An Information-Centric Architecture for Data Center Networks*

Bong Jun Ko, Vasileios Pappas, Ramya Raghavendra, Yang Song,
Raheleh B. Dilmaghani, Kang-Won Lee, Dinesh Verma
IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
{bongjun_ko,vpappas,rraghav,yangsong,rbdilmag,kangwon,dverma}@us.ibm.com

ABSTRACT

We propose a new Data Center Network (DCN) architecture, based on the principles of Information-Centric Networking (ICN). Our Info-Centric Data Center Network (IC-DCN) addresses many of the pain-points in current DCNs, such as network scalability, host mobility, etc. At the same time, IC-DCN introduces a number of new features to the current information-centric network architectures. We achieve this goal by decoupling the control-plane and data-plane functionalities. The control-plane is implemented in a centralized manner, while the data-plane is fully distributed. We show that IC-DCN effectively addresses many of the ICN challenges in the data center, such as routing scalability, full network utilization, and name space management.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

Keywords

Information-centric networking, Data center network

1. INTRODUCTION

Information-Centric Networking (ICN) has recently attracted the attention of networking researchers as an alternative architecture to the traditional IP-based networking. While there are quite a few proposals for ICN architectures and protocols [1, 10, 7], one fundamental principle that encompasses all of them is that information is accessed based on the identifiers (or 'names') of the information, instead of the identifiers of the hosts (i.e., addresses) providing the information.

This new paradigm of information-centric networking addresses the challenges imposed by the current network applications in delivering information efficiently to the consumers. Among other things, ICN enables the delivery of data from multiple replicas, deals with host and data mobility in a more systematic way, and facilitates universal data caching. Yet, the ICN vision still faces a number of challenges en route to its full realization:

- Routing scalability: Since data in ICN are routed based on their names, the data-plane forwarding speed must be fast enough to handle lookups from a vast name space, which is several orders of magnitude greater than IP address space [12].
- Name management: ICN needs to ensure name uniqueness for each piece of information. Due to its huge size, the name space management in ICN becomes challenging compared to the address and name management of IP networks.
- Deployment: Deploying ICN technologies in a global scale requires overhauling the existing network infrastructure, which faces a practical challenge for adoption similar to other network technologies (e.g., IPv6, IP multicast, IP anycast).

In contrast, we argue that data center networks (DCNs) are better candidates for applying ICN design principles. First, current data center applications require efficient and agile management of their network traffic and server workloads, evident by the use of: *a)* server consolidation through virtualization, *b)* workload distribution through replication, *c)* traffic offloading through caching. Existing approaches, such as load balancers, caching proxies, directory services, etc., are deployed as point solutions. In contrast, ICN addresses current DCNs' needs in a more systematic way. In addition, the self-contained environment of DCN makes the adoption of new networking technologies more feasible, while its scale makes effective data management and networking operations possible.

In this paper, we propose a novel Information-Centric Data Center Network (IC-DCN) architecture. IC-DCN follows the main principle of ICN, i.e., data are accessed by their names rather than their host identifiers. At the same time, IC-DCN design takes advantage of the unique environment of data centers in addressing the pain-points of current DCNs and in resolving the scalability and the name space management issues of ICNs. At the heart of our architectural design is the separation of the control-plane functionalities (e.g., name management, routing decision) from the data-plane ones (i.e., data delivery). We show that IC-DCN enables, in a relatively straightforward manner, more sophisticated routing and data management schemes compared to previous ICN architectures.

2. IC-DCN OVERVIEW

Next we describe the IC-DCN architecture. As shown in Figure 1, the main difference of IC-DCN from other ICN architectures is the separation of the control plane functionalities from the data plane ones.

*The authors are supported by the U.S. National Institute of Standards and Technology under Agreement Number 60NANB10D003.

ICN'12, August 17, 2012, Helsinki, Finland.

