is introduced to supplement the fast link failover feature of SDN protocol OpenFlow [32]. The QAP quantifies the node's neighbors that can be harnessed to infer more robust paths. The QAP centrality measure suggests paths bypassing a specific failed node. A novel topology-control algorithm uses the edge-betweenness centrality to ensure load balancing and higher QoS throughout the network [28]. An extended Dijkstra algorithm has been proposed for enhanced routing and reduced end-to-end latency in an SDN environment [16].

Unlike the classical Dijkstra algorithm concerning only edge weight, the extended Dijkstra assigns a non-negative weights to nodes as well as edges, thus computing the processing delay from both the node and edge weights. The shortest routing path in terms of delay is computed by summing-up both the weights. It maps the processing latency of a node and edge to their respective weights. In [38], it incorporates the extended Dijkstra algorithm to achieve load balancing of servers located in an SDN environment. Every OF switch acts as a load balancer and tries to figure out the nearest server from a client to reduce the end-to-end delay as well as to balance the load on servers. It focuses on server load balancing by installing the load balancer into every OF switch, to find out the nearest server from a requesting client, with priority. In addition to the intra-domain routing, centrality measures have lately been introduced in inter-domain routing too. An inter-domain routing centralization based on SDN is proposed and has shown significant performance improvements as an effect of centralization [47]. It shows a remarkable reduction in the average data-plane connectivity time if an SDN cluster contains Autonomous Systems (ASes) with high betweenness centrality. The implication of this research is that inter-domain centralization can improve the Internet performance considerably even with incremental deployment. New measures of centrality have been exploited for routing and optimization of link utilization in the SDN environment [46], [47].

Although inclusion of measures like centrality in link state algorithms may result in sub-optimal shortest path, however, for certain classes of routing algorithms like QoS-aware algorithms in which the traffic demands may be distributed to a little longer path [51]–[53]. An innovative QoS-aware routing algorithm determines an optimal path depending on the QoS requirement or type of flow (delay sensitive, bandwidth-sensitive and best-effort) [51]. In [52], a routing framework named SCOR is introduced for SDN that exposes simple abstract APIs to expedite SDN routing application development and leads an example demonstrating how this kind of framework opens up great opportunities for agile development, testing and innovation for new QoS-based routing. A new routing algorithm designed for SDN-based backbone networks aim to maximize utilization of network resources besides providing the required QoS [53]. Furthermore, it proposes flow-migration algorithms running on the SDN controller for dynamic routing in order to reduce the congestion and latency to improve the QoE.

Briefly, the articles mentioned in this section are intending to infer that the link state algorithms are going to stay in centralized control environment (such as SDN) too, however, a use case driven weight function customization is required to improve the network utilization and to defer the congestion (with low monitoring overhead) through efficient distribution of the traffic across nodes and edges. To address these requirements, we propose CentFlow algorithm, which achieves better traffic flow dispersion for QoS-aware use case and consequently, elevates the overall network utilization in the SDN.

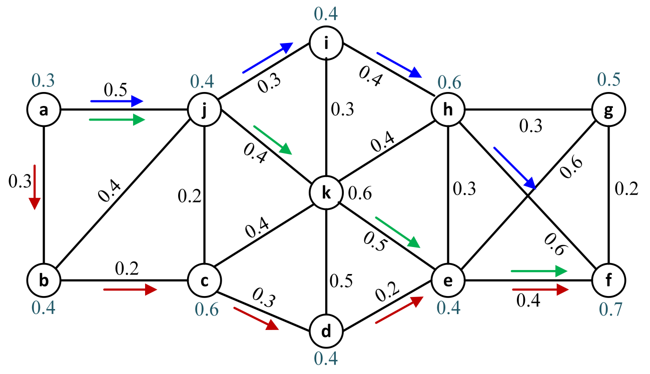

FIGURE 3. A weighted graph where node K has highest betweenness centrality and highest degree.

III. CENTFLOW: CENTRALITY-BASED FLOW BALANCING

A. PROBLEM CONCEPTUALIZATION

A network graph $G = (V, E)$ with non-negative node weight and edge weight is shown in Fig. 3. In a data network, switches are visualized as the nodes and the ports in each node are conceptualized as edges which connect any two nodes (switches). Each node has some processing capacity which, in the real world, is published in terms of Million packets per second ($Mpps$) or Megabits per second ($Mbps$), and the bandwidth of a port (edge) is calculated in terms of $Mbps$. A point worth noting here is that there are multiple shortest paths available in the graph, (as pointed out in the figure with red, blue and green paths) from the source (node A) to destination (node K) and there are more elephant flows in the network [39]. Furthermore, every node and edge can be assigned with a certain weight based on the monitoring feedback [40] or equivalent algorithms [29], [31]. Considering the graph in Fig. 3, the challenging scenarios mentioned below can crop up by using the algorithm proposed in [16], for the routing path computation process;

$S1$. As node k resides at the central position in the graph, it may be selected more frequently than other nodes such as nodes f and g during the shortest path computation process. It may lead to congestion and performance bottlenecks.

$S2$. In case, there exists two distinct shortest paths, one that includes node k and the other includes node c then the current weight function does not prefer c, though it offers a greater chance (being a less central node than k) for fair distribution of traffic [16].

$S3$. All nodes are treated with the equal importance, i.e. the heavily and loosely connected nodes are treated in the same manner. In the real world, a few nodes are more important (central), some nodes may have relative higher fan-out/fan-in. These attributes may result in better diffusion and circulation of network flux.

$S4$. If the network administrator wants to run monitoring function/software to improvise the routing path on-the-fly then the current routing scheme provides quite limited information regarding the most central/critical nodes (best candidates to host the monitoring function).

Succinctly, imposing the weights on nodes and edges may give more calibration points in computing shortest path between any pair of source and destination nodes, however, the algorithm does not take the load balancing and congestion avoidance scenarios into consideration. It also does not hold the intelligence to minimize the maximum-utilized node or edge. Therefore, it may result in a rather skewed network utilization that some nodes would be heavily-utilized whereas others would be under-utilized. Such situation leads to the unbalanced traffic routing and creates a performance bottleneck.