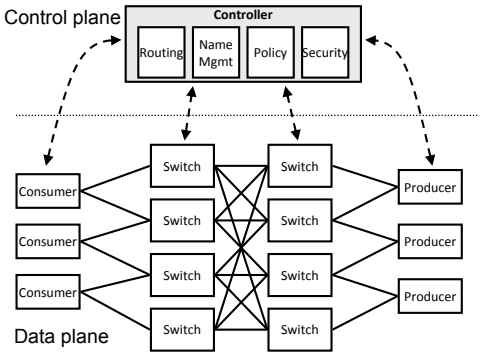

Figure 1: IC-DCN architecture

2.1 Network Entities

Like most other information-centric network architectures, IC-DCN introduces three types of network entities[1] that consume, provide and relay data in the network:

- **Consumers** request data by the network. A consumer can be a data center internal node, or a proxy for external nodes that access data in DCN from other networks (e.g., the Internet). In the latter case, the proxy-consumer performs the protocol and name translations between IC-DCN and the external network.
- **Producers** provide the data to the consumers. A producer can be a data center internal node, or a proxy for internal nodes that access data from other networks (e.g., the Internet). In the latter case, the proxy-producer performs the protocol and name translations between the external network and IC-DCN.
- **Switches** are the nodes that relay the request and response packets between the consumers and the producers. Additionally, switches can cache the data contained within the response packets, and they can respond to requests from the consumers on behalf of the producers.[2]

A data request from a consumer comes in the form of an *interest* packet, and a *data* packet is generated by a producer in response to an interest packet (or by a switch in case the requested data is found in its cache).

We assume that the names of the data are organized in a hierarchical manner, but this assumption is made only to accommodate already-established name space designs [7]. In fact, in our architecture, the issue of managing data names is decoupled from the issue of delivering data, rendering the name space design problem orthogonal to the IC-DCN architecture.

In addition to the above three basic types of entities in the data plane, IC-DCN introduces a new type of node, the **Controller**, that provides a set of services for determining the locations and the routes for the data-plane nodes accessing named data within the network. Specifically, the controller provides the following services:

- *Routing service* is responsible for collecting up-to-date information of the network topology and the locations of the data, and deciding the routing paths of the packets through the network.
- *Naming service* is responsible for assigning and managing name spaces to the producers of the named data, and for assuring the consistency and the integrity of the names published by individual producers.
- *Policy service* is a standard implementation of a repository of the policies for managing the system (host, data, network, etc.), defined in a common policy management architecture [16].

The controller is a logical entity, and its services can be implemented in a distributed manner by a set of servers. We discuss more about this issue in Section 5. In our architecture, the communication between the control-plane and the data-plane entities utilizes an IP-based network.

2.2 Control/Data Plane Decoupling

An important design decision for scaling the speed of name-based packet forwarding (to cope with a name space much larger than the conventional IP addresses) is to *decouple packet forwarding issue from data consumption*. The data are produced (announced) and consumed (requested) based on the names, whereas the packet forwarding is based only on the location of the data and the topology of the network. Specifically, the forwarding decision in data plane is made by the switches based on a fixed-length *label* that denotes one of the pre-configured paths in the network, whereas the routing decision in the control plane (i.e., assigning a path to a packet) is made by the controller when the packet enters the network. In Section 3, we will present more details on how we address the scalability issue by using a label-based forwarding mechanism.

The benefit of the centralized routing decision is that it enables sophisticated routing mechanisms to achieve better network resource utilization than decentralized routing solutions. In Section 4, we present two routing strategies that IC-DCN provides, namely, multi-path routing and cache-aware routing.

3. DATA PLANE

IC-DCN's scalable name-based packet forwarding solution takes advantage of the fact that DCN topologies change infrequently. Once an interest packet enters the network, the name of the packet is mapped to a predetermined path between the source and the destination node and the packet is *source-routed*. IC-DCN uses fixed-length labels to uniquely identify the different network paths and forwards packets based on their path labels, while it uses data names only to identify locally cached content.

3.1 Label-Based Forwarding

The controller is responsible for mapping between data-names and path labels. It maintains up-to-date information about the location of the data as well as the physical topology of the data center. For each pair of data-center nodes the controller identifies one or more network paths and assigns globally unique labels to each of them[3].