B. PROPOSED SCHEME

We propose CentFlow - an intelligent flow distribution algorithm that scatters the IP traffic and increases the network utilization by incorporating enhanced centrality measures. The main idea of CentFlow is to discourage the selection of nodes and edges which have high (node-/edge-) betweenness centrality values. Additionally, the selection criteria of a node is further calibrated by temporal node degree i.e. time-varying degree of the node. In short, we call this new attribute as effective-degree, δ_{eff}. Let $N(u)$ be the set of nodes adjacent to $u \in G$

$$N(u) = \{v \in V | \langle u, v\rangle \in E\} \tag{5}$$

Let θ_u be the number of edges $\langle u, v\rangle$ between u and its adjacent nodes whose utilization is greater than or equal to the link-utilization-threshold value τ_e, i.e.

$$\theta_u = \left|\{\langle u, v\rangle \in E \mid v \in N(u) \text{ and } util_e(u, v) \geq \tau_e\}\right| \tag{6}$$

The effective-degree of node u, $\delta_{eff}(u)$ in graph G is defined as its dynamic degree, depending on the utilization values. If the link utilization $util_e(u, v)$ of any edge is greater than or equal to the link-utilization-threshold value τ_e then the degree of corresponding head and tail nodes is reduced by one for each of such edges i.e. the effective-degree of these nodes is the value computed as total fan-in/fan-out of a node subtracted by the number of edges whose utilization touches or crosses the τ_e.

If the node-utilization $util_n(u)$ is greater than or equal to the node-utilization-threshold value τ_n then its effective-degree will become zero, regardless of the $util_e(u, v)$ of its attached edges. More precisely, effective-degree $\delta_{eff}(u)$ is defined as

$$\delta_{eff}(u) = \begin{cases} d(u), & \text{if } util_e(u, v) < \tau_e, \ \forall\, v \in N(u) \\ \\ d(u) - \theta_u, & \text{if } \exists\, util_e(u, v) \geq \tau_e, \ v \in N(u) \end{cases} \tag{7}$$

where $d(u)$ is the degree of node u; $util_e(u, v)$ represents link utilization between node u and v; and τ_e denotes the link-utilization-threshold value.

The role of effective-degree is to exclude the heavily-utilized node and/or edge in the actual path computation in a graph, by disconnecting them. The $util_e(u, v)$ and $util_n(v)$ are defined more specifically in Eq. (12) and Eq. (13). We explain the physical meaning of Eq. (7) in detail using Fig. 3. The degree (the number of edges attached to the node) of h and e is 5 respectively, and the numbers assigned on each edge represent its utilization. Assuming the utilization threshold τ_n as 50% (0.5), the $\delta_{eff}(h)$ becomes 0, because the node utilization of h is greater than τ_n (50%). If the utilization of a node reaches (or crosses) the threshold value, then it will be disconnected from rest of the network, therefore no more traffic can pass through the node. The effective-degree of all the neighbor nodes $(i, k, e, g, \text{and} f)$ decreases by 1. Similarly, $\delta_{eff}(e)$ is 3, as the utilization of the edge $\langle e, k\rangle$ and $\langle e, g\rangle$ is greater than or equal to τ_n.

CentFlow employs an additional impact factor to the node weight rather than the edge weight, due to following reasons; first, degree is the de facto attribute for determining the importance of a node, hence we wanted to illustrate its level of contribution in comparison to the betweenness centrality, however, the degree is a rather static attribute whereas traffic is dynamically changing input; second, the rerouting effort is usually higher in terms of recovery time and resource consumption for path computation of all the disrupted flows in case of the node failure compared to the link failure; lastly, a node usually caters to multiple edges which may lead to its frequent selection in geodesics and in turn affects its betweenness centrality. We define the node weight $wt_n(u)$ as

$$wt_n(u) = util_n(u) \times \bar{c}nb(u) \times \delta_{eff}(u) \tag{8}$$

where $\bar{c}_{nb}(u)$ is the normalized node betweenness centrality. The node utilization $util_n(u)$ is defined as

$$util_n(u) = \sum_{f \in flows(u)}^{n_f} \frac{pkts(f)}{cap(u)}, \tag{9}$$

where n_f is the total number of flows; f is the number of flows passing through node u; $cap(u)$ denotes the total processing capacity of a node and $pkts(f)$ denotes the total number of packets processed by the node.

The significance of Eq. (8) lies in the fact that each of its sub-component imposes a penalty on frequently chosen network entities. We assumed that our system has more elephant flows (having long lifespan) and hence the $util_n(u)$ values does not change rapidly. Initially, the traffic is distributed based on effective-degree i.e. higher the degree, relatively lesser the chances for its selection. However, as the network utilization increases, the $\delta_{eff}(u)$ reduces due to the edges hitting saturation point. At this time, the $\bar{c}nb(u)$ helps in realizing traffic distribution more granularly and the same has been confirmed in our simulation results.

Considering situation $S1$ in subsection III-A, the node-betweenness centrality of node k in Fig. 3 will keep on increasing because of its frequent reselection. To discourage the selection of central nodes or edges during the next computation cycle of shortest paths, the weights (node and edge) are manipulated depending on their respective betweenness centrality values. CentFlow picks up the smaller weights having smaller betweenness centrality co-efficient in subsequent computations.

The edge weight $wt_e(u, v)$ is defined by

$$wt_e(u, v) = util_e(u, v) \times \bar{c}eb(u, v), \tag{10}$$

where $util_e(u, v)$ is the link utilization and is defined as

$$util_e(u, v) = \sum_{f \in flows\langle u,v\rangle}^{n_f} \frac{bits(f)}{bw}, \tag{11}$$

where f is the number of flows passing through edge $\langle u, v\rangle$ and n_f is the total number of flows. The $bits(f)$ represents number of bits transmitted per unit time over the total max bandwidth, bw of the link.

Eventually, the final weight function is obtained by adding both the weights (node and edge), and is given by

$$wt(u) = wt_e(u, v) + wt_n(u) \tag{12}$$

Rephrasing Eq. (12) with the extended formulations given in Eq. (8) - Eq. (11),

$$wt(v) = \sum_{f \in flows\langle u,v\rangle}^{n_f} \frac{bits(f)}{bw} \times \bar{c}eb(u, v) + \sum_{f \in flows(u)}^{n_f} \frac{pkts(f)}{cap(u)} \times \bar{c}nb(u) \times \delta_{eff}(u) \tag{13}$$

The pseudo-code of traffic distribution algorithm - CentFlow is shown in Algorithm 1. Given a non-negative weighted graph $G\,(V,\ E)$ and source node src_v, the initialization steps involve initializing the distance of all nodes with respect to the source node (lines 2 to 4). The set of visited nodes ($Visited_v$) is initially an empty set and the set (Q_v) contains all nodes (lines 5 to 6). Line 8 selects a node with minimum distance, which is a function of node/edge utilization ($util_e(u, v)$, $util_n(v)$), node/edge centralities ($\bar{c}nb(v), \bar{c}eb(u, v)$) and the effective-degree ($\delta_{eff}(v)$) that are computed on lines 11 to 13. After appending the chosen node to $Visited_v$ set in line 9, a new shortest path is searched that has smaller distance than already currently existing and if found then the distance is updated with the newly found shortest value