In addition, it populates switches forwarding tables with the appropriate entries, i.e., for each switch it finds all the paths that go through that switch and updates its forwarding table. The forwarding tables maintain the mappings between labels and outgoing ports, and they change only when the physical topology of the data center changes. Temporary link failures are not reflected in the forwarding tables, but instead the controller refrains from using any

[1]These are the logical entities, meaning a single physical host can assume multiple roles, although we expect the switches are deployed as separate physical entities in typical scenarios.

[2]We use the term *switch*, instead of *router*, to denote the entities that relay the named data requests and responses since in our architecture, they perform forwarding functions for the data without participating in the routing decision.

[3]A 64-bit label is large enough to assign unique labels to all paths between all pairs of hosts in DCN consisting of hundreds of thousands or even millions of end hosts.

Speed (Gbps)	10	100	1000
DRAM (MTPS)	0.31	3.05	30.52
SSD (MIOPS)	0.15	1.53	15.26

Table 1: Hardware specification requirements with 8192 bytes MTU at three network speeds.

paths that go through failed links, taking advantage of the existence of multiple paths. Furthermore, backup paths are used by the switches to forward interest packets whose assigned paths are not available.

Label-based forwarding is done as follows. Once an interest packet enters the network, if the first node[4] that receives the packet does not have the named data locally cached, it requests, from the controller, a path label that maps to the named data. Mappings between names and paths are cached for a short period of time in order to avoid excessive mapping requests for popular contents. The node then attaches the label in the interest packet and forwards it into the next hop, by looking up its forwarding table with the provided label. Each hop along the path to the destination checks if the named data is locally cached. A switch can reply back with the response packet if the data is present in its cache; if it is not cached, it forwards the interest packet into the next hop by looking up the path label in its forwarding table.

The data packets are forwarded along the reverse paths of the interest packets, that is, the node that replies with the actual data changes the label to the one of the reverse path.

3.2 Forwarding Scalability

One of the challenges in IC-DCN is providing high speed named data forwarding. Next, we investigate the feasibility of the label-based packet forwarding in terms of its scalability, assuming currently available hardware.

There are three main operations that limit IC-DCN's forwarding scalability: a) the lookup of labels in the forwarding table, b) the lookup of names in the local cache, and c) the retrieval of data from the cache. The first two operations are performed on every packet and both of them require exact matches. They can be efficiently implemented in DRAM or SRAM using hash tables and with $O(1)$ lookup complexity. Table 1 shows the millions of transfers per second (MTPS) required for both operations at three different network speeds. We assume a maximum transfer unit (MTU) of 8192 byte, which is within the limit of jumbo Ethernet frames. The third operation, data retrieval, is performed only for cache hits. Table 1 also shows the millions of I/O per second (MIOPS) required in the most demanding scenario of the 100% cache hit rate.

The first two operations can scale even up to the speed of 1Tbps aggregate throughput, if the cache lookup tables are stored in DDR4 DRAM[5]. The third operation, the data retrieval from the cache, can scale up to 1Tbps assuming all cached data is stored in DDR4 DRAM. If SSDs are used to increase the cache size, then only speeds of 10Gbps can be supported with existing SSD hardware. Scaling to 100Gbps of aggregate throughput requires multiple SSDs on the same node. For example, in a switch with five linecards, one can assign two SSDs per linecard and achieve 100Gbps aggregate throughput. Note that all previous calculations assume a cache hit rate of 100%. Using more realistic cache hit rates, higher aggregated throughput can be achieved when using SDDs. Furthermore, increasing the MTU size further improves the forwarding and caching scalability, as they relate linearly with the packet size. In summary, our initial calculations show that current data center network speeds can be supported in IC-DCN using existing hardware.

[4]It can be either an endhost or a switch.

[5]DDR4 is expected to support from 2133 to 4266 MTPS.

4. CONTROL PLANE

In this section, we present IC-DCN routing strategies that can achieve more sophisticated routing objectives than existing decentralized ICN architectures.

4.1 Multi-Path Routing

In data centers, multiple data replicas, e.g., popular files, can be hosted by different nodes with the goal of improving the redundancy and improving the utilization of the network resources. The information-centric networking can in principle provide naturally the networking operation needed to retrieve the data from multiple replicas.

However, in existing proposals for ICN architectures such as [7, 10, 1] where routing is based on *longest-prefix matching*, fully utilizing all network resources and replicas is rather difficult to achieve. This is because all data requests for one file converge to the same path, even when there are multiple information publishers that can provide the requested data simultaneously. For example, if a producer announces "/movies/diehard" and another one announces "/movies," all data requests for that movie will be routed to the first node although the second publisher contains the same desired file, which leads to underutilization of network resources.

In contrast, our centralized routing mechanism can provide a more flexible and efficient means to exploit the abundance of network resources. For example, suppose three producers have the same copy of a movie file with name "/movies/diehard", with two announcing "/movies/diehard" and the third announcing "/movies'. When a data request arrives at the network for a file block (which can be delivered in one data packet, e.g., "/movies/diehard/1" $\cdots$ "/movies/diehard/1000"), the controller computes the forwarding path label for each of such requests by using one of the three producers as the destination – regardless of the length of the announced prefixes – in order to achieve multi-path routing and load balancing. As a consequence, the overall movie file can be transferred to the consumer from all three servers along different paths, improving the response time and network utilization.

Furthermore, the number of requests designated for each producer can be adjusted on the fly according to time-varying network conditions, e.g., network congestion level and end-to-end delays. If a producer or a particular network path experiences significantly heavier congestion than others, the controller can shift the requests (i.e., assign different labels) for this path to others that can provide the same copy of data. The proportion of the requests that are designated for each producer, namely, the *proportion weight*, can be adjusted dynamically in tune with the network conditions and traffic variations, in order to optimize certain performance metric (e.g., estimated delay).

4.2 Cache-Aware Routing

Given that switches locally cache named data, it is advantageous to send interest packets along paths that can hit the cached data en route to the producer as early as possible. In existing ICN architectures, however, en-route cache hit takes place only opportunistically as routes are determined in a decentralized manner. Next, we present strategies that can improve cache-hit ratios through coordinated route assignment.

In IC-DCN, the controller can influence the cache hit ratio of intermediate switches by assigning, to all interest packets for a given name, paths that converge to the same set of switches. As a first order strategy, given the multiple paths that exist between a consumer

and the producer(s) of some data, the controller can assign a single path consistently, so that all interest packets for the same data generated by the same consumer are routed along the same path. To balance the network load across multiple links, the controller can apply consistent hashing based on names so that interest packets for different named data follow different paths.

To improve the cache hit ratio for interest packets generated by different consumers, a hierarchical routing strategy is employed. The idea is to configure the routing paths from different consumers such that they merge as early as possible, en route to a producer of the requested data. To do this, the controller first pre-computes the shortest path trees (SPTs) rooted at each producer with all consumers at the leaves. When a request for a data object arrives at a consumer, the controller maps the request to a path in an SPT corresponding to a producer of the requested data. Here, the key to increasing the likelihood of cache hits is to have the requests from different consumers use the same SPT consistently.

Furthermore, the load-balancing across multiple paths is accomplished in two steps: (i) The controller computes multiple SPTs (k-SPTs) rooted at each producer, and (ii) when a routing request for a data object arrives, the controller uses hash-based assignment to map the request to a particular SPT rooted at a particular producer of the requested data. Note that calculating k-SPTs per producer is a one-time operation, happening once for each topology change, and the route assignment is reduced to a matter of hash-based assignment of the requested data names to one of the SPTs.