Algorithm 1 CentFlow Based Flow Distribution

```
   Input: $G=(V, E), wt_e, wt_n$
   Output: $d[0..|V|]$
1  Procedure CentFlow $(G, wt_e, wt_n)$
       // Initialization
2      $d[s] \leftarrow 0$
3      foreach $v \in \{V\} - src_v$ do
4          $d[v] \leftarrow \infty$
       // Initially,'Q' has all unvisited
          nodes
5      $Visited_v \leftarrow \emptyset$
6      $Q_v \leftarrow |V|$
7      while $Q_v \neq \emptyset$ do
           // Pick a node and move to
              'Visited'
8          $u \leftarrow min_{dist}(Q_v, d[v])$
9          $Visited_v \leftarrow Visited_v \cup \{u\}$
10         foreach $v \in neighbor[u]$ do
               // Compute effective-degree and
               // betweenness centralities
11             $\bar{c}nb(v) \leftarrow node\_betw\_centrality()$
12             $\bar{c}eb(u, v) \leftarrow edge\_betw\_centrality()$
13             $\delta_{eff}(v) \leftarrow effective\_degree()$
               // If any better geodesic
                  exists?
14             if $(d[v] > d[u] +$
               $[util_n(v) \times \bar{c}nb(v) \times \delta_{eff}(v)] +$
               $[util_e(u, v) \times \bar{c}eb(u, v)])$ then
15                 $d[v] \leftarrow$
                   $d[u] + [util_n(v) \times c_{nb}(v) \times \delta_{eff}(v)] +$
                   $[util_e(u, v) \times c_{eb}(u, v)]$
16     return $d[v]$
17
18 Procedure effective_degree $(G, v)$
       // Update edge and utilization values
19     $update\_utilization(util_e(u, v), util_n(v))$
20     if $(util_e(u, v) > \tau_e)$ then
21         $append(array_{disc}, e)$
22         $remove\_edge(G, e)$
23     if $(util_n(u) > \tau_n)$ then
24         foreach edge $e \in neighbor[u]$ do
25             $append(array_{disc}, e)$
26             $remove\_edge(G, e)$
27
28 Procedure show_all_edges $(G, eArr)$
       // Restore all edges temporarily
          removed
       // from the graph
29     foreach edge $e \in eArr$ do
30         $connect\_edge(G, e)$
31         $populate\_edge\_attributes(G, e)$
```

(lines 10 to 15). The $\delta_{eff}(v)$ is periodically evaluated against the threshold value (edge and node). The utilization values of node/edge are computed and evaluated (line 19) based on the shortest path computation. In case, edge utilization hits the threshold for any particular edge, the degree of affected nodes is re-evaluated. If the node utilization touches the threshold then it is temporarily removed from the Q_v and the degree of its adjacent nodes is recomputed (reduced by one) (lines 20 to 26). Lines 29 to 31 restores the graph.

C. PRACTICALITY OF CENTFLOW IN IP/SDN BASED NETWORKS

CentFlow falls under the link state routing family in which each router broadcasts and maintains the network state. Though this resource intensive approach is working well with the current data networks having distributed control, SDN - a centralized paradigm is expected to drive the future data networks that would eliminate the necessity of maintaining redundant network states per router basis and centralize the network state management at the SDN controller. In such a situation, a node, i.e. switch acts just as a simple forwarding network element that should be controllable just the way links (switch ports) are controlled (through an OSPF weight function). This alludes towards the need of yet another turning parameter - a node weight component, in the existing link state algorithms.

Secondly, the centrality measures were less relevant so far in data networks due to its distributed nature, however, SDN revives the relevance of centrality measures, i.e. a value which can quantify the importance of a network element at global network-wide level. This critical information can be used to identify: i) where monitoring tools and hooks should be installed optimally so that maximum amount of traffic can be monitored? ii) where a set of virtualized network functions be placed so that a certain set of flows can pass through a sequence of network elements with pre-determined importance value? iii) which NEs need fast failover recovery as their downtime may affect the network critically? and iv) which network elements can reduce the traffic load by distributing the demands across other equal cost (with same importance-level and utilization) paths?

CentFlow highlights the impact of node weight component in the link state algorithm. The routing application that computes and decides the weight functions, can be an SDN control application that runs on the controller. Whenever a new flow arrives at a switch that does not have a relevant flow table entry to steer this flow, the switch forwards it to the controller. The routing (control) application runs the CentFlow algorithm and installs a flow entry in all switches that fall under the shortest path suggested by CentFlow. As mentioned in Section II, OpenFlow is a protocol to realize the SDN paradigm. OpenFlow has a *counters* field that provides the node or edge utilization values and the *meter table* and *meter band field* that act as bandwidth rate limiter for a switch and port respectively.

IV. PERFORMANCE EVALUATION

We evaluate the performance of our proposed CentFlow algorithm against the extended Dijkstra, in short, ExtD [16], and other variants which consider any one metric between effective-degree (ED) and betweenness centrality (BC). As shown in Eq. (13), the proposed weight function integrates node-betweenness and edge-betweenness centralities as well as the effective-degree attributes while the BC variant does not consider effective-degree in the node weight and the ED variant relies on the effective-degree only.

For the performance evaluation and comparison, we developed a Python based simulation module that has three subsystems. The first subsystem parses a large scale graph and simulates an input network topology [21]. Second subsystem manages the flow generation and parsing, i.e. a flow is generated and assigned with a random bandwidth to resemble the current telecom network scenario where the control plane works for establishing the data path reactively with the configuration of the required bandwidth and Quality of Service (QoS) for a flow. Finally, the core subsystem implements the CentFlow algorithm that computes shortest paths and checks how many flows it is able to process. These subsystems can be plugged-in over any Python based SDN controller such as Ryu, as northbound control applications. Ryu already has topology recognition and flow parsing services which can be utilized if Mininet provided topology is fed as an input.

TABLE 1. Main attributes of assessed medium scaled topologies.

Attributes	Adjnoun	Polbooks	Erdos-Renyi
Number of Nodes	112	105	200
Number of Links	425	441	450
Diameter	9	13	10
Avg. Shortest Path Length	3.8	4.0	4.9
Avg. Nodal Degree	7.6	8.4	4.5
Max. Degree	49	25	10
Min. Degree	1	2	1
Node Betweenness Centrality	0.25	0.14	0.06
Edge Betweenness Centrality	0.02	0.07	0.22

We have performed extensive simulations with varied link and node capacities on different topologies whose basic characteristics are listed in Table 1. The betweenness centralities (BC) values shown in Table 1 are the default values computed in Python using their respective capacities as the input parameter. Each flow demand is represented by a 5-tuple information: flow ID, source IP address, target IP address, bandwidth, and shortest path. Bandwidth allocation is made by assuming minimum QoS a flow requires. For simplicity in our simulation environment, we have made following assumptions; i) currently, the queuing delay is not considered; ii) the required bandwidth (QoS) is known in advance; iii) a network snapshot is considered, so all the flows are assumed to be in progress. The reason for assumption ii) is due to the fact that in current mobile(3G/LTE) networks, any packet switched application or service like web browsing or voice over LTE (VoLTE) initiates the control-signaling messages. These messages in turn establish the data plane connectivity with pre-determined QoS which has a maximum bit rate (MBR) attribute that acts as an upper bound for the data rate what this particular data plane connection can utmost achieve.

The subsequent graphs and results shown in this article are based on Adjnoun topology [21]. For each node in a topology, a random function picks a link capacity between 700 Mbps to 50 Gbps. All links of a node are assigned the same link capacity. The node capacity is twice (full-duplex) the aggregate capacity of all connected edges. For instance, if a node has 4 links with 20000 Mbps capacity each then the node capacity is 4×40000 Mbps (considering full-duplex). Different nodes with same degree can have different capacity as its links' capacity may be different. In other words, we can say that the capacity of a node is first halved and then divided equally to its ports. The input flow demands range between 10000-32000. The link and node utilization threshold values are set to 50% and 70%, respectively, which are more realistic values in the real world although higher threshold values can be found in the literature [39], [41].

OpenFlow (OF) is an SDN control plane protocol that can be used to set up the required environment to realize CentFlow pre-requisites. The bandwidth rate limit per node or per edge (port) of a node is configured with the help of OF version 1.3 optional feature - *Meter Band*. The number of packets processed by a node and the number of bits passing through an edge can be calculated by using the OF *counters* field. Number of shortest paths can easily be determined by an control application on the SDN controller as it knows, on per node basis how many flows were passed. Therefore, centrality computation is fairly uncomplicated. The relevant details of OF protocol are explained in Section - II.

In a nutshell, after reading a network topology, random weights are assigned to the nodes and edges. Both the centrality measures (node and edge betweenness) and effective-degree are computed and stored. The flows are allowed to pass through and reach the destination, unless any network element (node/edge) hits the threshold during shortest path computation. If a network element touches the threshold then it will be disconnected and isolated from the network. This operation affects the effective-degree of neighboring nodes and hence, is recomputed. The network becomes disconnected at the saturation point (congestion), where the shortest path computation results in zero shortest paths.

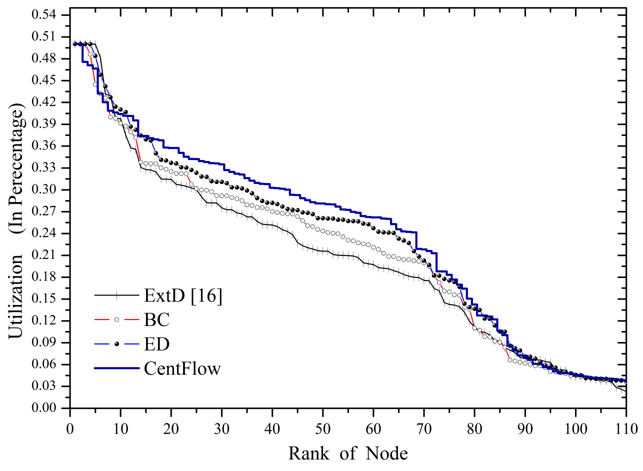

FIGURE 4. Node utilization comparison results.

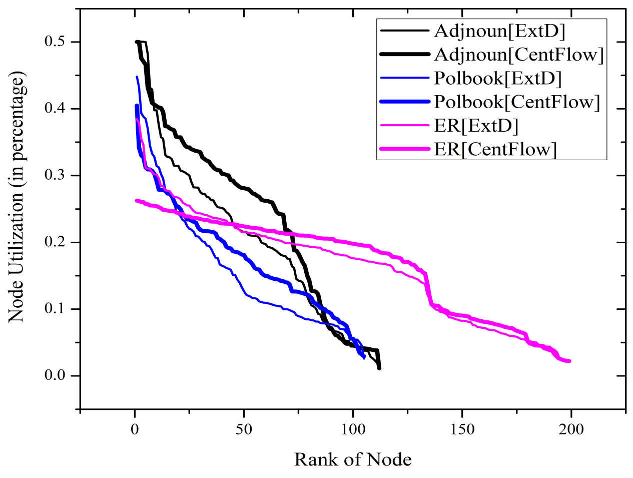

FIGURE 5. Trend of node utilization in three different topologies.

We have employed six critical metrics: i) node utilization, ii) link utilization, iii) number of flows addressed by each node/edge, iv) total number of flows being processed by the network, v) average hop count for each weight function and vi) average hop count difference between CentFlow and ExtD. Fig. 4 shows that nearly 62% of nodes have shown higher utilization. Furthermore, their individual utilization has also been increased on an average by nearly 34%. More specifically, the number of nodes with utilization ranging between 20 - 30% is doubled (nearly 32%) in case of CentFlow compared to ExtD and high-utilization values ranging between 30 - 50% (node-threshold value) is spread across 32% nodes with CentFlow metric against 19% of ExtD. Clearly, the number of nodes with full throttle is reduced to one-third with CentFlow metric. From the evaluation, we could establish the fact that the impact of effective-degree and betweenness centrality in traffic dispersion is higher around the mean rank nodes and less at the extremes. The metrics BC and ED distribute traffic evidently (up to 41% and 36%, respectively) between 20 - 30% range.

We intend to display how CentFlow performs against ExtD [16] that also harnessed node weight notion for better control on network elements selection across different types of topologies listed in Table 1. Out of the six metrics we took node utilization metric to establish the value addition of CentFlow metrics. Nevertheless, the other metrics too are found to follow the similar bias across these topologies. The trend of node utilization with CentFlow and ExtD metrics across topologies is illustrated in Fig. 5. The thick line of each color (Black, Blue and Magenta) represents the CentFlow metric in Adjnoun, Polbooks and ER topologies, respectively, and the thin line represents the ExtD metric. The fundamental benefits of CentFlow i.e. traffic distribution and NE utilization are seen to be elevated across the board. For a similar set of input flows and capacities of nodes and edges, the graph re-iterates the dispersion fact as well as a jump in the individual utilization of the NEs.