An alternative strategy – to further improve the cache hit yet at the expense of complexity – is to use a history of the routing requests and the corresponding label assignment. Here, the controller maintains a recent history of the origin of recent requests and the label corresponding to the route decision in its cache. When a route request is made, instead of assigning a path based on consistent hashing, the controller makes use of this access history to return a path in a previously selected SPT, which would have a cached copy of the requested object with a high probability[6]. In this approach, however, a tradeoff between the cache-hit and load-balancing effects must be considered, since the access history of individual data blocks must be maintained to maximize the load-balancing effect.

Note that none of the above strategies requires the controller to know the locations of the cached data. Instead, the routing decision is made either based on the consistent mapping of the names to path labels or on the history of the path label assignments.

5. DISTRIBUTED CONTROLLER

Next, we describe a distributed implementation of the IC-DCN control plane, that scales with the size of the DCN both in terms of the routing requests rate and the route re-computation frequency, while providing fault tolerance against control node failures.

Recall that the routing decision in IC-DCN is decomposed into two parts: (i) the route computation and path labelling given the network topology and locations of data, and (ii) assignment of routes (i.e., path label) to the routing requests for data objects. In this section, we present how a routing service can be built to address the scalability issue in these two tasks of the routing service.

5.1 Route Computation Scalability

A route computation is carried out whenever there are network topology changes. The computation is performed either by one controller node or multiple of them in parallel, for fault-tolerance purposes. Each route computation controller receives the topology updates and then distributes the computed routes to the rest of the controller nodes. All controller nodes store the computed routes and use them for the mapping of interest packets to the appropriate path labels, based on the routing strategies of Section 4. Note that, in the rare event of physical topology changes, e.g. new equipments are added in the data center, the route computation generates new path labels, which are pushed to all controller nodes and affected switches.

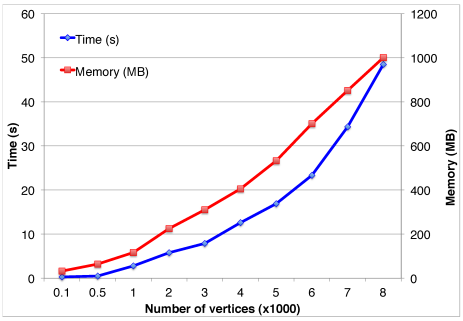

Figure 2: Scalability of routing algorithm, showing the time and memory needed to compute all pairs shortest paths when there are updates to network topology.

We evaluate the scalability of a centralized routing scheme in terms of the time it takes to recompute the routes and the memory needed to compute and store all pairs of shortest paths (APSP)[7]. We implement a dynamic version of APSP algorithm proposed by Demetrescu and Italiano [3] to recompute route updates. We consider the APSP algorithm for graph sizes ranging from one to ten thousand vertices[8]. The graphs are randomly generated, and comprise of 10 times as many edges as vertices. The graph edges are updated (inserts, deletes and edge weight updates) in a random manner and the shortest paths are recomputed. The simulation results are shown in Figure 2, indicating that graph sizes up to 5000 vertices can be recomputed in under one second.

5.2 Path Assignment Scalability

A more critical and demanding task of the routing service is to handle the large volume of the routing requests generated by the consumers, i.e., assigning the path label to each interest packet. While the first line of alleviating such a load of routing requests to the centralized routing service is to instrument routing caches at the consumers, a more fundamental solution we employ is to parallelize the path assignment task onto multiple servers, each of which handles a subset of the routing requests. More specifically, given K servers, called Route Assignment Elements (RAEs), that handle the routing requests, such a parallelism is achieved by the followings:

- Each consumer and producer is given a consistent hash func-

[6]The routing requests fulfilled by the local cache at each consumer, hence not observed by the controller, do not harm the effectiveness of this method since they are automatically routed along the same path that was previously enforced by the controller.

[7]As per the routing scheme used, APSP could be replaced with k-shortest paths, shortest-widest paths and so on. Experimental studies have shown similar practical running times for such algorithms [8].

[8]We consider only the switch nodes being vertices while the both consumer and producer nodes are not included in the route computation as they are stub nodes.

tion and a mapping of each hash value to one of K RAEs.

- When a producer publishes a data object, it calculates the hash of the data object's name, and publishes the name to the corresponding RAE.
- Similarly, when a routing request for a data object arrives at a consumer, the consumer queries the RAE mapped from the hash of the data name.

Note that such a parallelism by name space division with $O(1)$ operation can be naturally achieved without losing the benefits of the centralized routing service presented in Section 4. This is because the cache-awareness and the load-balancing of multiple paths can be performed separately for each name, while the global knowledge required in these operations is related only to what can be collected in much coarser time granularity (e.g., the congestion levels in network links). All in all, combined with the fact that each atomic operation of path assignment to named data request is a light-weight operation (e.g., $O(1)$ operations in hash-based, cache-aware, multi-path route assignment), such a parallel path assignment yields a highly scalable (and also flexible) routing service in IC-DCN.

A fault-tolerant feature can be built by configuring each RAE being backed up by other RAEs. Producers and consumers use multiple consistent hash functions, one for the primary RAE and the others for the backups; the producers publish the data names both to the primary and the backup RAEs, while the consumers query the backup RAEs only when the primary RAE does not respond.

6. DISCUSSION

Next we describe a number of issues, orthogonal to the IC-DCN architecture, that arise when using the ICN paradigm of communication within the data center context.

6.1 Name Space Management

The controller's naming service in IC-DCN enables efficient management of the name spaces across different producers, which would be difficult to achieve under decentralized architectures. For example, suppose a producer X announces (claims) a name "/movie/action/", indicating it can provide all movie files under that prefix, and later, another producer Y joins the network and announces names of several individual movie files that are not in X, yet under the same prefix (e.g., "/movie/action/y1", "/movie/action/y2", $\cdots$). As a result, the name space earlier announced by X becomes incorrect, because it can result in the requests for movies in Y to be directed toward X.

The controller, as a collection point of all names announced by the producers, can effectively resolve the inconsistency and fix invalid names, by (re)allocating, fragmenting, or aggregating the name spaces, in a manner similar to IP address and domain name management in the Internet. For example in the context of above scenario, the names announced by producer X can be fragmented into the individual names of movie files, e.g., into "/movie/action/x1", "/movie/action/x2", $\cdots$, or allocated some other name space, e.g., "/movie/action/x/". Note that other cases are also possible too; for example, the names announced by a producer for individual data objects can be aggregated into a single name.

6.2 Dynamic Content and Caching Policies

Often a large portion of data in DCN are dynamically generated by, e.g., services, rather than served from static contents. The issue of how to name and serve the dynamic contents in the context of info-centric networking is handled elsewhere in the literature[9], e.g., [14, 6]. What is relevant to us, however, is the impact of dynamically generated data to the scalability of the forwarding and caching plane.

In particular, there is no benefit in caching dynamically generated data that are accessed only once, as it reduces the speed of the forwarding nodes due to cache lookup and it causes eviction of other data that might be accessed more than once. Similarly, there is little benefit in caching data whose size is very small, e.g., much smaller than the MTU. Hence, to further increase the forwarding and caching scalability, different caching policies can be associated with each interest packet, based on the size and the type of the data that it names.

In IC-DCN, the enforcement of such policies is implemented at the controller. When a routing request is made, the controller informs the consumer of the caching policy for the requested name. The consumer then uses a special bit in the interest packet to indicate whether the requested data object is cacheable, so all switches can skip the cache lookup step if the request is for an un-cacheable object.

7. RELATED WORK

The Data-Oriented Network Architecture (DONA) [10] provides a clean-slate ICN design, based on self-certifying names. An alternate design was laid out by Jacobsen et. al. [7] where communication is driven by the consumers of data. The basic design concepts introduced in the above proposals and the follow up work [10, 7, 5, 13, 14] include basic labeling and grouping of information, while providing a publish-subscribe service model. A perspective on the ICN research can be found in the recent work by Perino et. al [12]. While the current focus of the ICN community is on Internet-scale development and deployment, our work examines the feasibility of designing an ICN-based data center network. The nature of DCN applications combined with the self-contained environment in the DCN makes it a viable candidate to deploy a clean-slate ICN design.