The comparative results of the link utilization by incorporating the above mentioned four cost functions, are

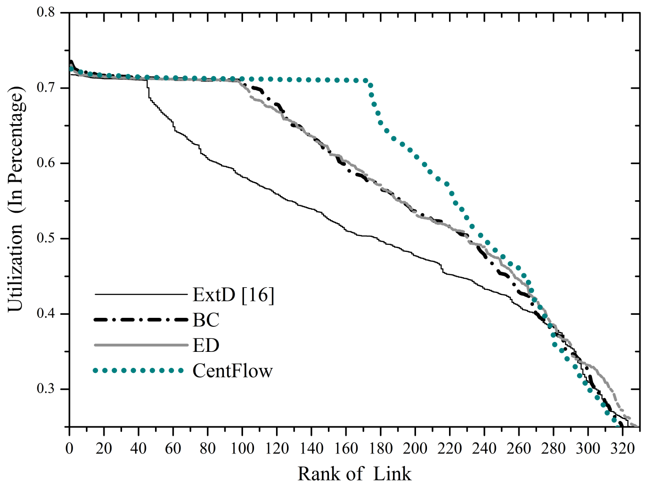

FIGURE 6. Comparative results between Rank of Link and its Utilization percentage.

illustrated in Fig. 6. The combined effect of BC and ED delivers appealing results. ED rebuffs the hub nodes and contributes to first level of distribution and the BC further fine grains the dispersion by relatively less central links. The impact of each metric seems quite promising, viz. the BC and ED display a significant jump in the link utilization across the network. The extent of traffic distribution is dependent on the network topology i.e., higher number of links and relatively smaller number of dense hub nodes are more beneficial. Correlating with real world, the hub nodes are potential risk for single point of failure. Therefore, appropriate diffusion of flows to less degree nodes make sense and also fruitful economically.

Statistically, nearly 12% of top ranked edges show higher utilization with BC and ED metrics and 28% with CentFlow against the ExtD metric in 50-70% (edge-threshold value) range. Furthermore, comparing between the contributions of individual betweenness centrality measures, we found that edge betweenness centrality contributes more than 50% of

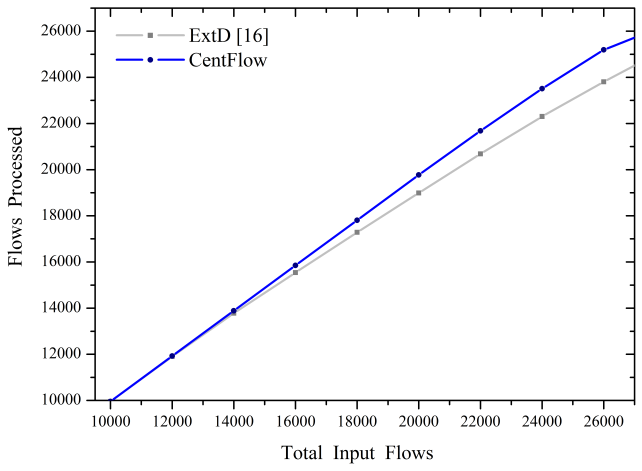

FIGURE 7. Comparison results of total number of flows processed.

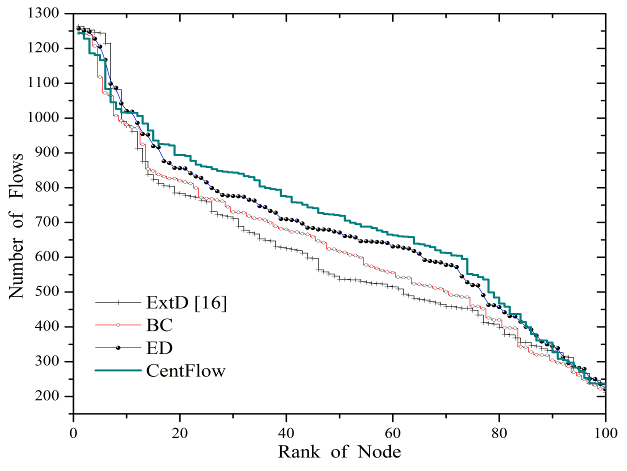

FIGURE 8. Comparison results of number of flows processed by a node.

traffic dispersion in this case. Overall, around 49% of the top ranked edges have shown higher utilization with CentFlow compared to ExtD. Specifically, 42% of top ranked edges touches the peak utilization compared to 28% and 14% for either of the metric (ED or BC) and ExtD, respectively.

The total number of flows processed in the network is depicted in Fig. 7. We execute the simulation with a wide range of flow demands ranging from 10000 to 32000, with an interval of 2000 demands. Both CentFlow and ExtD show the same network utilization (number of flows processed per unit time) but the network saturation point was extended by nearly 6.5% at 24000 input flows with the CentFlow metric. For the range of node and link capacities considered in our simulation and a given threshold setting, the number of flows processed increased between 5% - 6.5% at the cost of little higher average hop count. The impact of these metrics on hop count is discussed subsequently in this section. Concisely, CentFlow manifests better traffic distribution over the network, defers the saturation point, boosts up the individual utilization of top and middle ranked nodes and edges.

The number of flows processed by each node and edge is illustrated in Figs. 8 and 9 respectively. Maximum difference can be seen at mid-rank nodes in Fig. 8, where around 50% of nodes process additional 20% flows when compared with ExtD. Both ED and BC metrics show enhanced individual node utilization in around 30% of lower middle order ranked nodes. ED acts as a coarse traffic distribution factor and BC fine grains it due to the fact that BC values are spread across a huge number of shortest paths. Fig. 8 shows ExtD performs better than BC for top 8 - 10% edges, but BC overtakes the individual link utilization (with up to 11% enhancement) for rest all the edges. ED and CentFlow metrics show an improvement in link utilization of nearly 67% links in comparison to the ExtD weight function. The network-wide behavior shown in Figs. 4 and 6 are due to this individual performance at each node and link level that are captured in Figs. 8 and 9.

Higher utilization certainly comes at the cost of marginally higher hop count in CentFlow. Fig. 10 shows the average hop

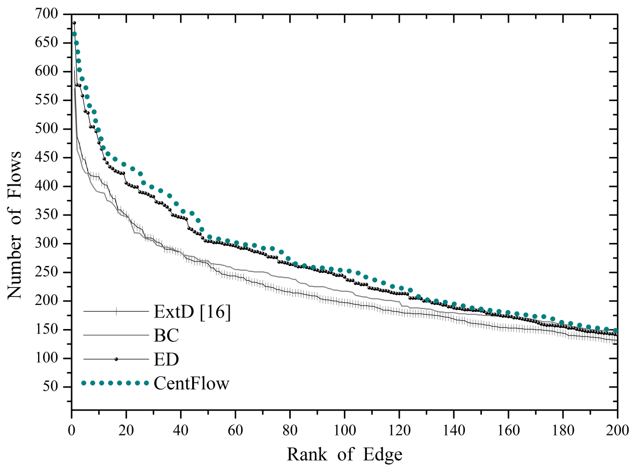

FIGURE 9. Comparison results of number of flows processed by an edge.

count values and the trend across all the four routing metrics viz. ExtD, BC, ED and CentFlow. The trend shows that the average hop count deviation remains more or less same after $28K$ flows in our topology for the aforementioned link and node capacities. The slight increase of average hop count (nearly 5%) is due to the selection of links and nodes which are relatively less central (small centrality values). Although the hop count is a basic (but not optimal) measurement of distance in a network, however, it does not always assure low packet latency. There are multiple random variables such as bandwidth, load, reliability, delay, and loss rate. Clearly, there is a trade-off between network utilization (due to increased traffic distribution) and average hop count, however, we opine that this marginal increase in the average hop count should not have any visible impact in delay tolerant network scenarios. Furthermore, for business scenarios and use cases where hop count may not be the only routing criteria for e.g., QoS-aware routing approaches; the CentFlow metrics are worth ruminating.

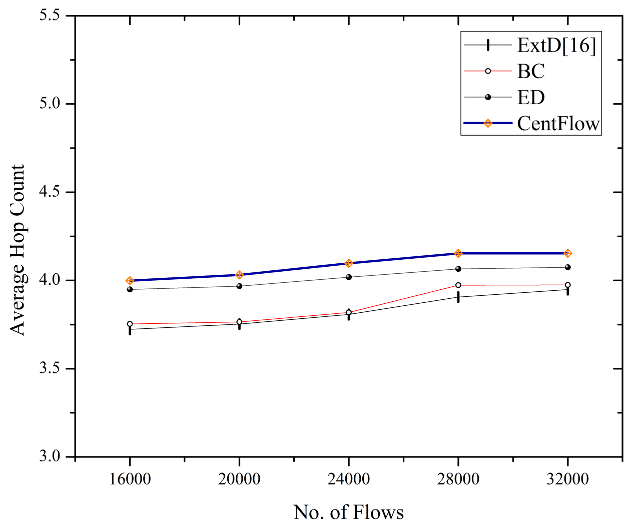

FIGURE 10. Average hop count comparison between the four different routing metrics - ExtD, BC, ED and CentFlow.

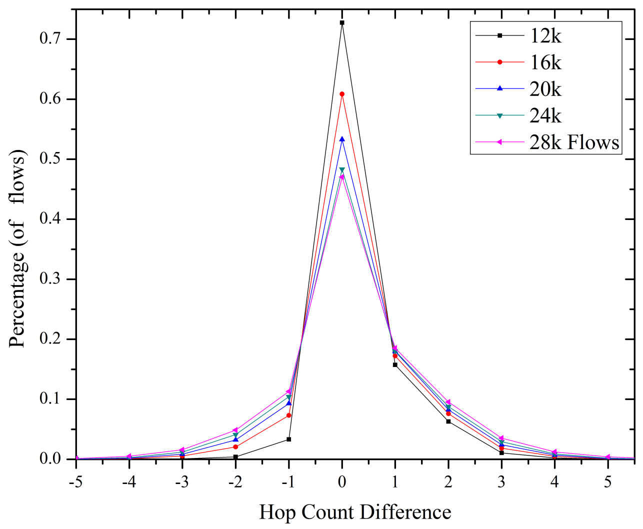

FIGURE 11. Average hop count difference between CentFlow and ExtD for different input flows.

A closer look at the number of flows affected by the hop count difference between the proposed metric (CentFlow) and ExtD [16] is shown in Fig. 11. The graph exhibits a normal distribution curve with around 70% flows (at 12k input flows) remain unaffected. As the traffic distribution increases with an increase in the number of input flows, the hop count deviation also increases, however, at a slower rate. The hop count deviation is seen on either side i.e. some flows have higher hop count, whereas some other flows have lower hop count. Usually, it is noticed that the reduced hop count flows are nearly half its counter side values, for e.g. around 10 - 20% flows have a hop count difference of 1 (increase) and nearly 5 - 10% flows have reduced (-1) hop count. A point worth noting here is that this normal distribution curve does not include additional flows that are processed by the network as a result of traffic distribution achieved through CentFlow.

V. CONCLUSION AND FUTURE WORK

This paper proposes CentFlow algorithm which comprises a new weight function to reduce the maximum-utilized nodes and edges for higher network utilization. The performance evaluation of the proposed cost function integrated with the Dijkstra algorithm has shown improvements in terms of node and link utilization over different large scale networks. Higher network utilization is attributed to the distribution of traffic flows to under-utilized nodes having low betweenness and degree centrality coefficients. The simulation results show that the utilization of around 62% nodes has increased and the link utilization has gone up for 49% of links in comparison to the extended Dijkstra. By integrating effective-degree of a node along with betweenness centrality of a node and edge, we could identify that the total number of flows processed network wide has grown up to nearly 6.5%. The limitation of computing betweenness centrality is time complexity. Therefore, reducing the complexity while increasing the performance is most importance task. In future, we will develop such enhanced algorithm and implement in ONOS open source SDN controller. Furthermore, we will investigate the behavior of centrality in MultiPath TCP (MPTCP) based scenarios and compare with Equal-cost multi-path (ECMP) routing.

ACKNOWLEDGEMENTS

The authors thank researchers at SKKU, Syed M. Raza, Pankaj Thorat, Duc-Tai Le, and Phan Tien Khoi for their contributions, suggestions and refinements.