The separation of control and data plane architecture has been first explored in the context of Internet routers [4], and more recently in the context of of software defined networking (SDN) [11, 2]. SDN advocates the need to separate the data and the control plane wherein a controller performs network services such as path computation, firewall and security functionality etc. Our IC-DCN design also incorporates the separation of control and data planes, but we differ in the way route computations are performed in the context of ICN. Specifically, routing is performed by the central controller using dynamic route computation algorithms and forwarding is done based on fixed-length labels.

Onix [9] provides a platform on top of which a network control plane can be implemented as a distributed system and provide network-wide management abstractions. Control planes written within Onix operate on a global view of the network, and use basic state distribution primitives provided by the platform. HyperFlow [15] is another distributed event-based control plane for OpenFlow. HyperFlow passively synchronizes network-wide views of OpenFlow controllers and localizes decision making to individual controllers. In contrast, our distributed controller design is specific to route computation and path assignment based on names. Onix and HyperFlow controllers on the other hand serve as a gen-

[9]The only assumption we need in IC-DCN is the presence of a proper naming scheme (e.g., announcing only the static, "routable" portion of the content names) and the application semantics to handle dynamic contents.

eral platform for development of scalable applications. In practice, a system such as the Onix controller could be used to build the IC-DCN platform.

8. CONCLUSION

In this paper, we presented an information-centric network architecture for data centers (IC-DCN). A distinctive feature of this architecture is the separation of the control plane from data plane, where the data forwarding is performed by the network entities based on path labels associated with a particular routing path determined by the network topology and data location. The proposed architecture is suitable for data center networks, in that it enables a scalable networking solution in DCN while facilitating the benefits of information-centric networking paradigm. We also showed additional benefits of IC-DCN in achieving better utilization of network resources and efficiently managing name spaces in a manner that would be otherwise difficult to implement with existing proposals for ICN. We are currently developing a research prototype of IC-DCN entities and algorithms introduced in this paper, and we are also investigating issues related to the support of dynamically generated data (e.g., Map/Reduce) within the proposed IC-DCN framework.

9. REFERENCES

[1] Scalable and adaptive internet solutions (SAIL). http://www.sail-project.eu/.

[2] M. Casado, T. Koponen, R. Ramanathan, and S. Shenker. Virtualizing the network forwarding plane. In *Proceedings of PRESTO*, 2010.

[3] C. Demetrescu and G. F. Italiano. A new approach to dynamic all pairs shortest paths. *J. ACM*, 51(6):968–992, Nov. 2004.

[4] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe. The case for separating routing from routers. In *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, FDNA, 2004.

[5] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *HotNets*, 2011.

[6] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard. VoCCN: Voice over content-centric networks. In *ReArch '09*.

[7] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. Briggs, and R. Braynard. Networking named content. In *CoNEXT*, 2009.

[8] V. Jiménez and A. Marzal. Computing the k shortest paths: A new algorithm and an experimental comparison. *Algorithm engineering*, pages 15–29, 1999.

[9] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: A distributed control platform for large-scale production networks. In *In Proc. OSDI*, 2010.

[10] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *Proceedings of the ACM SIGCOMM 2007*, 2007.

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2), Mar. 2008.

[12] D. Perino and M. Varvello. A reality check for content centric networking. In *ACM SIGCOMM workshop on Information-centric networking*, 2011.

[13] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *Proceedings of the Third International COST264 Workshop on Networked Group Communication*, NGC, 2001.

[14] S. Shanbhag, N. Schwan, I. Rimac, and M. Varvello. SoCCeR: Services over content-centric routing. In *ACM SIGCOMM workshop on Information-centric networking*, 2011.

[15] A. Tootoonchian and Y. Ganjali. Hyperflow: a distributed control plane for openflow. In *Proceedings of INM/WREN*, 2010.

[16] D. Verma. Simplifying network administration using policy-based management. *IEEE Network*, 16(2), 2002.