REFERENCES

[1] *View on 5G Architecture*, 5G-PPP Archit. Working Group, Jul. 2016.
[2] 5G Initiative Team, "5G white paper," Next Generation Mobile Networks, Frankfurt, Germany, White Paper, Feb. 2015.
[3] "Report on standards gap analysis," ITU-T Focus Group, Geneva, Switzerland, Tech. Rep. FG IMT-2020, Mar. 2016.
[4] "5G mobile communications systems for 2020 and beyond," 5GMF, Tokyo, Japan, White Paper, May 2016.
[5] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Netw.*, vol. 30, no. 3, pp. 52–58, May 2016.
[6] Z. Shu *et al.*, "Traffic engineering in software-defined networking: Measurement and management," *IEEE Access*, vol. 4, pp. 3246–3256, Jun. 2016.
[7] F. Bernard, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 118–124, Oct. 2002.
[8] X. Dahai, M. Chiang, and J. Rexford, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1717–1730, Dec. 2011.
[9] B. Anat *et al.*, "Restoration by path concatenation: Fast recovery of MPLS paths," in *Proc. ACM Symp. Principles Distrib. Comput.*, Aug. 2001, pp. 43–52.
[10] J. Moy, *Open Shortest Path First Routing Protocol (Version 2)*, document RFC 2328, IETF, Apr. 1998.
[11] B. Fortz and M. Thorup, "OSPF/IS-IS weights in a changing world," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 756–767, May 2002.
[12] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in SDN/OSPF hybrid network," in *Proc. IEEE ICNP*, Oct. 2014, pp. 563–568.
[13] M. Caria, A. Jukan, and M. Hoffmann, "SDN partitioning: A centralized control plane for distributed routing protocols," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 381–393, Sep. 2016.

[14] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2211–2219.

[15] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[16] J.-R. Jiang, H.-W. Huang, J.-H. Liao, and S.-Y. Chen, "Extending Dijkstra's shortest path algorithm for software defined networking," in *Proc. IEEE APNOMS*, Sep. 2014, pp. 1–4.

[17] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," *Social Netw.*, vol. 30, no. 2, pp. 136–145, May 2008.

[18] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Netw.*, vol. 1, no. 3, pp. 215–239, Jan. 1979.

[19] M. G. Everett and S. P. Borgatti, "The centrality of groups and classes," *J. Math. Sociol.*, vol. 23, no. 3, pp. 181–201, Jan. 1999.

[20] S. P. Borgatti, "Identifying sets of key players in a social network," *Comput. Math. Org. Theory*, vol. 12, no. 1, pp. 21–34, Apr. 2006.

[21] *The UCI Network Data Repository*. Accessed on Sep. 2016. [Online]. Available: http://networkdata.ics.uci.edu/data/

[22] C. Hill and E. Voit, "Embracing SDN in next generation networks," Cisco, San Jose, CA, USA, White Paper, 2015.

[23] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[24] "Software-defined networking: The new norm for networks," Bigswitch, Santa Clara, CA, USA, White Paper, Apr. 2012.

[25] S. P. Borgatti, "Centrality and network flow," *Social Netw.*, vol. 27, no. 1, pp. 55–71, Jan. 2005.

[26] J. Guan, Z. Yan, S. Yao, C. Xu, and H. Zhang, "GBC-based caching function group selection algorithm for SINET," *J. Netw. Comput. Appl.*, vol. 85, pp. 56–63, May 2017.

[27] X. Li, X. Wang, K. Li, and V. C. M. Leung, "CaaS: Caching as a service for 5G networks," *IEEE Access*, vol. 5, pp. 5982–5993, Mar. 2017.

[28] A. Cuzzocrea, A. Papadimitriou, D. Katsaros, and Y. Manolopoulos, "Edge betweenness centrality: A novel algorithm for QoS-based topology control over wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 35, no. 4, pp. 1210–1217, Jul. 2012.

[29] S. N. Avci and C. Westphal, "A content-based traffic engineering policy for information-centric networks," in *Proc. IEEE CCNC*, Jan. 2016, pp. 711–719.

[30] M. Malboubi, L. Wang, C.-N. Chuah, and P. Sharma, "Intelligent SDN based traffic (de)aggregation and measurement paradigm (iSTAMP)," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 934–942.

[31] Y. Wei, X. Zhang, L. Xie, and S. Leng, "Energy-aware traffic engineering in hybrid SDN/IP backbone networks," *J. Commun. Netw.*, vol. 18, no. 4, pp. 559–566, Aug. 2016.

[32] T. Hegr and L. Bohac, "Impact of nodal centrality measures to robustness in software-defined networking," *Adv. Elect. Electron. Eng.*, vol. 12, no. 4, pp. 252–259, Oct. 2015.

[33] A. Tizghadam and A. Leon-Garcia, "Betweenness centrality and resistance distance in communication networks," *IEEE Netw.*, vol. 24, no. 6, pp. 10–16, Nov. 2010.

[34] A. Hassidim, D. Raz, M. Segalov, and A. Shaqed, "Network utilization: The flow view," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1429–1437.

[35] M.-C. Lee and J.-P. Sheu, "An efficient routing algorithm based on segment routing in software-defined networking," *Comput. Netw.*, vol. 103, pp. 44–55, Jul. 2016.

[36] D. Jiang, P. Zhang, Z. Lv, and H. Song, "Energy-efficient multi-constraint routing algorithm with load balancing for smart city applications," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1437–1447, Dec. 2016.

[37] V. Chellappan, K. M. Sivalingam, and K. Krithivasan, "A centrality entropy maximization problem in shortest path routing networks," *Comput. Netw.*, vol. 104, pp. 1–15, Jul. 2016.

[38] W. Yahya, A. Basuki, and J. R. Jiang, "The extended Dijkstra's-based load balancing for openflow network," *Int. J. Elect. Comput. Eng.*, vol. 5, no. 2, pp. 289–296, Apr. 2015.

[39] F. P. Tso and D. P. Pezaros, "Improving data center network utilization using near-optimal traffic engineering," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1139–1148, Jun. 2013.

[40] V. Heorhiadi, M. K. Reiter, and V. Sekar, "Simplifying software-defined network optimization using SOL," in *Proc. USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Mar. 2016, pp. 223–237.

[41] J.-J. Huang, Y.-Y. Chen, C. Chen, and Y. H. Chu, "Weighted routing in hierarchical multi-domain SDN controllers," in *Proc. IEEE APNOMS*, Sep. 2015, pp. 356–359.

[42] E. Moreno, A. Beghelli, and F. Cugini, "Traffic engineering in segment routing networks," *J. Comput. Netw.*, vol. 114, pp. 23–31, Feb. 2017.

[43] L. Luo, H. Yu, S. Luo, M. Zhang, and S. Yu, "Achieving fast and lightweight SDN updates with segment routing," in *Proc. IEEE GLOBECOM*, Feb. 2017, pp. 1–6.

[44] F. Hao, M. Kodialam, and T. V. Lakshman, "Optimizing restoration with segment routing," in *Proc. INFOCOM*, Apr. 2016, pp. 1–9.

[45] G. Trimponias, Y. Xiao, H. Xu, X. Wu, and Y. Geng. (Mar. 2017). "On traffic engineering with segment routing in SDN based WANs." [Online]. Available: https://arxiv.org/abs/1703.05907

[46] S. Yoon, T. Ha, S. Kim, and H. Lim, "Scalable traffic sampling using centrality measure on software-defined networks," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 43–49, Jul. 2017.

[47] P. Sermpezis and X. Dimitropoulos. (Feb. 2017). "Can SDN accelerate BGP convergence? A performance analysis of inter-domain routing centralization." [Online]. Available: https://arxiv.org/abs/1702.00188

[48] J. Rodriguez, E. Logota, D. Corujo, S. Jeon, and R. L. Aguiar, "The 5G Internet," in *Fundamentals of 5G Mobile Networks*. Hoboken, NJ, USA: Wiley, 2015, pp. 29–62.

[49] H. Xu, X. Y. Li, L. Huang, H. Deng, H. Huang, and H. Wang, "Incremental deployment and throughput maximization routing for a hybrid SDN," *IEEE/ACM Trans. Netw.*, to be published.

[50] D. K. Hong, Y. Ma, S. Banerjee, and Z. M. Mao, "Incremental deployment of SDN in hybrid enterprise and ISP networks," in *Proc. Symp. SDN Res. (SOSR)*, Mar. 2016, p. 1.

[51] L. Han, S. Sun, B. Joo, X. Jin, and S. Han, "QoS-aware routing mechanism in OpenFlow-enabled wireless multimedia sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 7, p. 9378120, Jul. 2016.

[52] S. Layeghy, F. Pakzad, and M. Portmann. (Jul. 2016). "SCOR: Software-defined constrained optimal routing platform for SDN." [Online]. Available: https://arxiv.org/abs/1607.03243

[53] S. Tomovic and I. Radusinovic, "Fast and efficient bandwidth-delay constrained routing algorithm for SDN networks," in *Proc. IEEE NetSoft Conf. Workshops (NetSoft)*, Jun. 2016, pp. 303–311.

[54] *OpenFlow Switch Specification V1.3.1*, Open Netw. Found., Palo Alto, CA, USA, Sep. 2012.

[55] A. Mendiola, J. Astorga, E. Jacob, and M. Higuero, "A survey on the contributions of software-defined networking to traffic engineering," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 918–953, 2nd Quart., 2017.

[56] C. Filsfils, S. Previdi, B. Decraene, S. Litkowski, and R. Shakir, *Segment Routing Architecture*, document draft-ietf-spring-segment-routing-11 (work in progress), Feb. 2017.

[57] S. Previdi *et al.*, *IS-IS Extensions for Segment Routing*, document draft-ietf-isis-segment-routing-extensions-12 (work in progress), IETF, Apr. 2017.

[58] C. Rotsos *et al.*, "Network service orchestration standardization: A technology survey," *Comput. Standards Interfaces*, vol. 54, pp. 203–215, Feb. 2017.

RAJESH CHALLA received the master's degree in computer applications from the National Institute of Technology, Warangal, India, in 2003. He is currently pursuing the Ph.D. degree with Sungkyunkwan University, South Korea, with a focus on software-defined networking, 5G slicing, and service chaining. Since 2003, he has been with Samsung, where his areas of expertise include IP multimedia subsystem IMS, Android radio interface layer, Linux device driver development, data plane and NAS layer of LTE, and proprietary Samsung Handset Platform solution. He lead the Samsung Android System Team for Japan market from 2012 to 2015 and significantly contributed toward the commercialization of Samsung Galaxy and Note series devices for NTT DoCoMo and KDDI. He is also a Principal Engineer with the Samsung Electronics R&D Institute India, focused on mobile commercialization activities and conducting research in next-generation mobile systems.

SEIL JEON received the Ph.D. degree in information and communication engineering from Soongsil University, South Korea, in 2011. He was a Lecturer with the Graduate School, Soongsil University, in 2011. He was previously with the Instituto de Telecomunicações, Aveiro, Portugal, as a Post-Doctoral Research Fellow and a Research Associate, participating and contributing in FP7 EU MEDIEVAL project and several industry projects with Portugal Telecom. He is currently a Research Professor (and a Research Fellow) with Sungkyunkwan University, leading the 5G team, while contributing and managing national/industry projects in South Korea. His research includes 5G networks, network slicing, software-defined networking, network functions virtualization, and mobile edge computing. He has actively been contributing in IETF, particularly in NETLMM, MULTIMOB, NETEXT, and DMM WGs, since 2008, producing two IETF RFCs with dozens of IETF Internet-Drafts. He has been serving as a Secretary of the Korea IETF mirror forum since 2016. He is currently a Standardization Expert supported by the Telecommunications Technology Association, South Korea.

DONGSOO S. KIM received the M.S. degree in computer science from The University of Texas at Dallas, TX, USA, in 1994, and the Ph.D. degree in computer science and engineering from the University of Minnesota, Minneapolis, MN, USA, in 1998. He was a Research Scientist with the Electronics and Telecommunications Research Institute from 1986 to 1992, and a Project Leader with Megaxess Inc., from 1998 to 2000. In 2000, he joined the Department of Electrical and Computer Engineering, Indiana University–Purdue University Indianapolis (IUPUI), USA. He was with the Department of Electrical and Computer Engineering, Sungkyunkwan University, from 2013 to 2015. He is currently an Associate Professor of ECE with IUPUI. His research includes switch networks, optical switches, network survivability, protection switching, network planning, QoS provisioning in the Internet, mobile ad-hoc networks, mobility of wireless networks, sensor networks, power-aware routing, software-defined networks, bio-inspired networks, and complex network analysis.

HYUNSEUNG CHOO received the B.S. degree in mathematics from Sungkyunkwan University, South Korea, in 1988, the M.S. degree in computer science from The University of Texas at Dallas, USA, in 1990, and the Ph.D. degree in computer science from The University of Texas at Arlington, USA, in 1996. From 1997 to 1998, he was a Patent Examiner with the Korean Industrial Property Office. Since 1998, he has been with the College of Information and Communication Engineering, Sungkyunkwan University, and an Associate Professor and the Director of the Convergence Research Institute. Since 2005, he has been the Director of the Intelligent HCI Convergence Research Center (an eight year research program) supported by the Ministry of Knowledge Economy (South Korea) under the Information Technology Research Center support program supervised by the Institute of Information Technology Assessment. He has authored over 200 papers in international journals and refereed conferences. His research interests include wired/wireless/optical embedded networking, mobile computing, SDN, IoT, and cloud and grid computing.